Ph.D. DISSERTATION

Graph and Hypergraph Matching in Computer Vision: Algorithms and Analysis

컴퓨터비전을 위한 그래프정합과 고차그래프정합:
새로운 알고리즘과 분석에 관한 연구

BY

JUNGMIN LEE

August 2016

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Abstract

Establishing image feature correspondences is fundamental problem in computer vision and machine learning research fields. Myriad of graph matching algorithms have been proposed to tackle this problem by regarding correspondence problem as a graph matching problem. However, the graph matching problem is challenging since there are various types of noises in real world scenario; *e.g.*, non-rigid motion, view-point change, and background clutter. The objective of this dissertation is to propose robust graph matching algorithms for feature correspondence task in computer vision and to investigate an effective graph matching strategy.

For the purpose, at first, two robust simulation based graph matching algorithms are introduced: the one is based on Random Walks simulation and the other is based on Markov Chain Monte Carlo sampling simulation. Secondly, two different graph matching formulations and their transformal relation are studied since equivalence between two formulations are not well studied in graph matching fields. It is demonstrated that conventional graph matching algorithms can solve both types of formulations by proposing conversion principle between two formulations. Finally, these whole statements are extended into hypergraph matching problem by introducing two robust hypergraph matching algorithms which are based on Random Walks and Markov Chain Monte Carlo, by relating two different hypergraph matching formulations, and by reinterpreting previous hypergraph matching algorithms into their counterpart formulations. Throughout chapters in this dissertation, comparative and extensive experiments verify characteristics of formulations, transformal relations, and algorithms. Synthetic graph matching problems as well as real image feature correspondence problems are performed in various and severe noise conditions.

Key words: Graph Matching, Hypergraph Matching, Graph Matching Formulations, Markov chain Monte Carlo, Data-Driven, Random Walks

Student number: 2008-20943

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Graph Matching Problem

1.1.1 Graph Matching for Computer Vision

Graph matching is one of most widely used tools in computer vision, machine learning, and pattern recognition [3] research area. It is also essential in various computer vision problems, such as in object recognition [4, 5], wide-baseline stereo [6], feature tracking [7], and shape matching [8, 9]. In particular, it has been used for establishing correspondences between two sets of features [1, 2] because a graphical model can naturally encode features and their relations into node and edge attributes. As widely known, the correspondence problem is effectively formulated as graph matching [9, 10, 1], where each graph is constructed with nodes that represent the features and edges that describe the relations between the features. Correspondences are established by determining a mapping between the two graphs which best preserves attributes of nodes and edges (see Fig. 1.1(a)). Unlike popular matching techniques for rigid motion such as RANSAC [11] and the iterative closest points (ICP) [12], graph matching effectively handles non-rigid deformation with appearance changes, and finds reliable matches by minimizing distortion between corresponding features and their relations. This strong advantage enables graph matching to handle challenging correspondence problems in real-world images. Unlike other popular matching methods that are based on appearances [13, 14] or strong parametric constraints [15], the graph matching technique effectively handles both geometric distortion as well as appearance variation [9, 1]. The recent resurgence of graph matching in computer vision [10, 16, 17, 1] has settled a general formulation of quadratic assignment that explicitly considers both local appearances and pairwise geometric relations.

1.1.2 Graph Matching Formulation

The goal of graph matching is to find the best correspondence between two graphs, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$ and $\mathcal{G}' = \{\mathcal{V}', \mathcal{E}', \mathbf{A}'\}$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges. We denote the node and edge attributes with $\mathbf{A}_{ii} \in \mathbb{R}^{n}_{+}$ (on diagonal elements) and $\mathbf{A}_{ij} \in \mathbb{R}^{n \times n}_{+}$ (on off-diagonal elements), respectively, where $n = |\mathcal{V}|$ is the number of nodes, $\mathbf{A}_{ii}$ is the attribute of i-th node $v_i \in \mathcal{V}$ and $\mathbf{A}_{ij}$ is the attribute of directed edge $e_{ij} \in \mathcal{E}$.

We analyze two representative formulations in graph matching, *i.e. adjacency*-based and *affinity*-based representations, and show their transformational relations into equivalent counterparts. We revisit these two families of graph matching formulations, analyze their relations, and propose efficient graph matching strategies. We propose the efficient way to solve large size graph matching problem by modifying previously introduced matching techniques. In general, the problem of graph matching belongs to the quadratic assignment problem (QAP), and a myriad of graph matching algorithms have been proposed with different types of formulations and characteristics in the literature [3, 28, 29]. In this dissertation, we classify the most popular families of recent graph matching formulations into two types of formulations: *affinity-based* [18, 19, 10, 16, 20, 21, 1, 24, 30] and *adjacency-based* [31, 32, 33, 34, 23, 25] formulations. The affinity-based formulation takes a given *affinity* score for each candidate correspondence between two graphs as its input, and adopts any predefined scores for node and edge correspondences. The adjacency-based formulation only takes an attribute *adjacency* value for nodes and edges on each graph. A *affinity* score for a node correspondence or an edge correspondence between two graphs is computed on-the-fly using those attributes in the formulation. In this sense, the affinity-based formulation is more general than the adjacency-based one. At the cost of losing its generality, however, the adjacency-based formulation obtains a substantial advantage in optimization efficiency. While these two types of formulations have existed together for a few decades and cover most of recent state-of-the-art graph matching methods, a comparative analysis between them has been rarely done in the literature.

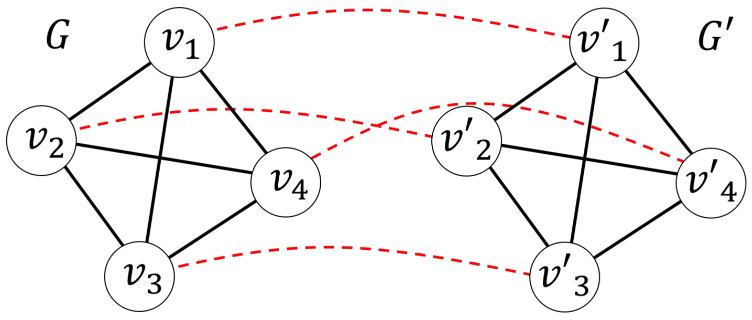
Adjacency-based formulation: Without loss of generality, we assume $n \leq n'$. Adjacency-based formulation finds the optimal mapping $\mathbf{X}^*$, which minimizes the difference between corresponding node and edge attributes

$$\mathbf{X}^* = \operatorname*{argmin}_{\mathbf{X}} \left|\left|\mathbf{A} - \mathbf{X}\mathbf{A}'\mathbf{X}^\intercal\right|\right|_F^2, \tag{1.1}$$

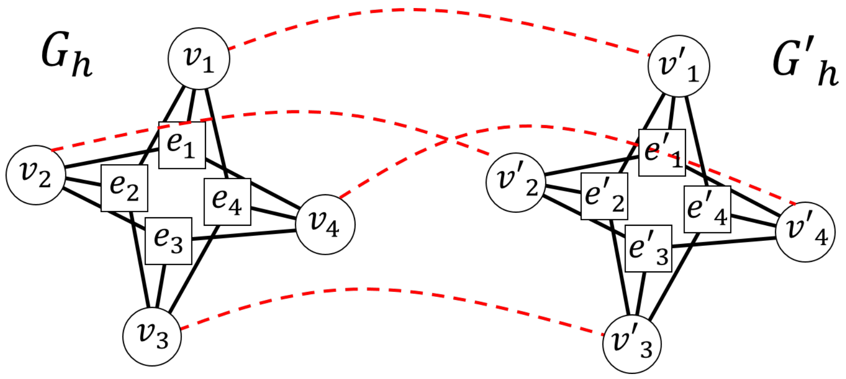
where $\mathbf{X} \in \Pi = \{\mathbf{X}|\mathbf{X} \in \{0,1\}^{n\times n'}, \mathbf{X}^\intercal\mathbf{1}_n \leq \mathbf{1}_{n'}, \mathbf{X}\mathbf{1}_{n'} = \mathbf{1}_n\}$ is constrained to be a one-to-one mapping. Simple linear algebra shows that the optimization problem in Eq.(1.1) is equivalent to the following maximization problem:

$$\mathbf{X}^* = \operatorname*{argmax}_{\mathbf{X}} \operatorname{Tr}\left(\mathbf{A}(\mathbf{X}\mathbf{A}'\mathbf{X}^\intercal)^\intercal\right) \tag{1.2}$$

Affinity-based formulation: Affinity-based formulation maximizes sum of



(a) ordinary graph matching



(b) hypergraph matching

Figure 1.1: Examples of graph and hypergraph matching. (a) Two ordinary (second order) graphs $\mathcal{G}$ and $\mathcal{G}'$ with 4 nodes and an example of their matching. Corresponding attributes of 4 nodes and 6 edges should be similar for the best matching. (b) Two hypergraphs $\mathcal{G}_h$ and $\mathcal{G}'_h$ with 4 nodes and an example of their matching. Note that e_1 connects v_1, v_2, and v_4 simultaneously, and so on. Corresponding attributes of 4 nodes ($\{v_1, v_2, v_3, v_4\}$ and $\{v'_1, v'_2, v'_3, v'_4\}$) and 4 hyperedges ($\{e_1, e_2, e_3, e_4\}$ and $\{e'_1, e'_2, e'_3, e'_4\}$) should be similar for the best matching.

affinity scores between corresponding graph attributes. It casts the graph matching problem into the following optimization problem:

$$\mathbf{X}^* = \operatorname*{argmax}_{\mathbf{X}} \sum_{\substack{(i,j)\in\mathcal{E}\\(a,b)\in\mathcal{E}'}} \mathbf{X}_{i,a}\mathbf{X}_{j,b} f(\mathbf{A}_{i,j}, \mathbf{A'}_{a,b}) \tag{1.3}$$

where the affinity function $f(\cdot,\cdot)$ measures similarity between two attributes. For node indices, $i, j \in \mathcal{V}$, we slightly abuse the notation ia to denote $(i + (a-1)n)$-th index, which corresponds to the (i, a)-component of $\mathbf{X}$. Since it is assumed that $f(\cdot,\cdot)$ is an arbitrary affinity function, affinity-based formulation can solve more general problems. We can encode affinity score using the affinity matrix:

$$\mathbf{M}_{ia,jb} = f(\mathbf{A}_{i,j}, \mathbf{A'}_{a,b}) \tag{1.4}$$

Then, Eq.(1.3) is equivalent to the following optimization problem:

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}} (\mathbf{x}^\intercal \mathbf{M}\mathbf{x}), \text{ s.t. } \mathbf{x} = \text{vec}(\mathbf{X}), \mathbf{X} \in \Pi \tag{1.5}$$

1.1.3 Extension to Hypergraph Matching

In the hypergraph matching problem, hyperedges naturally embed high-order relations of features while ordinary graphs can exploit only second order relations. In order to exploit high-order information beyond pairwise relations, several hypergraph matching algorithms are introduced which are based on high-order assignment formulation [20, 35]. The existing approaches, however, still suffer from severe distortion of the graphs (*e.g.*, deformation on graph attributes or existence of outliers nodes). Robustness to such distortion is of particular importance in a practical sense because real-world graph matching problems are widely exposed to deformation and outliers. For example, general image matching usually deals with deformed shapes and background features. This line of inquiry has further extended to hypergraph matching techniques, which overcome the limitation of pairwise relations and exploit high-order information [20, 35, 36]. There are several researches which focus on hypergraph matching problem [20, 35, 36, 2, 37]. Hyperedges are simultaneously connecting more than two nodes on hypergrphs, thus, more sophisticated and robust attributes can be embedded into attribute tensors (see Fig. 1.1(b)). To our best knowledge, this is the first work which related *affinity* formulation and *adjacency* formulation on hypergraph matching problem. Furthermore, we derive transformation rule of two formulations and reinterpret conventional hypergraph matching algorithms for the other type of formulation. Finally, performances of conventional approaches in both formulation are thoroughly compared in experiment section. Recently, various works are published to address hypergraph matching problem [20, 35, 36, 2]. Hypergraphs can embed more sophisticated information on their hyperedges, therefore, produce more robust performances on severe noises. In the proposed hypergraph matching formulation, multiple tensors of any orders can be integrated thus the formulation can exploit multiple types of feature relations. Instead of using the attribute $\mathbf{A}$ which is in the form of the matrix, we represent hyperedge attributes by employing tensor $\mathbf{T} \in \mathbb{R}_+^{n\times n\times n}$. Then we have two hypergraphs to be matched; $\mathcal{G}_h = \{\mathcal{V}, \mathcal{E}_h, \mathbf{T}\}$ and $\mathcal{G}'_h = \{\mathcal{V}', \mathcal{E}'_h, \mathbf{T}'\}$.

1.2 Outline of Dissertation

The main goal of this dissertation can be summarized in three folds. First, two novel and robust graph and hypergraph matching algorithms are introduced. One is random walks based graph matching algorithm and the other is based on stochastic sampling technique. Second, two different graph matching formulations are introduces and their relation is verified. Finally, before-mentioned two algorithms, two different formulations, and their relation are extended to hypergraph matching problem.

Chapter 2 proposes random walks based graph matching algorithm [1] and extends the algorithm for hypergraph matching problem [2]. The association graph is proposed on which nodes represent possible candidate matches between original two graphs, then random walks simulation is applied to find a set of nodes which are supporting each other most. The proposed random walks based algorithm adopt personalized jump scheme by which the 1-to-1 matching constraints are effectively applied during the simulation. Thorough experiments compares various graph matching algorithms on various noise conditions and demonstrate that the proposed algorithm shows state-of-the-art performance. Chapter 3 proposes Markov Chain Monte Carlo (MCMC) based graph and hypergraph matching algorithm [22]. MCMC theoretically guarantees that distribution of samples convergences to given target distribution. The proposed MCMC algorithm finds the state with the highest target energy value, therefore, the algorithm can maximize the objective function of the graph matching problem. In chapter 4, *adjacency* based formulation and *affinity* based formulation are introduced and their transformal relation is verified. Advantages and disadvantages of two formulations are discussed and reinterpretations of representative graph matching algorithms are introduced. Throughout Chapter 2–4, every algorithm, graph matching basic formulations, and transformal relation is extended to hypergraph matching problem. Random walks based hypergraph matching algorithm and MCMC based hypergraph matching algorithm are introduced in Chapter 2 and 3, respectively. Pros and cons between *adjacency* and *affinity* based formulations for hypergraph matching are discussed in Chapter 4.

Chapter 2

Graph Matching via Random Walks

2.1 Introduction

Establishing reasonable correspondences between two sets of features is one of the critical issues in machine learning and pattern recognition research fields. The correspondence problem can effectively be formulated as graph matching [9, 10, 1], where graph nodes represent the features and graph edges represent the relations between these nodes. After constructing two graphs, correspondences are established by finding a mapping between the two graphs which best preserves attributes between nodes and edges. Various approaches [18, 19, 10, 16, 21] are introduced to solve this graph matching problem, however, they still suffers on severe deformation and outlier noise conditions. This graph matching problem is extended to hypergraph matching problem, which overcome the limitation of pairwise relations and exploit high-order relation of node or features [20, 35, 36]. The existing hypergraph matching approaches, however, still have trouble with severe distortion of the graphs (*e.g.*, deformation noise of graph attributes or existence of outliers graphs). Acquiring robustness to such various types of noises is critical issue in a practical sense since graph matching problems are with more noise in real-world scenario [1].

This chapter introduces a robust graph matching algorithm which is based on the Random Walks (RW) model, and generalizes the algorithm for hypergraph matching with any order. The proposed RW-based algorithm adopts reweighting jumps for enforcing the one-to-one matching constraints which is critical issue in graph matching problem and achieves a noise-robust performance since RW algorithm updates confidence scores in probabilistic manner.

Three contributions are addressed in this chapter. First, it establishes a novel RW view on graph and hypergraph matching and provides bases for the RW interpretations of other methods [10, 16, 18, 35]. Second, in this view, a noise-robust graph and hypergraph matching algorithm that is inspired by the personalization strategy of web ranking [38]. Third, the quantitative comparisons are extensively performed with several state-of-the-art graph matching [10, 16, 21, 18, 19] and hypergraph matching algorithms [20, 35, 36]. The experiments not only show the performance of the proposed algorithm but also facilitates a comprehensive study of recent graph and hypergraph matching algorithms.

2.1.1 Related Works

In general, graph matching problem is proven to be NP-hard, therefore, researchers have tried to tackle this problem by proposing various relaxed approximations [3]. In [39], the graph matching problem is formulated as a constrained integer quadratic programming (IQP), however, the complexity of its optimization is non-polynomial.

Graduated Assignment (GAGM) [18] is proposed by Gold and Rangarajan; GAGM solves the graph matching problem by a scheduled annealing technique. Successive Projection (SPGM) [19] iteratively projects the current solution to approximation on the convex solution space of the graph matching constraints. Leordeanu and Hebert proposed the Spectral Matching (SM) [10]. They introduce a spectral relaxation of the graph matching formulation while Cour *et al.*extend this Spectral Matching relaxation for satisfying the Affine Constraint (SMAC) [16]. Integer Projected Fixed Point (IPFP) [21] algorithm is proposed by Leordeanu *et al.*. IPFP iteratively optimizes the IQP graph matching problem in the discrete solution space. There are several approaches [20, 35, 36] which extend the ordinary graph matching to hypergraph matching problem. Hyper-Graph Matching (HGM) [20] is proposed by Zass and Shashua. HGM introduces probabilistic interpretation of graph and hypergraph matching by relating Kronecker product and the hypergraph matching formulation. Tensor Matching (TM) [35] is introduced by Duchenne *et al.*, and can be interpreted as an extension of SM for hypergraph matching. Chertok and Keller propose Singular Value Decomposition based hypergraph matching algorithm (SVD) [36] which calculates the rank-1 approximation by approximated SVD of the given affinity tensor.

The proposed RW graph matching algorithm in this chapter adopts the IQP formulation of [18, 10, 16, 22] and its extensions [20, 35, 36]. Our probabilistic interpretation follows an RW view of [40, 41]. Conventional RW-based methods [42, 43] have limitation that they adopt RW to calculate a signature of each graph and their objective formulations are not generalized for widely used IQP. However, the proposed algorithm introduces an association graph and interprets the graph matching problem as a node selection problem. In addition, the matching constraints are satisfied by adopting the personalized jumps [38] thus noise-robust graph matching performances are achieved in various conditions.

2.2 Problem Formulation

In this chapter, we follow the formulations of graph and hypergraph matching from [1, 2]. The goal of graph or hypergraph matching problem is to find the best correspondences between two attributed graphs (or hypergraphs) $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$ and $\mathcal{G}' = \{\mathcal{V}', \mathcal{E}', \mathbf{A}'\}$, where $\mathcal{V}$, $\mathcal{E}$, and $\mathbf{A}$ represent a set of nodes, a set the edges (or hyperedges), and corresponding attributes, respectively. A solution can be represented by an assignment matrix $\mathbf{X} \in \{0,1\}^{n \times n'}$, where n and n' are the numbers of nodes in $\mathcal{G}$ and $\mathcal{G}'$, respectively. An element the assignment matrix $\mathbf{X}_{i,a} = 1$ when $v_i \in \mathcal{V}$ matches to $v'_a \in \mathcal{V}'$ and $\mathbf{X}_{i,a} = 0$ otherwise. In this thesis, column-wise vectorized replica of $\mathbf{X}$ is represented by $\mathbf{x} \in \{0,1\}^{nn'}$. Graph matching problems can be mathematically formulated by finding the assignment vector (or assignment matrix) $\mathbf{x}^*$ which maximizes an objective score function $S(\mathbf{x})$, as the following equation.

$$\begin{aligned} \mathbf{x}^* &= \operatorname*{argmax}_{\mathbf{x}} S(\mathbf{x}) \\ s.t.\ \mathbf{X}\mathbf{1}_{n'} &\preceq \mathbf{1}_n,\ \mathbf{X}^\intercal \mathbf{1}_n \preceq \mathbf{1}_{n'} \end{aligned} \tag{2.1}$$

where the constraints of Eq.(2.1) indicate the one-to-one matching between $\mathcal{G}$ and $\mathcal{G}'$ and make $\mathbf{X}$ an binary *assignment matrix*.

2.2.1 Graph Matching Formulation

In the ordinary graph $\mathcal{G}$, node $v_i \in \mathcal{V}$ and edge $e_{ij} \in \mathcal{E}$ have their associated node attribute $\mathbf{A}_{ii}$ (on the diagonal elements of $\mathbf{A}$) and edge attribute $\mathbf{A}_{ij}$, respectively.

Graph matching between $\mathcal{G}$ and $\mathcal{G}'$ is to find node correspondences, which best preserve the attribute relations (see Fig. 1.1(a)). The similarity function $f(\cdot,\cdot)$ measures compatibility score between two attributes. For example, $f_v(\mathbf{A}_i, \mathbf{A'}_a)$ measures node similarity between v_i and v_a and $f_e(\mathbf{A}_{ij}, \mathbf{A'}_{ab})$ measures edge similarity between e_{ij} and e'_{ab}.

The affinity matrix $\mathbf{M}$ is introduced which embeds similarity scores from $f_v(\cdot)$ and $f_e(\cdot)$. A non-diagonal element $\mathbf{M}_{ia,jb} = f_e(\mathbf{A}_{ij}, \mathbf{A'}_{ab})$ contains a pairwise similarity of two correspondences (v_i, v'_a) and (v_j, v'_b), whereas a diagonal term $\mathbf{M}_{ia,ia} = f_v(\mathbf{A}_i, \mathbf{A'}_a)$ represents a unary similarity of a correspondence (v_i, v'_a). With the assignment vector $\mathbf{x}$ and basic calculations of linear algebra, the objective score function that accumulates all the relevant similarity values is defined as the following:

$$S(\mathbf{x}) = \mathbf{x}^\intercal \mathbf{M}\mathbf{x}. \tag{2.2}$$

2.2.2 Hypergraphs Matching Formulation

Several hypergraph matching algorithms [20, 35, 36] are introduced to overcome the limitations of second order similarity measure by embedding high-order similarity information. Hyperedges in a hypergraph are able to connect more than two nodes simultaneously (see Fig. 1.1(b)). Although all hyperedges in Fig. 1.1(b) are with third order, there is no limitation on number of connected node for each hyperedge. Same as the ordinary graph matching, the goal of hypergraph matching is to establish node correspondences with most similar corresponding high-order attributes.

In this subsection, hypergraph matching formulation is introduced by following conventions from [2]. The quadratic formulation in the previous subsection can be generalized to high-order formulation for hypergraph matching. The kth-order similarity function f_k is introduced as a generalization of f_2 and f_1 by defining $f_k(\mathbf{T}_{p_1,\cdots,p_k}, \mathbf{T'}_{q_1,\cdots,q_k})$ measures the similarity between two comparing high-order attributes $\mathbf{T}_{p_1,\cdots,p_k}$ and $\mathbf{T'}_{q_1,\cdots,q_k}$ which are attributes of kth-order hyperedges $e_{p_1,\cdots,p_k}$ and $e'_{q_1,\cdots,q_k}$. For a simple representation, we define an instance of graph matching $\mathcal{M}$ as $\mathcal{M} \subset \mathcal{V} \times \mathcal{V}'$. Then, without loss of generality, $f_k(\mathbf{T}_{p_1,\cdots,p_k}, \mathbf{T'}_{q_1,\cdots,q_k})$ can be alternatively represented using the correspondences $\mathbf{m}_{w_1} = (v_{p_1}, v'_{q_1}), \cdots, \mathbf{m}_{w_k} = (v_{p_k}, v'_{q_k})$, as follows [2]:

$$
\begin{aligned}
& f_k(\mathbf{m}_{w_1}, \cdots, \mathbf{m}_{w_k}) \\
& = \begin{cases} f_k(\mathbf{T}_{p_1,\cdots,p_k}, \mathbf{T'}_{q_1,\cdots,q_k}) & \text{if } e_{p_1,\cdots,p_k} \in \mathcal{E} \text{ and } e'_{q_1,\cdots,q_k} \in \mathcal{E}' \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{2.3}
$$

which means that the similarity of the k-tuple of correspondences is equivalent to the similarity of the two kth-order hyperedges, if both of the hyperedges $e_{p_1,\cdots,p_k}$ and $e'_{q_1,\cdots,q_k}$ exist. Note that $f_k(\mathbf{m}_{w_1}, \cdots, \mathbf{m}_{w_k})$ is an invariant measure under any permutation of $\{\mathbf{m}_{w_1}, \cdots, \mathbf{m}_{w_k}\}$.

For a given hypergraph matching instance $\mathcal{M}$, the generalized hypergraph matching score function can be formulated as the following [2]:

$$
\begin{aligned}
S(\mathcal{M}) = \lambda_\delta & \sum_{(\mathbf{m}_{w_1},\cdots,\mathbf{m}_{w_{\delta-1}},\mathbf{m}_{w_\delta})} f_\delta(\mathbf{m}_{w_1}, \cdots, \mathbf{m}_{w_{\delta-1}}, \mathbf{m}_{w_\delta}) \\
& + \lambda_{\delta-1} \sum_{(\mathbf{m}_{w_1},\cdots,\mathbf{m}_{w_{\delta-1}})} f_{\delta-1}(\mathbf{m}_{w_1}, \cdots, \mathbf{m}_{w_{\delta-1}}) \\
& \cdots \\
& + \lambda_1 \sum_{(\mathbf{m}_{w_1})} f_1(\mathbf{m}_{w_1})
\end{aligned}
\tag{2.4}
$$

where $(\mathbf{m}_{w_1}, \cdots, \mathbf{m}_{w_k})$ represents a set of k correspondences from $\mathcal{M}$, and λ_k is the weight of the kth-order similarity measure. The maximum order δ denotes the highest order among all the hyperedges involved in hypergraph matching.

Following the tensor representation [35], kth-order similarity function can be embedded into the k-dimensional affinity tensor $\mathbf{H}^{(k)}$ as the following:

$$\mathbf{H}^{(k)}_{w_1,\cdots,w_k} = f_k(\mathbf{m}_{w_1},\cdots,\mathbf{m}_{w_k}). \tag{2.5}$$

Here, the superscript of $\mathbf{H}$ is the dimension of the tensor, and the subscript of $\mathbf{H}$ is its element index. Since f_k is invariant under any permutation, $\mathbf{H}^{(k)}$ naturally becomes a super-symmetric tensor, (*i.e.*, invariant under any permutation of $\{w_1,\cdots,w_k\}$). Using an assignment vector $\mathbf{x}$, the score function of Eq.(2.4) is then expressed as follows;

$$\begin{aligned} S(\mathbf{x}) = \lambda_\delta & \sum_{(w_1,\cdots,w_{\delta-1},w_\delta)} \mathbf{H}^{(\delta)}_{w_1,\cdots,w_{\delta-1},w_\delta}\mathbf{x}_{w_1}\cdots\mathbf{x}_{w_{\delta-1}}\mathbf{x}_{w_\delta} \\ & + \lambda_{\delta-1} \sum_{(w_1,\cdots,w_{\delta-1})} \mathbf{H}^{(\delta-1)}_{w_1,\cdots,w_{\delta-1}}\mathbf{x}_{w_1}\cdots\mathbf{x}_{w_{\delta-1}} \\ & \cdots \\ & + \lambda_1 \sum_{(w_1)} \mathbf{H}^{(1)}_{w_1}\mathbf{x}_{w_1} \end{aligned} \tag{2.6}$$

where $\mathbf{x}_{w_k}$ represents a single correspondence $\mathbf{m}_{w_k} \in \mathcal{M}$. The score function denotes the summation of all similarity values related to the assignment vector $\mathbf{x}$. Using the n-mode tensor multiplication [35, 44], Eq.(2.6) can be reduced to a simple form:

$$\begin{aligned} S(\mathbf{x}) &= \sum_{k=1}^{\delta} \lambda_k \mathbf{H}^{(k)} \times_1 \mathbf{x} \cdots \times_k \mathbf{x} \\ &= \sum_{k=1}^{\delta} \lambda_k \mathbf{H}^{(k)} \times_{1:k} \mathbf{x}. \end{aligned} \tag{2.7}$$

where $\times_k$ means the multiplication along with the k-th dimension. In this chapter, for simple representation, we define the following: $\mathbf{H}^{(k)} \times_a \mathbf{x} \times_{a+1} \mathbf{x} \cdots \times_b \mathbf{x} = \mathbf{H}^{(k)} \times_{a:b} \mathbf{x}$. Eq.(2.7) extends and generalizes Eq.(2.2) for arbitrary orders. Given $\delta = 2$, Eq.(2.7) is reduced to Eq.(2.2). Combined with Eq.(2.1), the score function of Eq.(2.7) consists in a generalized hypergraph matching formulation, which effectively embeds high-level constraints or invariances. In the following, the matching of conventional graphs is presented in Section 2.3, and it is then generalized for hypergraphs in Section 2.4.

2.3 Graph Matching via Random Walks

By constructing an association graph $\mathcal{G}^{\times} = (\mathcal{V}^{\times}, \mathcal{E}^{\times}, \mathbf{A}^{\times})$, the problem of graph matching between two ordinary graphs $\mathcal{G}$ and $\mathcal{G}'$ can be interpreted in an RW view [1]. Given the affinity matrix $\mathbf{M}$, each candidate correspondence $(v_i, v'_a) \in \mathcal{V} \times \mathcal{V}'$ is considered as a node $v_{ia} \in \mathcal{V}^{\times}$, and its associated affinity score $\mathbf{M}_{ia,jb}$ is considered as the attribute $\mathbf{A}^{\times}_{ia,jb} \in \mathbf{A}^{\times}$ of the edge $e^{\times}_{ia,jb} \in \mathcal{E}^{\times}$ (see Fig. 2.1) [1]. With the definition of the association graph, graph matching between $\mathcal{G}$ and $\mathcal{G}'$ can be interpreted as node selection problem on $\mathcal{G}^{\times}$. To solve this problem, the statistics of the RW is adopted; this is widely used technique for calculating the ranking of cites in Web [40, 41].

2.3.1 Random Walks for Graph Matching

In conventional RW simulation, random walker on a single node travels to one of its connected node with the probability that is proportional to its connected edge weight [45]. Given an edge weight matrix $\mathbf{W}$, row-stochastic transition matrix $\mathbf{P}$ can be derived by $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$, where $\mathbf{D}$ is diagonal matrix with all node degrees. Then, with the current RW distribution probability $\mathbf{x}^{(t)}$, RW distribution probability can be simulated according to the following update rule: $\mathbf{x}^{(t+1)\intercal} = \mathbf{x}^{(t)\intercal}\mathbf{P}$. To apply

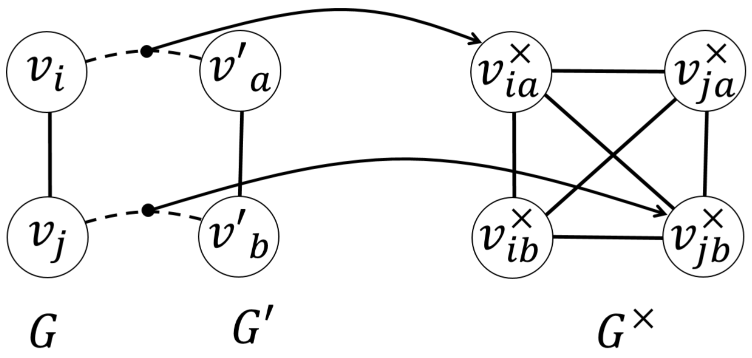


Figure 2.1: An example of graph matching and association graph. For given two graphs $\mathcal{G}$ and $\mathcal{G}'$ with two nodes, there are four possible candidate matches. Thus, the association graph $\mathcal{G}^{\times}$ becomes a single graph with four nodes, in which each edge encodes the pairwise compatibility of two candidate correspondences.

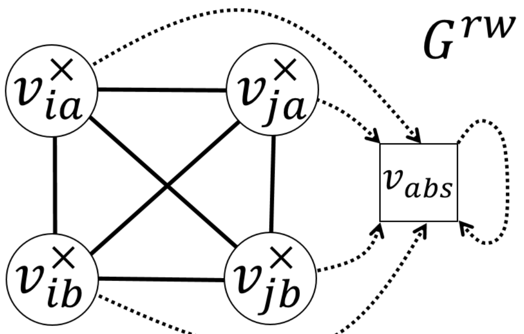


Figure 2.2: The augmented association graph for RW. The association graph $\mathcal{G}^{rw}$ for the affinity-preserving RW has an absorbing node that can soak the random walkers from every node. Note that random walker on v_{abs} is not able to travel to the other node.

RW on the graph matching problem, the affinity matrix $\mathbf{M}$ is used as the weight matrix $\mathbf{W}$. Therefore, the resulting RW transition matrix is $\mathbf{P} = \mathbf{D}^{-1}\mathbf{M}$, where $\mathbf{D}_{i,i} = \sum_j \mathbf{M}_{j,i}$. By applying RW using transition matrix $\mathbf{P}$ which is row-stochastic matrix, random walker on any node has outgoing probabilities which are sum to one; This phenomenon is so-called *internet democracy* [41, 46, 1, 2]. This conventional stochastic normalization, however, has a critical drawback for solving graph matching problems. In association graph $\mathcal{G}^{\times}$ from Fig. 2.1, for example, there are two correct nodes (candidate matches in the original problem) as well as two incorrect nodes. However, all four nodes have the same amount of outgoing probability and outgoing probabilities from incorrect nodes may cause errors in RW simulation.

To overcome this limitation, an absorbing node v_{abs} is augmented to the association graph $\mathcal{G}^{\times}$ which is defined as $\mathcal{G}^{rw}$ in Fig. 2.2. The augmented absorbing node v_{abs} is designed to soak probability from the node v_i with the amount of $d_{\max} - d_i$ while $d_i = \mathbf{D}_{i,i}$ and $d_{\max} = \max_i d_i$. Note that a random walker in v_{abs} is unable to move toward the other node [47]. A correct node in $\mathcal{G}^{rw}$, for example, has higher chance to have higher edge weights with other correct nodes, therefore, its corresponding degree d_i should be with higher value which means that a random walker on v_i has smaller chance to move toward v_{abs}. An incorrect node, on the other hand, may be with smaller d_i, therefore, a random walker easily goes to v_{abs}. This is called *affinity-preserving* RW and transition matrix $\mathbf{P}$ is derived as the following [1, 2]:

$$\mathbf{P} = \begin{pmatrix} \mathbf{M}/d_{\max} & \mathbf{1} - \mathbf{d}/d_{\max} \\ \mathbf{0}^{\intercal} & 1 \end{pmatrix},$$

$$\begin{pmatrix} \mathbf{x}^{(t+1)\intercal} & x_{abs}^{(t+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{x}^{(t)\intercal} & x_{abs}^{(t)} \end{pmatrix} \mathbf{P}, \tag{2.8}$$

where the degree vector is defined as $\mathbf{d}_{ia} = d_{ia}$, $\mathbf{1}$ and $\mathbf{0}$ represents all 1s and 0s, respectively. In the proposed RW simulation, any random walker eventually gets stuck in the absorbing node v_{abs} which makes the stationary distribution as $(\mathbf{0}^\intercal\ 1)^\intercal$. Therefore, it is required to define *quasi-stationary* distribution [1, 2] to achieve reasonable probability distribution of $\mathbf{x}$ for the graph matching problem.

$$\bar{\mathbf{x}}^{(t)} = \frac{\mathbf{x}^{(t)}}{1 - x_{abs}^{(t)}}, \tag{2.9}$$

where $\mathbf{x}$ is quasi-stationary if $\bar{\mathbf{x}}^{(t+1)} = \bar{\mathbf{x}}^{(t)}$.

The stationary distribution $\mathbf{x}$ of the affinity-preserving RW can be efficiently calculated by using simple power iteration method. However, $\mathbf{x}$ should be an assignment vector to be a solution of graph matching problem. One way to make $\mathbf{x}$ satisfying the constraints is a simple greedy manner discretization scheme which is introduced in [10]. In optimal sense, the Hungarian algorithm [48] is recommended instead of the greedy way.

2.3.2 Reweighting Jumps for Graph Matching

In the previous subsection, affinity-preserving RW is introduced. However, the matching constraints in Eq.(2.1) is not enforced during iterations of RW procedure. To reflect the 1-to-1 matching constraints during iterations, personalization approach is adopted [38, 46]. For a given personalized vector $\mathbf{r}$, a random walker moves along with its connected edges with probability α or jumps to any other node with probability $1-\alpha$, where α is the weighting factor between walking or jumping. RW update rule 2.8 is modified for personalization as the following [1, 2]:

$$\left(\mathbf{x}^{(t+1)\intercal}\ \ x_{abs}^{(t+1)}\right) = \alpha \left(\mathbf{x}^{(t)\intercal}\ \ x_{abs}^{(t)}\right) \mathbf{P} + (1-\alpha)\mathbf{r}^\intercal, \tag{2.10}$$

where $\mathbf{r}$ is the *personalized* vector which guides a random walker's jumps in RW iterations. The proposed RW graph matching algorithm effectively enforcing a random walker to obeying the matching constrains by making the personalized vector $\mathbf{r}$ satisfying the constraints. In the proposed Algorithm 2, the personalized vector $\mathbf{r}$ is updated on every iteration of RW simulation and is named as *reweighting* vector. The proposed reweighting procedure consists of two steps: inflation and bistochastic normalization. The inflation step suppresses small values in $\mathbf{x}$ and amplifies large values in $\mathbf{x}$ which makes unreliable correspondences less important in RW iterations. Bistochastic Sinkhorn [49] normalization is adopted for making inflated state $\mathbf{x}$ satisfy one-to-one matching constraints [18]. Sinkhorn step consists of iterative row-wise normalization and column-wise normalization which leads any non-negative square matrix to a bistochastic matrix. Finally, reweighting RW update rule becomes as the following [1, 2]:

$$\begin{aligned}&\left(\mathbf{x}^{(t+1)\intercal} \quad x_{abs}^{(t+1)}\right) \\ &= \alpha \left(\mathbf{x}^{(t)\intercal} \quad x_{abs}^{(t)}\right) \mathbf{P} + (1-\alpha) \left(f_R(\mathbf{x}^{(t)\intercal}\mathbf{M})^\intercal \quad 0\right),\end{aligned} \tag{2.11}$$

where $f_R(\cdot)$ represents the reweighting function which enforces the 1-to-1 constraints. Note that the steady state of the proposed reweighted RW can be calculated by iterative updates by following 2.11 unlike conventional RWs [38, 46]. For producing a graph matching solution, calculating x_{abs} is not necessary, thus Eq.(2.11) is reduced for the graph matching problem as the following:

$$\mathbf{x}^{(t+1)\intercal} = \frac{\alpha}{d_{\max}} \mathbf{x}^{(t)\intercal}\mathbf{M} + (1-\alpha) f_R(\mathbf{x}^{(t)\intercal}\mathbf{M})^\intercal. \tag{2.12}$$

The objective of the proposed algorithm is to find quasi-stationary distribution by using the update rule of Eq.(2.12) and the algorithms is summarized in Algorithm 1 [1].

Algorithm 1: Reweighted Random Walks for Graph Matching

```
Input: affinity matrix $\mathbf{M}$, parameter $\{\alpha\}$
Output: assignment vector $\mathbf{x}$
Set the maximum degree $d_{\max} \leftarrow \max_i \sum_j \mathbf{M}_{i,j}$;
Initialize the starting probability $\mathbf{x}$ as uniform;
repeat
    $\bar{\mathbf{x}}^\mathsf{T} \leftarrow \mathbf{x}^\mathsf{T}\mathbf{M}$;
    $\mathbf{y} \leftarrow$ re-weight $\bar{\mathbf{x}}$ by Algorithm 2;
    ( Random walking with reweighted jumps )
    $\mathbf{x}^\mathsf{T} \leftarrow \frac{\alpha}{d_{\max}}\bar{\mathbf{x}}^\mathsf{T} + (1-\alpha)\mathbf{y}^\mathsf{T}$;
    $\mathbf{x} \leftarrow \mathbf{x}/\|\mathbf{x}\|_1$;
until $\mathbf{x}$ converges;
Discretize $\mathbf{x}$ by the matching constraints;
```

Algorithm 2: Reweighting for Graph Matching

```
Input: vector $\mathbf{x}$, parameter $\{\beta\}$
Output: reweighted vector $\mathbf{y}$
( Inflation )
$\mathbf{y}^\mathsf{T} \leftarrow \exp(\beta\mathbf{x}/\max\mathbf{x})$;
( Sinkhorn )
repeat
    normalize rows by $\mathbf{y}_{ai} \leftarrow \mathbf{y}_{ai}/\sum_i \mathbf{y}_{ai}$;
    normalize columns by $\mathbf{y}_{ai} \leftarrow \mathbf{y}_{ai}/\sum_a \mathbf{y}_{ai}$;
until $\mathbf{y}$ converges;
$\mathbf{y} \leftarrow \mathbf{y}/\sum \mathbf{y}_{ai}$;
```

2.4 Hypergraph Matching via Random Walks

In this section, the proposed algorithm is extended to hypergraph matching based on a hypergraph RW. As explained in Section 2.2.2, the hypergraph matching problem aims to maximize the score of Eq.(2.7) under the assignment constraints of Eq.(2.1).

2.4.1 Hypergraph Random Walks

A random walker in an ordinary graph travels along with edges by following the transition matrix $\mathbf{P}$. In this section, RW transition is generalized for hypergraphs [50]. To generalize RW for hypergraphs, two steps are required: one is hyperedge selection and the other is node selection (see Fig. 2.3) [2]. In hyperedge selection step, a random walker selects one hyperedge from its connected hyperedges by considering weighted of hyperedges. Given a selected hyperedge, one of its node is selected under uniform distribution and the random walker travels to that selected node. The transition probability matrix $\mathbf{P}$ of Eq.(2.8) and RW update rule can be extended for hypergraph RW as the following [2]:

$$\begin{aligned}\mathbf{x}^{(t+1)} &= \mathbf{P}^{(k)} \times_2 \mathbf{x}^{(t)} \cdots \times_k \mathbf{x}^{(t)} \\ &= \mathbf{P}^{(k)} \times_{2:k} \mathbf{x}^{(t)}. \end{aligned} \tag{2.13}$$

Note that $\mathbf{P}^{(k)}$ is kth-oder transition tensor, however, it is assumed that a hypergraph consists of hyperedges with any order. Therefore, we defined generalized RW update as the following:

$$\mathbf{x}^{(t+1)} = \sum_k \lambda_k \mathbf{P}^{(k)} \times_{2:k} \mathbf{x}^{(t)}. \tag{2.14}$$

Note that λ_k is a importance weight for kth-order hyperedge and λ is subject to $\lambda_k \in [0,1]$ and $\sum_k \lambda_k = 1$ [2]. Eq.(2.14) corresponds to higher-order power iteration

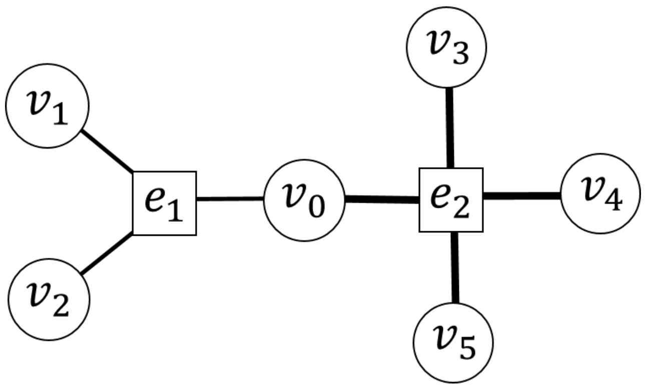


Figure 2.3: An example of RW on a hypergraph. A random walker on v_0 first selects one hyperedge e_1 or e_2 according to the hyperedge weights. Assume that e_2 has higher weight than e_1, for example, the random walker has higher chance to select e_2 and travels to v_3, v_4, or v_5.

of [51].

2.4.2 Reweighting Jumps for Hypergraph Matching

As same as in ordinary graph matching, the association graph is constructed for hypergraph matching. However, in hypergraph matching, the association graph becomes hypergraph since more than two nodes (candidate matches in the graph matching problem) should be compared simultaneously. Thus, the association hypergraph $\mathcal{G}_h^{\times} = \{\mathcal{V}^{\times}, \mathcal{E}_h^{\times}, \mathbf{T}^{\times}\}$. Note that $\mathcal{E}_h^{\times}$ is the set of hyperedges and $\mathbf{T}^{\times}$ is a multi-dimensional tensors for embedding attributes of $\mathcal{V}^{\times}$, $\mathcal{E}_h^{\times}$. For a given kth-order hyperedge $e^{\times}_{w_1,\cdots,w_k}$ in $\mathcal{G}_h^{\times}$ which connects $\{v^{\times}_{w_1}, \cdots, v^{\times}_{w_k}\}$, there are k matches related: $m_{w_1} = (v_{p_1}, v'_{q_1})$, $\cdots$, $m_{w_k} = (v_{p_k}, v'_{q_k})$. Then, kth-order similarity function $f_k(m_{w_1}, \cdots, m_{w_k})$ measures a compatibility score of k matches which is recorded to $\mathbf{T}^{\times}_{w_1,\cdots,w_k}$ in Eq.(2.3). After construction of the association hypergraph $\mathcal{G}_h^{\times}$, hypergraph RW can be simulated, thus it is possible to solve graph matching problem by selecting nodes from $\mathcal{G}_h^{\times}$. Note that the attribute tensor $\mathbf{T}^{\times}$ becomes equivalent to the set of affinity tensors $\{\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(\sigma)}\}$ in Eq.(2.5).

As same as in ordinary graph matching problem, the affinity-preserving hypergraph RW is realized by augmenting an *absorbing node* with the same manner in Section 2.3.1. To calculate the degree d_w of the node v_w, all affinities which are associated with v_w should be considered [2]:

$$\begin{aligned} d_w &= \sum_{k=2}^{\delta} \sum_{w_2,\cdots,w_k} \lambda_k \mathbf{H}^{(k)}_{w,w_2,\cdots,w_k} + \lambda_1 \mathbf{H}^{(1)}_w \\ &= (\sum_{k=1}^{\delta} \lambda_k \mathbf{H}^{(k)} \times_{2:k} \mathbf{1})_w, \end{aligned} \tag{2.15}$$

where the subscript $(\cdot)_w$ denotes the w-th element value in a vector. With the maximum degree $d_{\max} = \max_w d_w$, it is able to construct *augmented* hypergraph RW as same to the ordinary RW case in which the absorbing node soaks the weight $d_{\max} - d_w$ from each node $v_w \in \mathcal{V}^{\times}$ [2].

Extending Eq.(2.8) to generalized hypergraph affinity-preserving RW using Eq.(2.14) becomes as the following:

$$\begin{aligned} \mathbf{x}^{(t+1)} &= \frac{1}{d_{\max}} \sum_{k=1}^{\delta} \lambda_k \mathbf{H}^{(k)} \times_{2:k} \mathbf{x}^{(t)}, \\ x^{(t+1)}_{\text{abs}} &= 1 - \frac{1}{d_{\max}} \mathbf{x}^{(t)T} \mathbf{d} \end{aligned} \tag{2.16}$$

With the same definition of the *quasi-stationary* state in Eq.(2.9), it is possible to discard the second update rule in Eq.(2.16). Similar to [51, 35], the proposed affinity-preserving RW updates of the affinity tensor $\mathbf{H}$ can be considered as rank-1 approximation of the tensor. Eq.(2.16) can be interpreted as power iteration for the affinity tensor of the hypergraph matching problem and produces an approximated solution for Eq.(2.7) under $\|\mathbf{x}\| = 1$.

The matching constraints of Eq.(2.1) is also imposed on the proposed hypergraph affinity-preserving RW by employing the reweighting jump. As the jump is

Algorithm 3: Generalized Reweighted Random Walks Matching

```
Input: affinity tensors $\lambda_\delta \mathbf{H}^{(\delta)}, \lambda_{\delta-1}\mathbf{H}^{(\delta-1)}, \cdots, \lambda_1 \mathbf{H}^{(1)}$, reweighting factor $\alpha$
Output: assignment vector $\mathbf{x}$
Set $d_{\max} \leftarrow \max_w (\sum_{k=1}^{\delta} \lambda_k \mathbf{H}^{(k)} \times_{2:k} \mathbf{1})_w$;
Initialize the starting probability $\mathbf{x}$ as uniform;
repeat
    $\bar{\mathbf{x}} \leftarrow \sum_{k=1}^{\delta} \lambda_k \mathbf{H}^{(k)} \times_{2:k} \mathbf{x}$;
    $\mathbf{y} \leftarrow$ re-weight $\bar{\mathbf{x}}$ by Algorithm 2;
    ( Random walking with reweighted jumps )
    $\mathbf{x} \leftarrow \frac{\alpha}{d_{\max}} \bar{\mathbf{x}} + (1-\alpha)\mathbf{y}$;
    $\mathbf{x} \leftarrow \mathbf{x} / \|\mathbf{x}\|_1$;
until $\mathbf{x}$ converges;
Discretize $\mathbf{X}$ by the matching constraints;
```

performed on nodes (not on hyperedges), the process of reweighting jumps is exactly the same as in the case of conventional graphs [1, 2]. Again, the reweighting function $f_R(\cdot)$ is adopted which is already introduced in Section 2.3.2, the reweighted hypergraph affinity-preserving RW is formulated as the following and summarized in Algorithm 3:

$$\begin{aligned}\mathbf{x}^{(t+1)} &= \frac{\alpha}{d_{\max}}\sum_{k=1}^{\delta}\lambda_k\mathbf{H}^{(k)}\times_{2:k}\mathbf{x}^{(t)} \\ &\quad +(1-\alpha)f_R(\sum_{k=1}^{\delta}\lambda_k\mathbf{H}^{(k)}\times_{2:k}\mathbf{x}^{(t)}).\end{aligned} \tag{2.17}$$

2.5 Experiments

In this section, thorough experiments are performed for demonstrating performances of the proposed method through four types of experiments [1, 2]: (i) graph matching on synthetically generated random graphs, (ii) 2D point matching on synthetically generated random points, (iii) point matching task on the CMU House image sequence[1], and (iv) feature matching on real images.

On the ordinary graph matching problem, various state-of-the-art algorithms are compared: SM [10], SMAC [16], HGM [20], IPFP [21], GAGM [18], and SPGM [19]. On the hypergraph matching problem, HGM [20], TM [35], and EHOM [36] are compared. Since HGM [20] provides generalized graph matching algorithm, HGM is tested on both ordinary graph matching and hypergraph matching problem. Authors of TM [35] provide two versions of their algorithm, therefore, both versions are tested and compared (see details of two versions in [35][2]). The proposed algorithm for the

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

[2]`http://www.di.ens.fr/~duchenne/`

ordinary graph matching problem is termed RRWM, and the proposed hypergraph matching algorithm is named RRWHM. For hypergraph matching algorithms, only 3rd-order affinity tensors are utilized since not all hypergraph algorithms are able to cover hypergraphs with any order. Note that the proposed RRWHM [2] is able to cover hypergraphs with any order.

For SM[3], SMAC[4], HGM[5], and TM, the publicly available codes from the authors were utilized with little adaptation, while GAGM, IPFP, SPGM, EHOM were implemented [1, 2]. MATLAB is used for implementing all algorithms. For every noise setting, 30 experients are performed and their averages are reported. The same affinity matrix or tensor was shared as the input of each algorithms and the Hungarian algorithm was commonly used at the final discretization step for all methods [1, 2]. Parameters of RRWM and RRWHM are empirically set to $\alpha = 0.2$ and $\beta = 30$ for the best performance.

2.5.1 Random Graph Matching

This section presents comparative evaluations on the matching of synthetic random graph and hypergraphs. Random graphs are synthetically generated to compare performances of graph and hypergraph matching methods by following the protocol of [1, 2].

For ordinary graph matching problems, two graphs $\mathcal{G}$, $\mathcal{G}'$ are generated as the following: At first, the graph attribute $\mathbf{A}$ is randomly generated using uniform distribution $\mathcal{U}(0,1)$; *i.e.*, $\mathbf{A} \sim \mathcal{U}(0,1)^{n\times n}$. Note that $n = n' = n_{in} + n_{out}$, where n_{in} and n_{out} are inlier and outlier numbers, respectively. To synthesize more difficult

[3]`http://86.34.14.245/code.php`

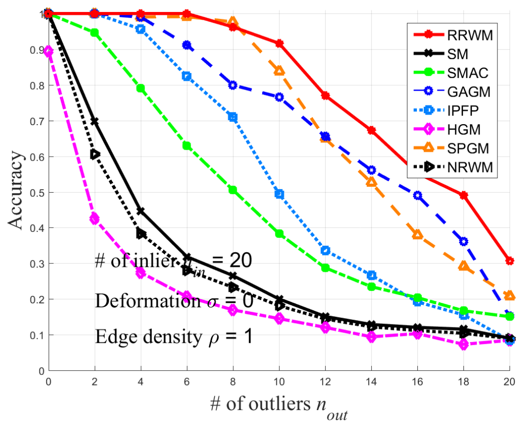
[4]`http://www.seas.upenn.edu/~timothee/`

[5]`http://www.cs.huji.ac.il/~zass/`

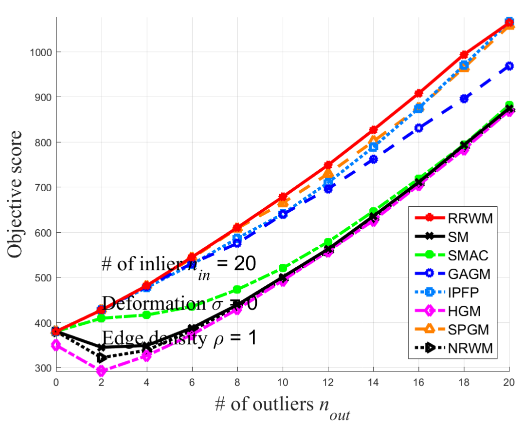
matching problem, we assume that there is no attribute values for nodes so we set $\mathbf{A}_{i,i} = 0$. Then, $\mathbf{A}'$ is generated by following $\mathbf{A}' = \mathbf{A} + \mathbf{N}$ and $\mathbf{N} \sim \mathcal{N}(0, \sigma)^{n' \times n'}$ for simulating deformation noises using Gaussian distribution $\mathcal{N}$. Finally, outlier noises are imposed by overwriting noises to $\mathbf{A}$ and $\mathbf{A}'$ with $\mathbf{A}_{1:n_{out},1:n_{out}} \sim \mathcal{U}(0,1)^{n_{out} \times n_{out}}$ and $\mathbf{A}'_{1:n_{out},1:n_{out}} \sim \mathcal{U}(0,1)^{n_{out} \times n_{out}}$. For making $\mathcal{G}$ and $\mathcal{G}'$ sparse, every edges in two graphs are deleted with the probability of $1 - \rho$ [1]. $\mathbf{A}$ and $\mathbf{A}'$ are randomly permuted for preventing one method accidentally achieving high score.

Then, the affinity matrix $\mathbf{M}$ is calculated using $\mathcal{G}$ and $\mathcal{G}'$ by $\mathbf{M}_{ia,jb} = \exp(-||\mathbf{A}_{i,j} - \mathbf{A}'_{a,b}||^2/\sigma_s)$. The scaling parameter σ_s is empirically set to 0.15 for the best performances. The accuracy was measured by the number of detected true matches over the total number of ground truths, whereas the objective score was measured by computing $S(\mathbf{x})$ [1, 2]. The results are reported in Figs. 2.4 and 2.5. In Fig. 2.4, the number of inliers n_{in}, deformation noise σ, and edge density ρ are fixed to 20, 0, and 1, respectively, while the number of outlier n_{out} varies from 0 to 20. In Fig. 2.5(a) and (b), the number of inliers n_{in}, the number of outliers n_{out}, and edge density ρ are fixed to 20, 0, and 1, respectively, while deformation noise σ varies from 0 to 0.4. Finally, in Fig. 2.5(c), the number of inliers n_{in}, the number of outliers n_{out}, and deformation noise σ are fixed to 20, 10, and 0.1, respectively, while edge density ρ varies from 0.3 to 1.

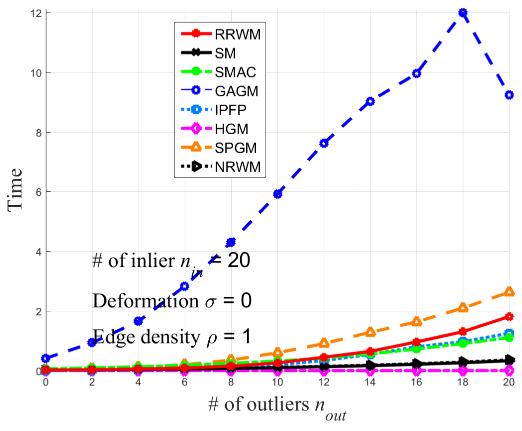
Throughout all experiments in Figs. 2.4 and 2.5, the proposed RRWM outperforms all the other algorithms under three different types of noises in the sense of accuracy and objective score. NRWM denotes the proposed algorithm without absorbing node and reweighting jumps, and shows effects of the proposed augmented node and reweighting scheme in the graph matching problems. By comparing SM and RRWM, the effect of reweighting jumps is revealed since RRWM is equivalent



(a) matching accuracy

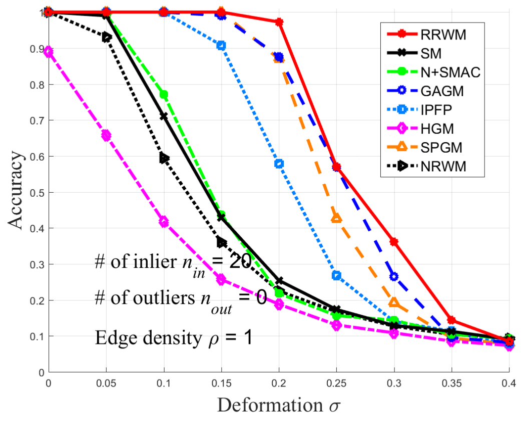


(b) objective score

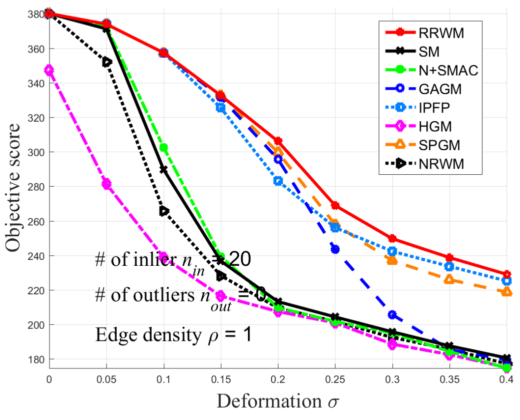


(c) execution time

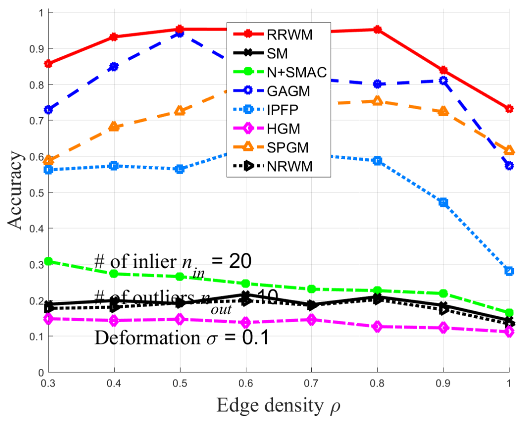
Figure 2.4: Performance comparison according to outlier noises on matching of synthetic ordinary graphs [1]. The number of inliers n_{in}, deformation noise σ, and edge density ρ are fixed to 20, 0, and 1, respectively, while the number of outlier n_{out} varies from 0 to 20.



(a) matching accuracy



(b) objective score



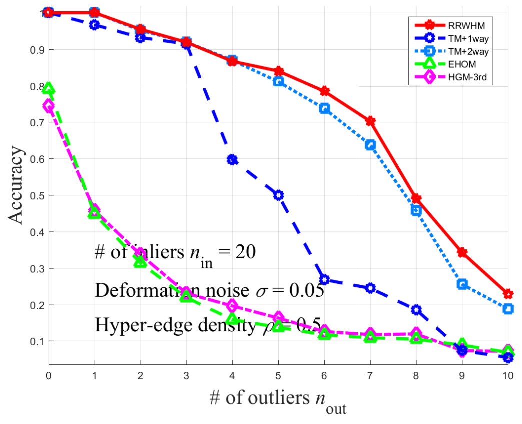
(c) matching accuracy

Figure 2.5: Performance comparison according to deformation noises and graph densities on matching of synthetic ordinary graphs [1]. (a)-(b) The number of inliers n_{in}, the number of outliers n_{out}, and edge density ρ are fixed to 20, 0, and 1, respectively, while deformation noise σ varies from 0 to 0.4. (c) The number of inliers n_{in}, the number of outliers n_{out}, and deformation noise σ are fixed to 20, 10, and 0.1, respectively, while edge density ρ varies from 0.3 to 1.

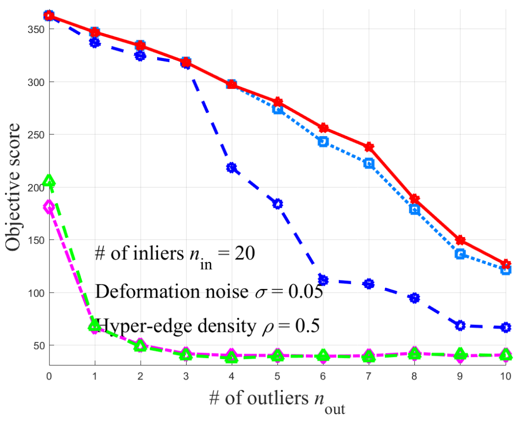
to SM without reweighting jumps (*i.e.*, RW only with the absorbing node). Considering that RRWM, GAGM, and SPGM are robust algorithms and that they all commonly focus on incorporating the 1-to-1 constraints during their iterations, the importance of obeying the matching constraints are revealed.

For hypergraph matching comparison, the experimental setup was almost the same as that of ordinary graph matching; the main difference was the use of 3rd-order hyperedges instead of 2nd-order edges. First, attribute tensors are generated by following $\mathbf{T} \sim \mathcal{U}(0,1)^{n\times n\times n}$ and $\mathbf{T}' \sim \mathcal{U}(0,1)^{n'\times n'\times n'}$. Next, deformation noises are generated by $\mathbf{N} \sim \mathcal{N}(0,\sigma)^{n_{in}\times n_{in}\times n_{in}}$. We overwrite $\mathbf{T}'$ by $\mathbf{T}'_{1:n_{in},1:n_{in},1:n_{in}} = \mathbf{T}_{1:n_{in},1:n_{in},1:n_{in}} + \mathbf{N}$ and then permute indices of $\mathbf{T}'$ in the same manner as ordinary graph case. Then, the affinity tensor $\mathbf{H}$ is calculated by $\mathbf{H}_{ia,jb,kc} = \exp(-||\mathbf{T}_{i,j,k} - \mathbf{T}'_{a,b,c}||^2/\sigma_s)$ where the scaling parameter σ_s is again set to 0.15. Since the affinity tensor $\mathbf{H}$ is 3-dimensional tensor, its computational complexity grows to $\mathcal{O}(n^6)$. This complexity problem is overcome by following sparsifying strategy from [35]. The total number of affinity scores are set similar to that of ordinary graph matching problem for fair comparison.

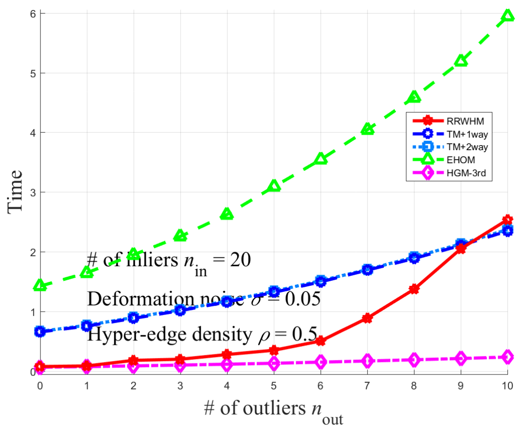
Throughout all experiments in Figs. 2.6 and 2.7, the proposed RRWHM outperforms all the other algorithms under three different types of noises in the sense of accuracy and objective score. The proposed RRWHM outperformed all other methods of hypergraph matching in both accuracy and objective score by adopting the affinity-preserving property and the reweighting scheme which enable the avoidance of adverse effects from outliers and deformation [2]. In Fig. 2.6, the number of inliers n_{in}, deformation noise σ, and edge density ρ are fixed to 20, 0.05, and 0.5, respectively, while the number of outlier n_{out} varies from 0 to 10. In Fig. 2.7(a) and (b), the number of inliers n_{in}, the number of outliers n_{out}, and edge density ρ are



(a) matching accuracy

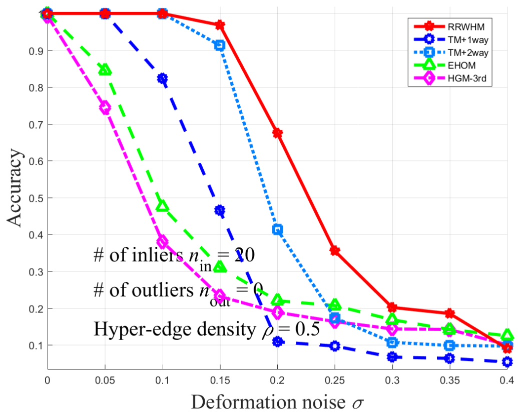


(b) objective score

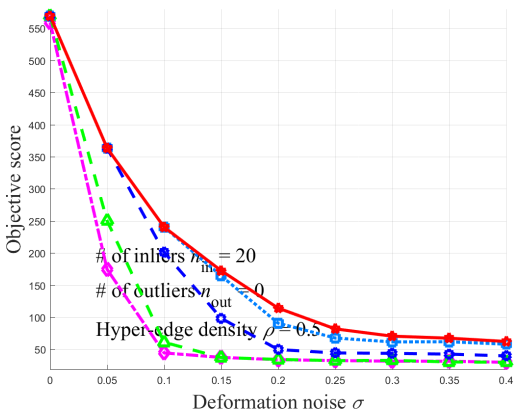


(c) execution time

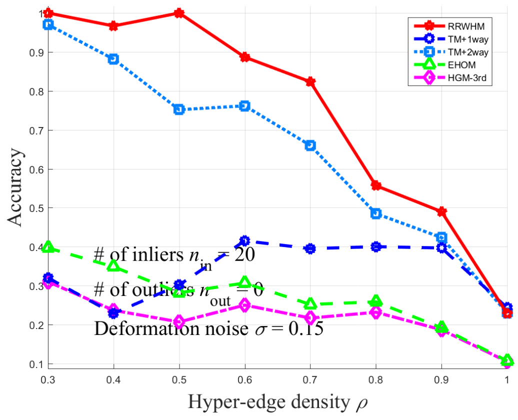
Figure 2.6: Performance comparison according to outlier noises on matching of synthetic hypergraphs [2]. The number of inliers n_{in}, deformation noise σ, and edge density ρ are fixed to 20, 0.05, and 0.5, respectively, while the number of outlier n_{out} varies from 0 to 10.



(a) matching accuracy



(b) objective score



(c) matching accuracy

Figure 2.7: Performance comparison according to deformation noises and graph densities on matching of synthetic hypergraphs [2]. (a)-(b) The number of inliers n_{in}, the number of outliers n_{out}, and edge density ρ are fixed to 20, 0, and 0.5, respectively, while deformation noise σ varies from 0 to 0.4. (c) The number of inliers n_{in}, the number of outliers n_{out}, and deformation noise σ are fixed to 20, 0, and 0.15, respectively, while edge density ρ varies from 0.3 to 1.

fixed to 20, 0, and 0.5, respectively, while deformation noise σ varies from 0 to 0.4. Finally, in Fig. 2.7(c), the number of inliers n_{in}, the number of outliers n_{out}, and deformation noise σ are fixed to 20, 0, and 0.15, respectively, while edge density ρ varies from 0.3 to 1.

2.5.2 Synthetic Point Matching

Synthetic random graph matching in the previous subsection is unbiased graph matching problem, however, has one limitation that graph and hypergraph matching algorithms are unable to be compared simultaneously. To compare graph and hypergraph matching algorithms with the same problem, this subsection presents comparative evaluations on random point set matching [10, 16, 20, 35]. To build synthetic point sets, n_{in} inlier points were randomly generated using $\mathcal{N}(0,1)$ in the first 2D domain. Then, each point in the first domain was copied to the second domain and Gaussian deformation noise $\mathcal{N}(0,\sigma^2)$ are added to the second domain. By adding n_{out} random 2D points from $\mathcal{N}(0,1)$ to each domain, outlier noise is simulated. The resultant points in two domains became nodes in graphs and hypergraphs $\mathcal{G}$ and $\mathcal{G}'$, respectively. In this setup, both graph matching algorithms (SM, RRWM) and hypergraph matching algorithms (RRWHM, HGM, TM, EHOM) are tested on the same point set problem.

For calculating 2nd-order affinity matrix $\mathbf{M}$, a difference between the Euclidean distances [2] of two point pairs was adopted;

$$\mathbf{M}_{ia,jb} = \exp(-(\|\mathbf{x}_i - \mathbf{x}_j\| - \|\mathbf{x}'_a - \mathbf{x}'_b\|)^2/\sigma_s), \tag{2.18}$$

where the scale parameter σ_s is set to 0.1. This distance-based measure is widely utilized in other works [9, 10, 16, 17, 2].

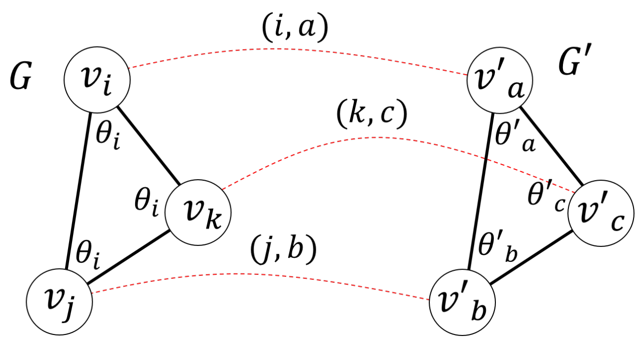


Figure 2.8: Calculation of 3rd-order similarity for hypergraph matching. Two triangles are formed from three candidate correspondences (i, a), (j, b), and (k, c). There are three corresponding angles in each triangle, therefore, it is possible to calculate similarity between two triangles by comparing three corresponding angles. This measure is invariant to translation, rotation, and scale changes [2].

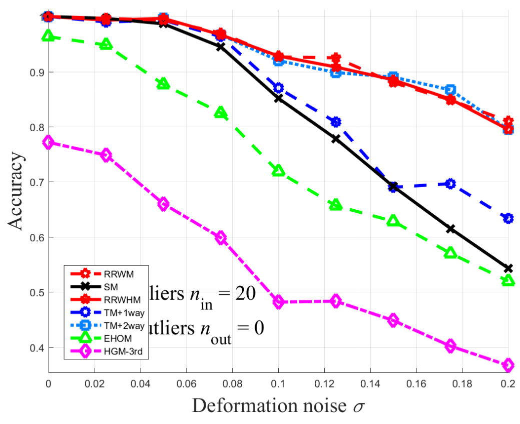
For calculating 3rd-order affinity tensor **H**, as illustrated in Fig. 2.8, a sum of sine value differences between corresponding angles [20, 35, 2] was adopted:

$$d_{ia} = |\sin(\theta_i) - \sin(\theta'_a)|,$$

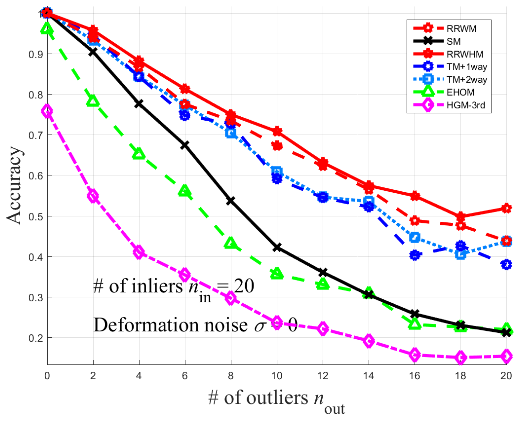
$$\mathbf{H}_{ia,jb,kc} = \exp\Big(- (d_{ia} + d_{jb} + d_{kc})/\sigma_s\Big), \tag{2.19}$$

where θ_i and θ_a denote the node angles of candidate correspondence (i, a). The scale parameter σ_s was set to 0.5 for the best performance. Unlike the distance-based similarity of Eq.(2.18), this angle-based 3rd-order similarity of Eq.(2.19) has an invariance up to scale as well as translation and rotation.

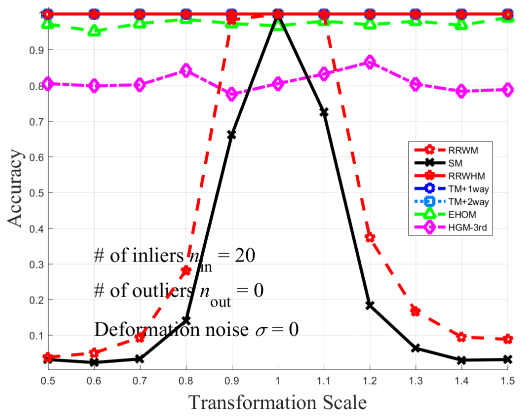
The experimental results on the point matching problem are reported in Figs. 2.9. In deformation test of Fig. 2.9(a), the number of inliers n_{in} and the number of outliers n_{out} are fixed to 20, and 0, respectively, while deformation noise σ varies from 0 to 0.2. In outlier noise test of Fig. 2.9(b), the number of inliers n_{in} and deformation noise σ are fixed to 20, and 0, respectively, while the number of outliers n_{out} varies from 0 to 20. Finally, in scale variation test of Fig. 2.9(c), the number



(a) varying deformation without outliers



(b) varying outliers without deformation



(c) varying scale differences

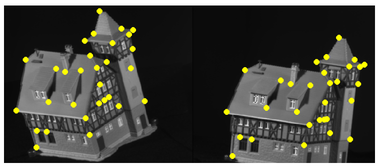
Figure 2.9: Performance comparison on point set matching problem according to deformation, outlier, and scale changes [1, 2]. (a) The number of inliers n_{in} and the number of outliers n_{out} are fixed to 20, and 0, respectively, while deformation noise σ varies from 0 to 0.2. (b) The number of inliers n_{in} and deformation noise σ are fixed to 20, and 0, respectively, while the number of outliers n_{out} varies from 0 to 20. (c) The number of inliers n_{in}, the number of outliers n_{out}, and deformation noise σ are fixed to 20, 0, and 0, respectively, while transformation scale varies from 0.5 to 1.5.

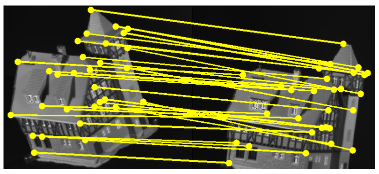
of inliers n_{in}, the number of outliers n_{out}, and deformation noise σ are fixed to 20, 0, and 0, respectively, while transformation scale varies from 0.5 to 1.5. RRWHM and RRWM performs best in Figs. 2.9(a) and (b), however, RRWM shows fragile performance on scale variation test in Figs. 2.9(c). Actually, every ordinary graph matching algorithms are failed to achieve reasonable performances on scale variation test since the affinity measure of the ordinary graph matching is not robust to scale changes.

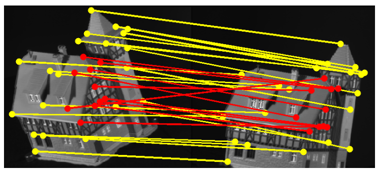
2.5.3 Image Sequence Matching

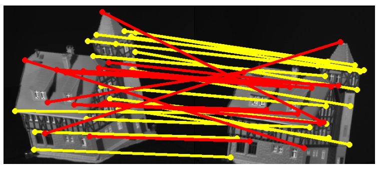
In this subsection, point matching experiment was performed on the CMU House image sequence; this is one of the most popular benchmark datasets. To achieve 2D points from image sequence, 30 landmark interest points are manually labeled along with all frames. In this experiment, as same as the previous point matching problem, the same similarity measures of Eq.(2.18) and (2.19) were used. Note that σ_s for Eq.(2.18) is set to 2500 in this experiment since the range of 2D points is changed. The performance of algorithms are measure by matching images with varying sequence gaps: the larger the sequence gap between the frames, the larger the relative deformation, and the more difficult the matching [1, 2].

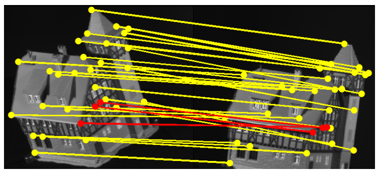
The experimental results of CMU point set matching problem is reported in Fig. 2.10(g). The proposed RRWHM and RRWM shows best performance in this experiment while TM-2way also records almost perfect performance.

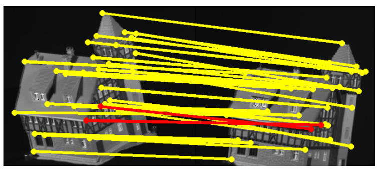

(a) a test pair example

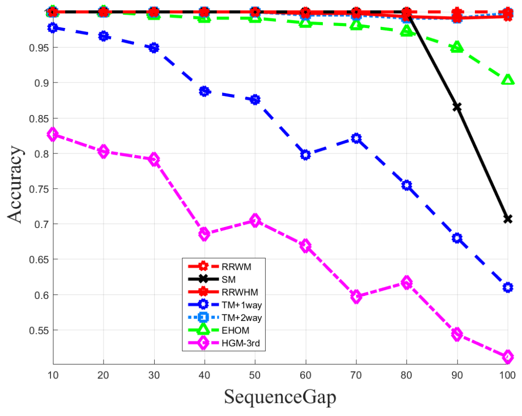

(b) RRWM & RRWHM (30/30)


(c) SM 2nd (20/30)


(d) HGM 3rd (17/30)


(e) EHOM 3rd (27/30)


(f) TM-2way 3rd (28/30)



(g) the CMU House: accuracy vs. sequence gap

Figure 2.10: Point set matching performance of CMU house images sequence [1, 2]. (a) An example of image pairs with labeled points. (b) RRWHM finds all 30 point pairs correctly. (c) SM finds 20 correct matches out of 30. (d) HGM finds 17 correct matches out of 30. (e) EHOM finds 27 correct matches out of 30. (f) TM finds 28 correct matches out of 30. (g) Point matching accuracies according to size of the sequence gap. The sequence gap varies from 10 to 100. These figures are best viewed in color.

2.5.4 Image Feature Matching

The final experiment in this chapter is a real image matching problem using local region features. 30 image pairs are collected from Caltech-101[6] and MSRC datasets. Unlike the previous CMU point matching problem, the MSER features [6] are automatically detected and two graphs are generated by taking MSER features as nodes. There is a complexity problem since detected MSER feature are usually more than 100 in numbers. Therefore, to sparsify the affinity matrix or tensor, candidate correspondences are pruned according to their first order similarities (see details in [1, 2]). To evaluate performance of algorithms in real image matching experiment, the ground truths correspondences are manually labeled.

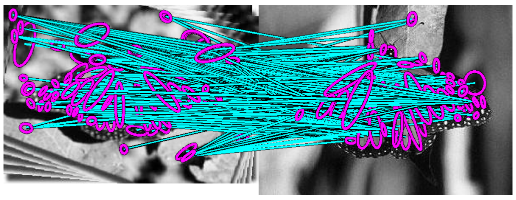
Unlike the similarity measures in the previous experiments, robust similarity measures based on properties of local features are used in this experiment as the following: For 2nd-order similarity measure between two pairs of features (i, a) and (j, b), the Symmetric Transfer Error (STE) used in [8] was adopted. The nature of STE measure is that every feature region has its descriptive ellipse and similarity can be derived by comparing ellipse transformations [8, 4]. For a feature point pair (i, a), it is able to calculate ellipse transformation $\mathcal{T}_{i,a}$ which transforms the ellipse of point i to the that of point a. Then the similarity between two feature correspondences is calculated as the following (details in [8]):

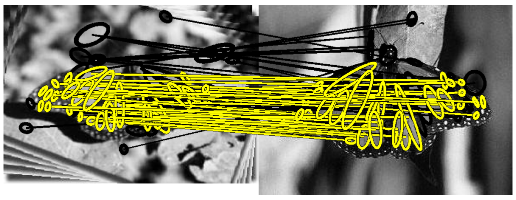
$$d_{jb|ia} = \frac{1}{2}\Big(\|(\mathbf{x'}_b - \mathbf{x'}_a) - \mathcal{T}_{ia}(\mathbf{x}_j - \mathbf{x}_i)\| + \|(\mathbf{x}_j - \mathbf{x}_i) - \mathcal{T}_{ia}^{-1}(\mathbf{x'}_b - \mathbf{x'}_a)\|\Big), \quad (2.20)$$

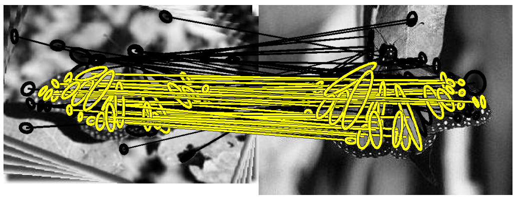
where $\mathcal{T}^{-1}$ denote the inverse transformation and $\mathbf{x}$ denotes feature location in images. Based on this, the 2nd-order STE similarity $\mathbf{M}_{ia;jb}$ is defined as the follow-

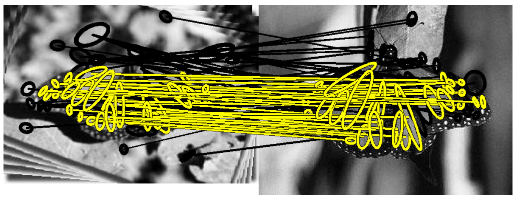
[6]`http://www.vision.caltech.edu/Image_Datasets/Caltech101/`

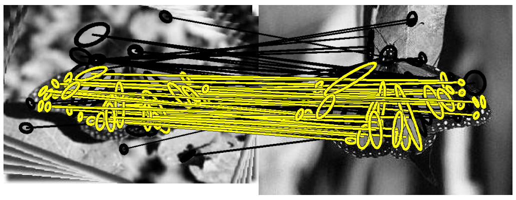
(a) input images

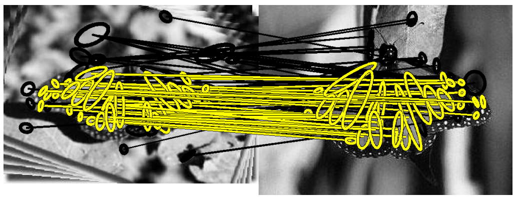

(b) initial matches


(c) RRWM 50/66


(d) RRWHM 49/66


(e) GAGM 45/66


(f) SM 47/66


(g) TM 46/66

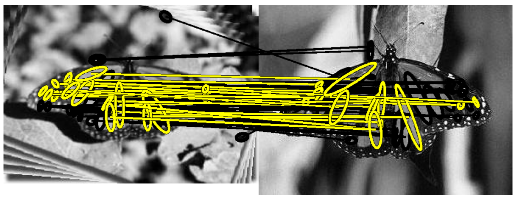
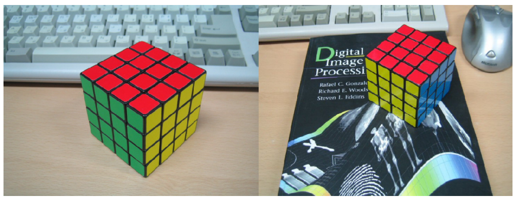
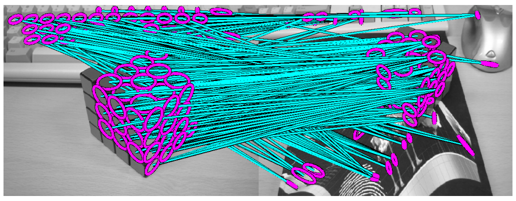

(h) HGM 26/66

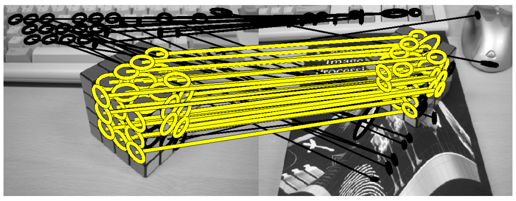
Figure 2.11: Example results of real image matching on a butterfly image pair. (a) Input image pair. (b) Initial candidate matches after pruning. (c)-(h) Matching results of RRWM, RRWHM, GAGM, SM, TM, and HGM. Correct correspondences are with yellow color. This figure is best viewed in color.



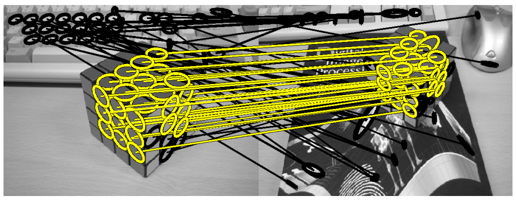
(a) input images



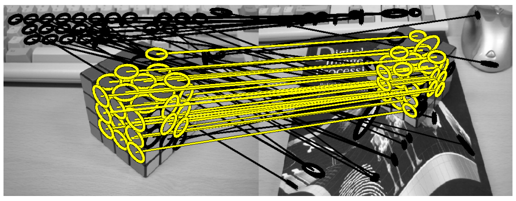
(b) initial matches



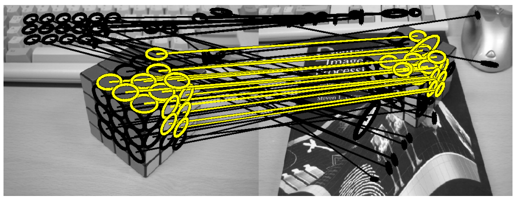
(c) RRWM 27/27



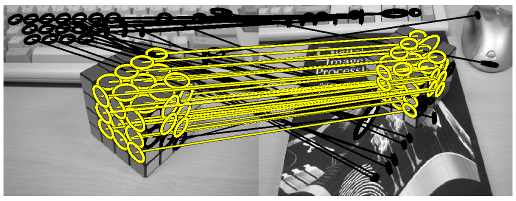
(d) RRWHM 26/27



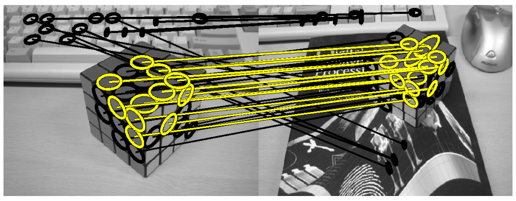
(e) GAGM 25/27



(f) SM 16/27



(g) TM 27/27



(h) HGM 13/27

Figure 2.12: Example results of real image matching on a cube image pair. (a) Input image pair. (b) Initial candidate matches after pruning. (c)-(h) Matching results of RRWM, RRWHM, GAGM, SM, TM, and HGM. Correct correspondences are with yellow color. This figure is best viewed in color.

ing [8]:

$$\mathbf{M}_{ia,jb} = \exp\Big(- (d_{jb|ia} + d_{ia|jb})/\sigma_s \Big), \tag{2.21}$$

where the resulting Eq.(2.21) is invariant to affine changes.

In [2], the 3rd-order STE similarity measure in introduced which is the extension of Eq.(2.21). For given three candidate correspondences (i, a), (j, b) and (k, c), the conditional STE is defined as the following:

$$d_{jb,kc|ia} = (d_{jb|ia} + d_{kc|ia})/2. \tag{2.22}$$

Based on this, the 3rd-order STE similarity $\mathbf{H}_{ia,jb,kc}$ is designed as the following [2]:

$$\mathbf{H}_{ia,jb,kc} = \exp\Big(- \max(d_{jb,kc|ia}, d_{ia,kc|jb}, d_{ia,jb|kc})/\sigma_s \Big). \tag{2.23}$$

Note that all three conditional STE measures $d_{jb,kc|ia}$, $d_{ia,kc|jb}$, and $d_{ia,jb|kc}$ should be smaller for higher affinity score of $\mathbf{H}_{ia,jb,kc}$.

In this experiment, σ_s is empirically set to 25 for the best performance. Examples of feature matching problem are demonstrated in Figs. 2.11 and 2.12. Besides, in Table 2.1, average accuracies and relative scores along with all 30 image pairs are reported. 2nd STE stands for 2nd-order STE affinity measure of Eq.(2.21) while 3rd STE stands for 3rd-order STE affinity measure of Eq.(2.23). In addition, the distance-comparing similarity measure of Eq.(2.18) and the angle-comparing similarity measure of Eq.(2.19) are also compared. In a matching accuracy sense, 3rd STE is the best measure for the feature matching problem and RRWHM is recommended as the best hypergraph matching algorithm. In this experiment, because both the 2nd-order STE and the 3rd-order STE similarities are invariant up to affine transformation, the accuracy difference between them was not as significant as the difference between the distance-based and the angle-based similarities. In general, hypergraph matching could benefit more from better high-order measures.

Table 2.1: Performances of various graph and hypergraph algorithms on the real image dataset of 30 pairs.

Measure	**Algorithm**			
2nd STE	RRWM	SM	SMAC	GAGM
Accuracy (%)	**69.77**	65.20	54.40	66.98
Rel. score (%)	**97.54**	86.19	70.89	94.18
3rd STE	RRWHM	HGM	TM-1W	TM-2W
Accuracy (%)	**72.64**	46.76	69.13	70.96
Rel. score (%)	**95.16**	22.60	81.08	80.74
2nd Distance	RRWM	SM	SMAC	GAGM
Accuracy (%)	24.00	22.68	24.65	**27.96**
Rel. score (%)	95.66	87.00	72.80	**95.82**
3rd Angle	RRWHM	HGM	TM-1W	TM-2W
Accuracy (%)	54.08	32.09	**54.65**	49.80
Rel. score (%)	**97.25**	10.02	80.75	76.46

2.6 Conclusion

In this chapter, a general graph matching formulation for graph matching and hypergraph matching is introduced. Also a novel RW view [41, 38] algorithm is proposed. The 1-to-1 matching constraints are critical in robust graph matching problem and the proposed algorithm effectively incorporates those constraints by adopting reweighing jumps. For solving the hypergraph matching problem in RW sense, the proposed method is extended to generalized hypergraph RW. Extensive experiments on both synthetic and real problems demonstrated that the proposed algorithm outperforms current state-of-the-art methods for graph matching [10, 16, 21, 18, 19] and hypergraph matching [20, 35, 36] in the presence of outliers and deformation.

Chapter 3

Graph Matching via Markov Chain Monte Carlo

3.1 Introduction

Graph matching problem is one of widely used techniques for problems regarding pattern recognition and computer vision research fields. Usually, correspondence problem (*e.g.*, feature correspondence between two images) can be interpreted as graph matching. Recent approaches [18, 19, 10, 16, 20, 21, 22, 1, 24, 23, 25, 26, 27] try to solve the graph matching problem by formulating it as Integer Quadratic Programming (IQP).

Since the graph matching problem is proven to be NP-hard problem, various approaches try to tackle this problem in approximated or relaxed ways. Conventional graph matching algorithms, however, suffers two major limitations when there exists large perturbation on the graphs due to outliers or deformation noises. First, local minima problem arises since most of conventional approaches are deterministic solver which calculates their solution in one-shot. There is no escaping mechanism in deterministic approaches. Second, relaxations are erroneous IQP graph matching formulation imposes 1-to-1 and integer constraints. However, most of previous approaches relax these constraints to acquire their solution, therefore, they produce erroneous solutions. For example, Spectral Matching (SM) [10] drops 1-to-1 matching constraints to get rank-1 approximation of the affinity matrix and Reweighted Random Walk Matching (RRWM) [1] drops integer constraints and finds solution on real number space.

To solve this challenging problem, in this chapter, we adopt data-driven Markov Chain Monte Carlo (DDMCMC) framework [52] which enables to avoid local minima efficiently. Unlike stochastic graph matching methods in the literature [53, 54], our work addresses a general graph matching problem and exploits domain knowledges from conventional graph matching approaches by data-driven proposals in a principled MCMC framework [55].

Recently, various works are published to address hypergraph matching problem [20, 35, 36, 2]. Hypergraphs can embed more sophisticated information on their hyperedges, therefore, produce more robust performances on severe noises. We extend our MCMC based graph matching algorithm to hypergraph matching problem. Energy function for MCMC framework is modified using hypergraph matching objective function. Again, we impose domain knowledges onto data-driven proposal distribution by taking solutions from Hyper-Graph Matching (HGM) [20] and Reweighted Random Walks Hypergraph Matching (RRWHM) [2].

In the experiment section, we demonstrate robustness of our DDMCMC algorithm by performing synthetic random graph and hypergraph matching problems [1]. For revealing effectiveness of our data-driven proposal scheme, we compare three different versions of our MCMC algorithm; pure MCMC algorithm and DDMCMC with two conventional approaches. In ordinary graph matching problems, SM [10] and RRWM [1] are compared and exploited into data-driven proposals. For solving hypergraph matching problems, HGM [20] and RRWHM [2] are used for comparison and data-driven proposals. In both experiments, we demonstrated that our MCMC algorithm itself outperforms standard (hyper)graph matching algorithms and our data-driven proposals enable our graph matching algorithm more robust to deformation and outlier noises.

3.2 Graph Matching Formulation

The graph matching problem is formulated by following several conventional publications including [10, 1, 2]. There are two graphs $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$ and $\mathcal{G}' = \{\mathcal{V}', \mathcal{E}', \mathbf{A}'\}$ to be matched, while $\mathcal{V}$ is set of nodes, $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ is set of edges, and $\mathbf{A}$ is attribute values. The attribute $\mathbf{A} \in \mathbb{R}^{n \times n}$ describes characteristics of nodes $v_i \in \mathcal{V}$ (by diagonal elements of $\mathbf{A}$) and edges $e_{i,j} \in \mathcal{E}$ (by off-diagonal elements of $\mathbf{A}$) where $n = |\mathcal{V}|$ is number of nodes in $\mathcal{G}$. To represent node correspondences between $\mathcal{G}$ and $\mathcal{G}'$, the assignment matrix $\mathbf{X} \in \{0,1\}^{n \times n'}$ is adopted where $\mathbf{X}_{i,a} = 1$ if node $v_i \in \mathcal{V}$ is matched to $v'_a \in \mathcal{V}'$ and $\mathbf{X}_{i,a} = 0$ if there is no match between two nodes. In general, the one-to-one matching constraints are imposed which enforce one node to be matched one node in maximal; $\mathbf{X}\mathbf{1}_{n' \times 1} \preceq \mathbf{1}_{n \times 1}$ and $\mathbf{X}^\intercal \mathbf{1}_{n \times 1} \preceq \mathbf{1}_{n' \times 1}$.

The goal of graph matching problem is to find the best mapping between elements of $\mathcal{V}$ and $\mathcal{V}'$ which best preserves attributes between $\mathbf{A}_{i,j}$ and $\mathbf{A'}_{a,b}$ when $\mathbf{X}_{i,a} = 1$ and $\mathbf{X}_{j,b} = 1$. To measure how similar $\mathbf{A}_{i,j}$ and $\mathbf{A'}_{a,b}$ are, similarity functions $f_v(\cdot,\cdot)$ and $f_e(\cdot,\cdot)$ are introduced where $f_v(\cdot,\cdot)$ and $f_e(\cdot,\cdot)$ measure node and edge attribute compatibilities, respectively. For a given assignment $\mathbf{X}$, the graph matching score $S(\mathbf{X})$ can be defined as follows:

$$\begin{aligned} S(\mathbf{X}) = \lambda \sum_{i,a} f_v(v_i, v'_a)\mathbf{X}_{i,a} \\ + \sum_{ia,jb} f_e(e_{i,j}, e'_{a,b})\mathbf{X}_{i,a}\mathbf{X}_{j,b}. \end{aligned} \tag{3.1}$$

Thus, the best mapping $\mathbf{X}^*$ can be obtained by maximizing the matching score: $\mathbf{X}^* = \operatorname{argmax}_{\mathbf{X}} S(\mathbf{X})$.

As in [16, 10, 56], the graph matching problem can be formulated as Integer Quadratic Programming (IQP) problem. For formulating the graph matching by IQP, the assignment vector $\mathbf{x} \in \{0,1\}^{nn' \times 1}$ is required which is column-wise concatenation of $\mathbf{X}$. Additionally, the affinity matrix $\mathbf{M}$ is also introduced which embeds node and edge similarity scores by the following rule:

$$\mathbf{M}_{ia,jb} = \begin{cases} f_v(v_i, v'_a) & \text{if } ia = jb \\ f_e(e_{i,j}, e'_{a,b}) & \text{otherwise} \end{cases} \tag{3.2}$$

With simple mathematics, the objective function $S(\mathbf{X})$ can be rewritten as follows;

$$S(\mathbf{X}) = S(\mathbf{x}) = \mathbf{x}^\intercal \mathbf{M} \mathbf{x}. \tag{3.3}$$

Then, the goal of the problem could be redefined as finding an assignment vector which maximizes the quadratic objective score function as follows;

$$\begin{aligned} \mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}}(\mathbf{x}^\intercal \mathbf{M}\mathbf{x}), \\ \mathbf{X}\mathbf{1}_{n'} \preceq \mathbf{1}_n,\ \mathbf{X}^\intercal \mathbf{1}_n \preceq \mathbf{1}_{n'}. \end{aligned} \tag{3.4}$$

3.3 Algorithm

The proposed graph matching algorithm is based on MCMC sampling technique, which can efficiently explore high dimensional solution space. To generate effective samples, data-driven state transition proposal is adopted. In the following subsections, state transition, energy function, and data-driven proposal distribution are explained.

3.3.1 State Transition

With one-to-one constraints, a binary assignment vector $\mathbf{x}$ can be equivalently interpreted as a permutation vector. In Fig. 3.1(a) and 3.1(b), for example, we can represent 9-bit current states $\mathbf{x}$ and $\mathbf{x}'$ as (a, b, c) and (a, c, b), respectively. Since our method employs the permutation vector as the state representation, it is natural to take the switch operation as the state transition move for the MCMC framework. Switch transition move ensures that an arbitrary state $\mathbf{x}$ can reach all possible states in graph matching problem and reach back to $\mathbf{x}$ from any other state $\mathbf{x}'$. These *reachability* and *irreducibility* are key properties which enable MCMC sampling to follow the target distribution in the next subsection.

3.3.2 Energy Formulation

Theoretically, MCMC technique samples states which are following given target distribution [55]. Key idea of the proposed MCMC graph matching algorithm is to select a sample state with the highest probability among those samples. Thus, To find the best correspondence in MCMC framework, it is necessary to define the

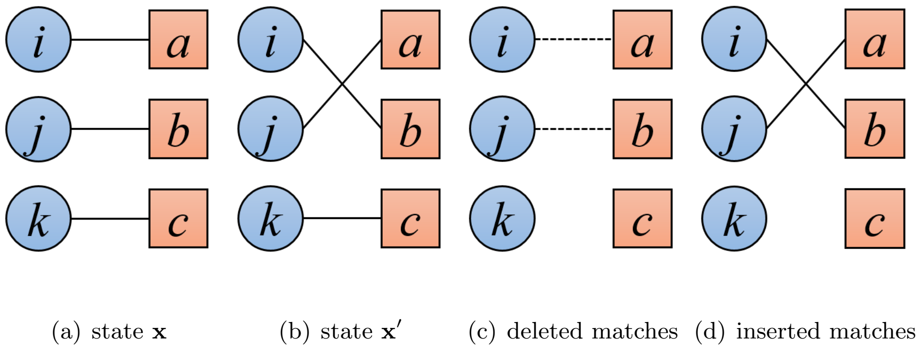


Figure 3.1: Illustration of state transition. (a)-(b) If we want to move from current state $\mathbf{x}$ to proposed state $\mathbf{x}'$, (c) there are two matches which will be deleted and (d) two matches will be inserted.

target distribution probability $\pi(\mathbf{x})$ or the energy $E(\mathbf{x})$ of the state $\mathbf{x}$:

$$\pi(\mathbf{x}) \propto \exp(\mathbf{x}^{\intercal}\mathbf{M}\mathbf{x}/T), \quad E(\mathbf{x}) = -\log[\pi(\mathbf{x})], \tag{3.5}$$

where T is the temperature for Simulated Annealing (SA) process [55]. T is scheduled to have start value T_s at first and decrease gradually (with the factor R) to the final value T_f with certain number of samples $N_{\max}$.

The MCMC framework also requires the acceptance ratio [55] $r(\mathbf{x} \to \mathbf{x}')$ which is defined as follows;

$$r(\mathbf{x} \to \mathbf{x}') = \min\left[\frac{\pi(\mathbf{x}')q(\mathbf{x}' \to \mathbf{x})}{\pi(\mathbf{x})q(\mathbf{x} \to \mathbf{x}')}, 1\right], \tag{3.6}$$

where $q(\mathbf{x}' \to \mathbf{x})$ is the proposal probability from state $\mathbf{x}'$ to state $\mathbf{x}$ and $q(\mathbf{x} \to \mathbf{x}')$ is from $\mathbf{x}$ to $\mathbf{x}'$ (defined in the following subsection). By accepting the proposed state $\mathbf{x}'$ as a new state with the probability of $r(\mathbf{x} \to \mathbf{x}')$, the Markov chain is theoretically guaranteed to follow its target distribution $\pi(\mathbf{x})$ [55].

3.3.3 Data-Driven Proposal

Since our objective in the graph matching problem is to find the best state with the highest score, it is efficient to design the proposal distribution $q(\mathbf{x} \to \mathbf{x}')$ prefers states with higher scores. To achieve this efficiency by incorporating domain knowledge in proposing new states of the Markov chain, we adopt the data-driven techniques to guide our Markov chain using achievements of previously existing graph matching approaches. For example, SM algorithm [10] is based on the fact that a match which has large corresponding element in the principal eigenvector has a great chance to be a true match. Our method also utilizes this fact in a probabilistic way. We use data-driven proposal as the state transition kernel. The state transition from $\mathbf{x}$ to $\mathbf{x}'$ enforces two matches to be deleted and two matches to be inserted (see Fig. 3.1(c) and 3.1(d)). The probability of state transition from $\mathbf{x}$ to $\mathbf{x}'$ is defined by

$$q(\mathbf{x} \to \mathbf{x}') \propto \mathbf{x}^{alg}_{ib} + \mathbf{x}^{alg}_{ib} - \mathbf{x}^{alg}_{ia} - \mathbf{x}^{alg}_{jb} + C, \tag{3.7}$$

where $\mathbf{x}^{alg}$ denotes rank-1 approximation of a deterministic algorithm alg and indices refer Fig. 3.1. The offset value C is required to prevent a computed value from becoming a negative value. The value of $q(\mathbf{x} \to \mathbf{x}')$ becomes greater when the proposed state contains matches with larger confidence score than the current state. This proposal is likely to include a state which has more chances to contain more true matches.

Our proposed method can overcome the local minima problem of deterministic methods (because of the relaxation) by exploring large space with suitable samples which are generated by the data-driven proposal. This fact enables our algorithm to find a solution with higher matching score. The proposed DDMCMC graph matching algorithm is summarized in Algorithm 4.

Algorithm 4: Data-Driven MCMC Graph Matching

```
input  : an initial state $\mathbf{x}$, parameters $\{T_s, T_f, R, N_{\max}\}$
output: an optimal state $\mathbf{x}^*$
$\mathbf{x}^* = \mathbf{x}$; $T = T_s$; $N = 0$;
while $T > T_f$ do
    Calculate the proposal distribution: $q(\mathbf{x} \to \mathbf{x}')$;
    Sample $\mathbf{x}'$ from $q(\mathbf{x} \to \mathbf{x}')$;
    Increase $N$ by 1;
    Calculate the acceptance ratio: $r(\mathbf{x} \to \mathbf{x}')$;
    if random() $< r(\mathbf{x} \to \mathbf{x}')$ then
        $\mathbf{x} = \mathbf{x}'$;
        if $\pi(\mathbf{x}) > \pi(\mathbf{x}^*)$ then
            $\mathbf{x}^* = \mathbf{x}$;
        end
    end
    if $N = N_{\max}$ then
        $T = T \times R$; $N = 0$;
    end
end
```

3.4 Hypergraph Extension

In this section, we propose extension of our novel MCMC based graph matching algorithm into hypergraph matching problem. In this chapter, we discuss only third-order hypergraph like [2].

3.4.1 Hypergraph Matching Problem

Two hypergraphs are defined as $G_h = \{\mathcal{V}, \mathcal{E}_h, \mathbf{T}\}$ and $G'_h = \{\mathcal{V}', \mathcal{E}'_h, \mathbf{T}'\}$, where $\mathcal{E}_h$, $\mathcal{E}'_h$ are sets of hyperedges and $\mathbf{T}$, $\mathbf{T}'$ are corresponding attribute tensors. As the same manner with the ordinary graph matching problem, the goal of hypergraph matching can be defined as the following:

$$S_h(\mathbf{X}) = \sum_{ia,jb,kc} f_h(\mathbf{T}_{ijk}, \mathbf{T}'_{abc})\mathbf{X}_{i,a}\mathbf{X}_{j,b}\mathbf{X}_{k,c}, \tag{3.8}$$

where user-defined similarity function $f_h(\mathbf{T}_{ijk}, \mathbf{T}'_{abc})$ measures the compatibility score between two hyperedges e^h_{ijk} and e'^h_{abc} (or two attributes $\mathbf{T}_{ijk}$ and $\mathbf{T}'_{abc}$). By using linear algebra and tensor-product [2], $S_h(\mathbf{X})$ can be represented as the following:

$$S_h(\mathbf{x}) = \mathbf{H} \times_1 \mathbf{x} \times_2 \mathbf{x} \times_3 \mathbf{x}, \tag{3.9}$$

where the affinity tensor $\mathbf{H}$ is defined by $\mathbf{H}_{ia,jb,kc} = f_h(\mathbf{T}_{ijk}, \mathbf{T}'_{abc})$. Naturally, the goal of hypergraph matching problem becomes;

$$\begin{aligned} \mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}} &(\mathbf{H} \times_1 \mathbf{x} \times_2 \mathbf{x} \times_3 \mathbf{x}), \\ &\mathbf{X}\mathbf{1}_{n'} \preceq \mathbf{1}_n,\ \mathbf{X}^\intercal\mathbf{1}_n \preceq \mathbf{1}_{n'}. \end{aligned} \tag{3.10}$$

3.4.2 Energy Formulation & Data-Driven Proposal

By defining $\pi_h(\mathbf{x})$ using hypergraph matching objective score function, the energy function for MCMC hypergraph matching is achieved.

$$\begin{aligned} \pi_h(\mathbf{x}) &\propto \exp(S_h(\mathbf{x})/T), \\ E_h(\mathbf{x}) &= -\log[\pi_h(\mathbf{x})]. \end{aligned} \tag{3.11}$$

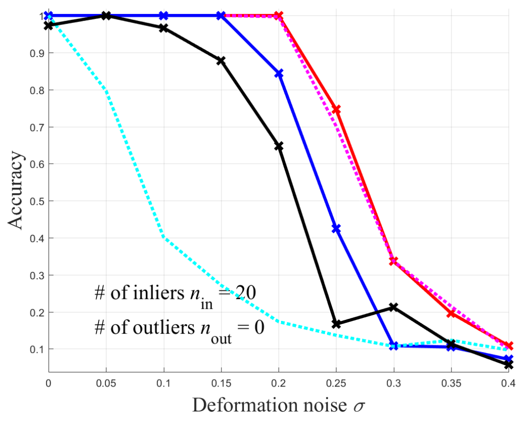
Hypergraph DDMCMC algorithm can be derived by simply replacing $\pi(\mathbf{x})$ with $\pi_h(\mathbf{x})$ in Algorithm 4.

At the same way in DDMCMC for ordinary graph matching, proposal distribution $q(\mathbf{x} \to \mathbf{x}')$ can be designed using conventional hypergraph matching approaches. In our experiments, we are adopting two hypergraph matching methods for demonstrating effectiveness of the proposed data-driven proposal. One is Hyper-Graph Matching (denoted by HGM+DDMCMC) [20] which interprets hypergraph matching problem in probabilistic way and marginalizes attribute tensor $\mathbf{H}$ to produce rank-1 approximation of the tensor. The other is Reweighted Random Walks for Hypergraph Matching (denoted by RRWHM+DDMCMC) [2] which defines random walks simulation on hypergraph for interpreting hypergraph matching problem and produces state-of-the-art performance.

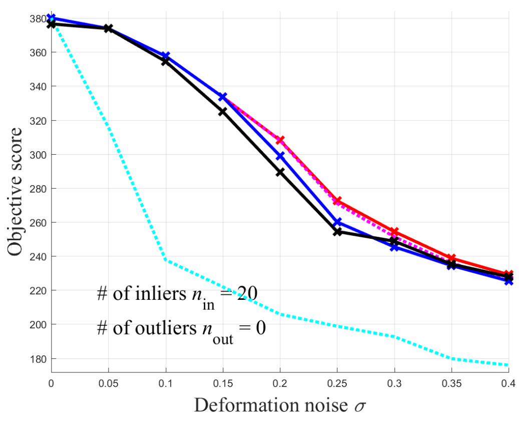
3.5 Experiment

3.5.1 Random Graph Matching Problem

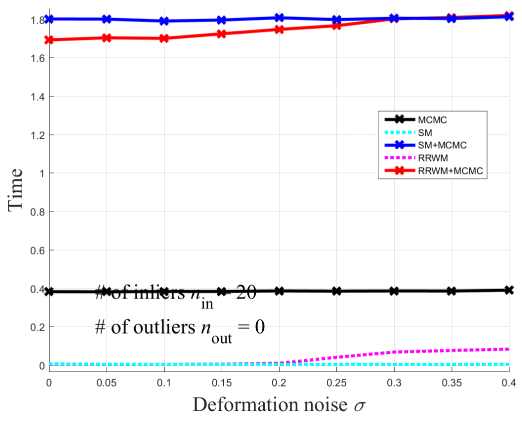
In this section, we compare our algorithm with SM [10] and RRWM [1], which are representative graph matching algorithms in matching problem. RRWM especially shows state-of-the-art performance on various noise conditions. We also inves-



(a) accuracy

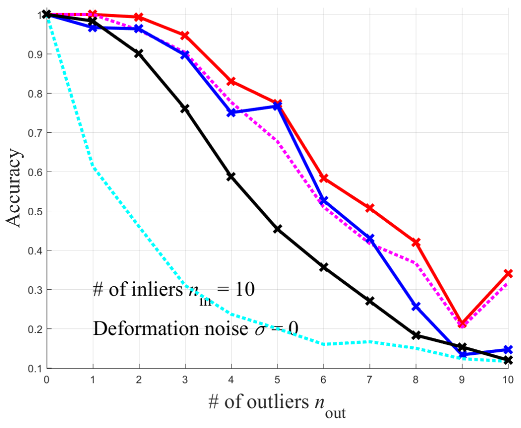


(b) objective Score

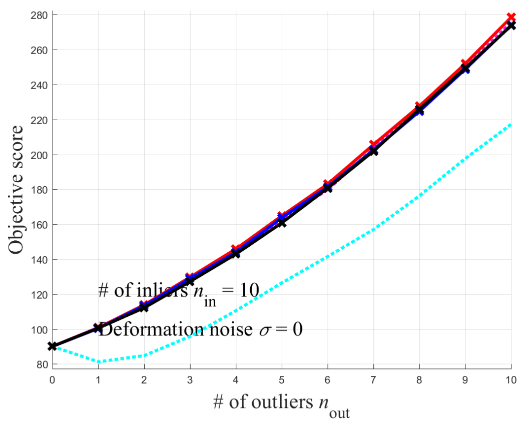


(c) execution Time

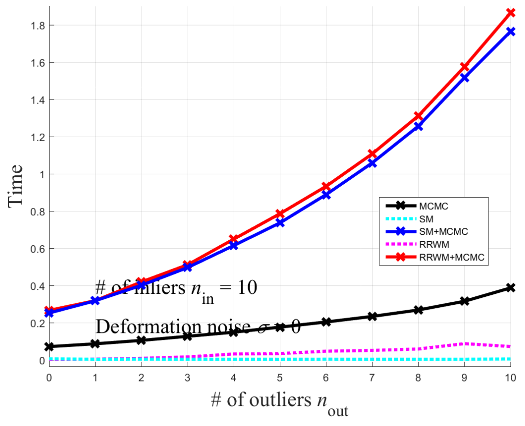
Figure 3.2: Performance on graph matching problem according to deformation noise changes. From the left to the right, accuracies, objective scores and execution times are reported. Note that proposed algorithms are with solid lines while conventional algorithms are with dashed lines.



(a) accuracy



(b) objective Score



(c) execution Time

Figure 3.3: Performance on graph matching problem according to outlier noise changes. From the left to the right, accuracies, objective scores and execution times are reported.

tigate the effect of data-driven proposal distribution by comparing MCMC results with DDMCMC results. Two DDMCMC algorithms are compared: one is combined with SM (denoted by SM+DDMCMC) and the other is with RRWM (denoted by RRWM+DDMCMC).

Quantitative evaluation of our algorithm is basically similar to the experiment of [1]. We generate two graphs with $n = n_{in} + n_{out}$ nodes where n_{in} and n_{out} are number of inliers and outliers, respectively. In one graph, a graph $\mathcal{G}$ with n nodes are generated and attribute $\mathbf{A}$ is randomly assigned by $\mathbf{A} \sim \mathcal{U}(0,1)^{n\times n}$. The other graph $\mathcal{G}'$ to be matched is generated by copying attribute $\mathbf{A}$ and adding noises: $\mathbf{A}' = \mathbf{A} + \sigma\mathbf{N}$, where Gaussian noise $\mathbf{N} \sim \mathcal{N}(0,1)^{n\times n}$ and σ controls amount of deformation noises. By overwriting n_{out} node-related elements on $\mathbf{A}'$ with uniform random numbers, we can simulated both deformation and outlier noises on $\mathcal{G}'$.

To construct affinity matrix M, we have to measure how similar two matches are for every possible pairs. The following measure is used to construct $\mathbf{M}$ as in [1]:

$$\mathbf{M}_{ia,jb} = \exp(-\|\mathbf{A}_{ij} - \mathbf{A}'_{ab}\|^2/\sigma_d), \tag{3.12}$$

where σ_d is empirically set to 0.15 for the best performance.

All experimental results show the average score of 30 trials for each setting. To fairly compare the effect of proposal distribution, parameters of MCMC and DDMCMC algorithms are fixed but proposal distribution. The random proposal is used for MCMC algorithm. In this experiment, T_s and T_f are set to 10 and 0.1, respectively. The decreasing factor R is set to 0.95. We try to adjust the number of samples per temperature, thus we assign $N_{\max} = n^2$. This allows the method to have more samples when the problem size is larger. Performance of the algorithms is evaluated with the accuracy of matches and with the matching score $S(\mathbf{x}^*)$. The accuracy can be obtained with the ratio of correct matches over n inliers.

Quantitative results are shown in Fig. 3.2 and 3.3. The figures in the left column show the average accuracy of matches, while the figures in the middle column show the average matching score. Our three methods almost always show better performance than its base algorithm, *i.e.*, SM+DDMCMC shows better performance than SM and RRWM+DDMCMC outperforms RRWM in outlier experiment. Our DDMCMC algorithm also show best performance in the sense of objective function maximizer (or energy minimizer). We can also verify that our data-driven proposal works as a good proposal since DDMCMC show better performance than MCMC with the same number of samples.

3.5.2 Random Hypergraph Matching Problem

We also perform experiments on hypergraph matching problem. As a similar manner with ordinary graphs, hypergraph $\mathcal{G}_h$ is generated with n nodes and attributed tensor $\mathbf{T} \sim \mathcal{U}(0,1)^{(n\times n\times n)}$. Then the second hypergraph $\mathcal{G}'_h$ generated by following the same procedure of previous subsection.

Next, we calculate the affinity tensor $\mathbf{H}$ by comparing two corresponding attributes;

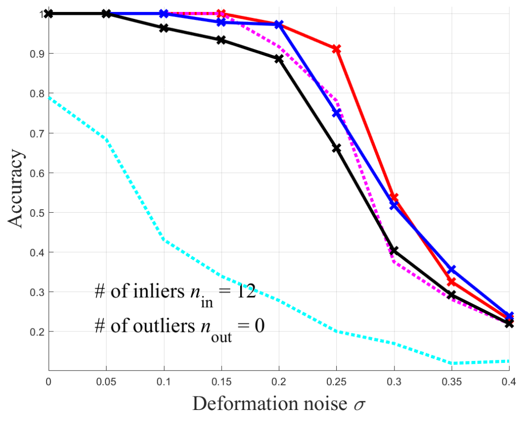
$$\mathbf{H}_{ia,jb,kc} = \exp(-\|\mathbf{T}_{ijk} - \mathbf{T}'_{abc}\|^2/\sigma_d). \tag{3.13}$$

Again, the sensitivity parameter is set to 0.15 as in the ordinary graph matching experiment.

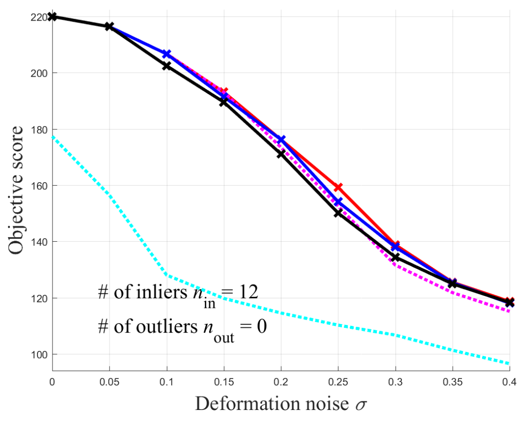
We compare our three MCMC algorithms with two conventional hypergraph matching algorithms; HGM [20] and RRWHM [2]. Experimental results on hypergraph matching problem is reported in Fig. 3.4 and 3.5. Throughout both deformation and outlier test, DDMCMC algorithms show better performance than their base algorithm. Again, data-driven proposals are proven to be efficient since DDMCMC always outperforms MCMC algorithm.

3.6 Conclusion

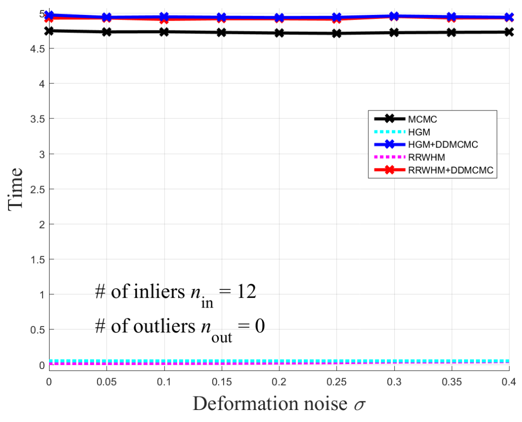
We proposed a stochastic graph matching algorithm which can avoid local minima problem of deterministic approaches. It adopts MCMC framework with the data-driven state transition proposals, which efficiently explores the solution space of graph matching. Experiments show that our graph matching algorithm is robust to deformation and outliers arising from the practical correspondence problems, and outperforms the state-of-the-art graph matching algorithms.



(a) accuracy

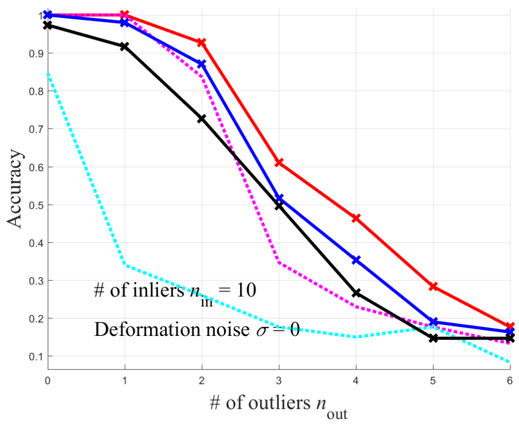


(b) objective Score

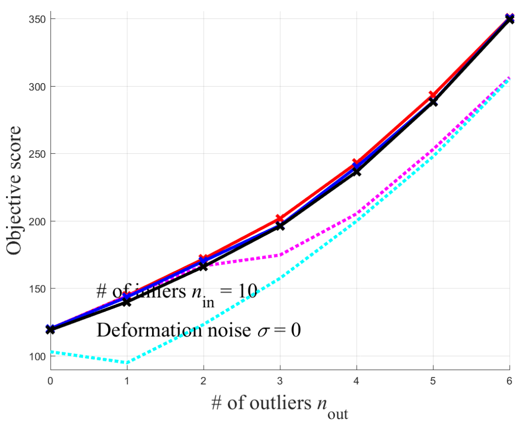


(c) execution Time

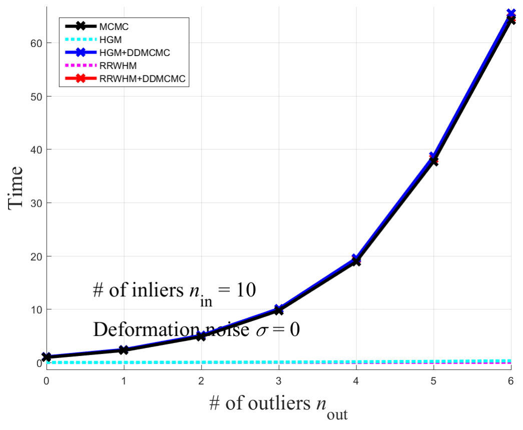
Figure 3.4: Performance on hypergraph matching problem according to deformation noise changes. From the left to the right, accuracies, objective scores and execution times are reported.



(a) accuracy



(b) objective Score



(c) execution Time

Figure 3.5: Performance on hypergraph matching problem according to outlier noise changes. From the left to the right, accuracies, objective scores and execution times are reported.

Chapter 4

Graph and Hypergraph Matching Revisited

4.1 Introduction

Graph matching is one of most widely used tools in computer vision, machine learning, and pattern recognition [3] research area. In particular, it has been used for establishing correspondences between two sets of features [1, 2] because a graphical model can naturally encode features and their relations into node and edge attributes. Unlike popular matching techniques for rigid motion such as RANSAC [11] and the iterative closest points (ICP) [12], graph matching effectively handles non-rigid deformation with appearance changes, and finds reliable matches by minimizing distortion between corresponding features and their relations. This strong advantage enables graph matching to handle challenging correspondence problems in real-world images.

In general, the problem of graph matching belongs to the quadratic assignment problem (QAP), and a myriad of graph matching algorithms have been proposed with different types of formulations and characteristics in the literature [3, 28, 29]. In this chapter, we classify the most popular families of recent graph matching formulations into two types of formulations: *affinity-based* [18, 19, 10, 16, 20, 21, 1, 24, 30] and *adjacency-based* [31, 32, 33, 34, 23, 25] formulations. The affinity-based formulation takes a given *affinity* score for each candidate correspondence between two graphs as its input, and adopts any predefined scores for node and edge correspondences. The adjacency-based formulation only takes an attribute *adjacency* value for nodes and edges on each graph. A *affinity* score for a node correspondence or an edge correspondence between two graphs is computed on-the-fly using those attributes in the formulation. In this sense, the affinity-based formulation is more general than the adjacency-based one. At the cost of losing its generality, however, the adjacency-based formulation obtains a substantial advantage in optimization efficiency. While these two types of formulations have existed together for a few decades and cover most of recent state-of-the-art graph matching methods, a comparative analysis between them has been rarely done in the literature.

We revisit these two families of graph matching formulations, analyze their relations, and propose efficient graph matching strategies. The main contributions of this chapter can be summarized as follows: (i) we analyze two representative formulations in graph matching, *i.e.*, adjacency-based and affinity-based representations, and show their transformational relations into equivalent counterparts. (ii) we study the characteristics of those formulations using recent state-of-the-art algorithms and popular similarity measures (iii) we propose the efficient way to solve large-size graph matching problem by modifying previously introduced matching techniques.

There are several researches which focus on hypergraph matching problem [20, 35, 36, 2, 37]. Hyperedges are simultaneously connecting more than two nodes on hypergraphs, thus, more sophisticated and robust attributes can be embedded into attribute tensors. We introduce two types of hypergraph matching formulations and verify relationships between them. To our best knowledge, this is the first work which related *affinity* formulation and *adjacency* formulation on hypergraph matching problem. Furthermore, we derive transformation rule of two formulations and reinterpret conventional hypergraph matching algorithms for the other type of formulation. Finally, performances of conventional approaches in both formulation are thoroughly compared in experiment section.

4.2 Related Works

One of the most popular form is affinity-based formulation [18, 19, 10, 16, 20, 21, 1, 24, 30]. The affinity-based formulation can be represented by maximization on an affinity matrix under the matching constraints. Another popular approach is adjacency-based formulation which solves the graph matching by rearranging node order for minimizing corresponding graph attribute differences [31, 32, 33, 34, 23, 25]. Some researchers are solving the graph matching problem by adopting energy minimization techniques [53, 17, 22]. Besides these conventional approaches, there are interesting works using the pooling strategy [26], game theory [57], bottom-up clustering technique [8] or higher order graph representations [2, 35, 36].

Spectral Matching (SM) [10] interprets QAP using the affinity matrix. SM introduces simple and effective spectral approaches. Matching constraints are not considered during the optimization step. Balanced Graph Matching (BGM) [16] proposed modified objective function. BGM puts the matching constraints into the original objective function thus their optimization produces more constraint satisfying solution. Integer Projected Fixed Point method (IPFP) [21] is the representative work which solve QAP while constraining the solution space onto the matching constraints. Also IPFP shows outstanding performances with faster convergence speed. Reweighted Random Walk Matching (RRWM) [1] introduces novel interpretation of QAP. RRWM solves QAP by node-ranking problem on the association graph. Also they proposed effective reweighting scheme that enables the matching constraints which are well imposed during the optimization. Factorized Graph Matching (FGM) [23, 25] shows powerful performances by introducing convex and concave relaxation of the original objective function and optimizing them using well-known Concave-Convex Procedure (CCCP) and modified Frank-Wolfe (MFW) techniques. Besides, FGM also introduces the way to construct adjacency matrices which lead to the same result with the affinity matrix.

Two conventional graph matching formulations are introduced and their transformational relation is discussed in Section 4.3. Then, two affinity measures and their combination with the formulations are explained in Section 4.4. We introduce reinterpretation of famous graph matching methods for the two formulations in Section 4.5. Finally, performances and characteristics according to formulations, affinity measures, and algorithms are shown in Section 4.7.

4.3 Two Types of Formulations

We first formulate ordinary graph matching and then expand it to hypergraph case. Since most of hypergraph matching approaches solve third order hypergraph, we also assume third order hypergraphs. Previous graph matching algorithms can be divided into two streams according to the problem formulation: adjacency-based (ADJ) formulation [23] and affinity-based (AFF) formulation [19, 10, 21, 1]. We explain each of the approaches and address the relationship between them in the following sections.

Basic graph matching notations: The goal of graph matching is to find the best correspondence between two graphs, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$ and $\mathcal{G}' = \{\mathcal{V}', \mathcal{E}', \mathbf{A}'\}$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges. We denote the node and edge attributes with $\mathbf{A}_{i,i} \in \mathbb{R}_+^n$ and $\mathbf{A}_{i,j} \in \mathbb{R}_+^{n\times n}$ respectively, where $n = |\mathcal{V}|$ is the number of nodes, $\mathbf{A}_{i,i}$ is the attribute of i-th node and $\mathbf{A}_{i,j}$ is the attribute of directed edge $(i,j) \in \mathcal{E}$.

Extension to hypergraph matching: Instead of using attributes $\mathbf{a}$ and $\mathbf{A}$, we represent hyper-edge attributes by employing tensor $\mathbf{T} \in \mathbb{R}_+^{n\times n\times n}$. Then we have two hypergraphs to be matched; $\mathcal{G}_h = \{\mathcal{V}, \mathcal{E}_h, \mathbf{T}\}$ and $\mathcal{G}'_h = \{\mathcal{V}', \mathcal{E}'_h, \mathbf{T}'\}$.

Assumptions: For simple description, we assume that all attributes are none-negative and symmetric (super-symmetric for high order attributes). Additionally, we assume third order hypergraphs.

4.3.1 Adjacency-based Formulation

ADJ formulation for ordinary graph matching: Without loss of generality, we assume $n \le n'$. Adjacency-based formulation finds the optimal mapping $\mathbf{X}^*$, which minimizes the difference between corresponding node and edge attributes

$$\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmin}} \left|\left|\mathbf{A} - \mathbf{X}\mathbf{A}'\mathbf{X}^\intercal\right|\right|_F^2, \tag{4.1}$$

where $\mathbf{X} \in \Pi = \{\mathbf{X} | \mathbf{X} \in \{0,1\}^{n\times n'}, \mathbf{X}^\intercal \mathbf{1}_n \le \mathbf{1}_{n'}, \mathbf{X}\mathbf{1}_{n'} = \mathbf{1}_n\}$ is constrained to be a one-to-one mapping. Simple linear algebra (proof in supplementary material) shows that the optimization problem in Eq.(4.1) is equivalent to the following maximization problem:

$$\mathbf{X}^* = \operatorname*{argmax}_{\mathbf{X}} \operatorname{Tr}\left(\mathbf{A}(\mathbf{X}\mathbf{A}'\mathbf{X}^\intercal)^\intercal\right) \tag{4.2}$$

Proof. The following Eq.(4.3) proves how Eq.(4.1) and Eq.(4.2) are equivalent. Thus, the original minimization problem in Eq.(4.1) can be solved by maximizing Eq.(4.2). Note that $\mathbf{X} \in \Pi$, where Π is the set of assignment matrices so that $\mathbf{X}\mathbf{X}^\intercal = \mathbf{I}_{n\times n}$ and $\mathbf{X}^\intercal\mathbf{X} = \begin{pmatrix} \mathbf{I}_{n\times n} & \mathbf{0}_{n\times(n'-n)} \\ \mathbf{0}_{(n'-n)\times n} & \mathbf{0}_{(n'-n)\times(n'-n)} \end{pmatrix}$.

$$\begin{aligned}
\mathbf{X}^* &= \operatorname*{argmin}_{\mathbf{X}} \|\mathbf{A} - \mathbf{X}\mathbf{A}'\mathbf{X}\|_F^2 \\
&= \operatorname*{argmin}_{\mathbf{X}} \operatorname{Tr}\{(\mathbf{A} - \mathbf{X}\mathbf{A}'\mathbf{X})^\intercal(\mathbf{A} - \mathbf{X}\mathbf{A}'\mathbf{X})\} \\
&= \operatorname*{argmin}_{\mathbf{X}} \operatorname{Tr}(\mathbf{A}^\intercal\mathbf{A} - \mathbf{A}^\intercal\mathbf{X}\mathbf{A}'\mathbf{X}^\intercal - \mathbf{X}\mathbf{A}'^\intercal\mathbf{X}^\intercal\mathbf{A} \\
&\qquad\qquad + \mathbf{X}\mathbf{A}'^\intercal\mathbf{X}^\intercal\mathbf{X}\mathbf{A}'\mathbf{X}^\intercal) \\
&= \operatorname*{argmin}_{\mathbf{X}} \operatorname{Tr}(\mathbf{A}^\intercal\mathbf{A}) - 2\operatorname{Tr}(\mathbf{A}^\intercal\mathbf{X}\mathbf{A}'\mathbf{X}^\intercal) \\
&\qquad\qquad + \operatorname{Tr}(\mathbf{X}\mathbf{A}'^\intercal\mathbf{A}'\mathbf{X}^\intercal) \\
&= \operatorname*{argmin}_{\mathbf{X}} \operatorname{Tr}(\mathbf{A}^\intercal\mathbf{A} + \mathbf{A}'^\intercal\mathbf{A}') - 2\operatorname{Tr}(\mathbf{A}^\intercal\mathbf{X}\mathbf{A}'\mathbf{X}^\intercal) \\
&= \operatorname*{argmax}_{\mathbf{X}} \operatorname{Tr}(\mathbf{A}\mathbf{X}\mathbf{A}'\mathbf{X}^\intercal) \\
&= \operatorname*{argmax}_{\mathbf{X}} \operatorname{Tr}(\mathbf{A}(\mathbf{X}\mathbf{A}'\mathbf{X}^\intercal)^\intercal).
\end{aligned} \tag{4.3}$$

□

It reorders the components of attributes according to the mapping $\mathbf{X}$ and measures the similarity between two attributes as a dot product. We further extend the objective function in Eq.(4.2) to deal with multiple adjacency matrices as follows:

$$\mathbf{X}^* = \operatorname*{argmax}_{\mathbf{X}} \sum_{c=1}^{C} \operatorname{Tr}\left(\mathbf{A}^c(\mathbf{X}\mathbf{A}'^c\mathbf{X}^\intercal)^\intercal\right). \tag{4.4}$$

It enables informative C-dimensional node and edge attributes, respectively. Note that we denote the attribute of edge $(i,j) \in \mathcal{E}$ with $\mathbf{A}_{i,j} \in \mathbb{R}^{C\times 1}$, where its c-th component is $\mathbf{A}^c_{i,j}$.

ADJ formulation for hypergraph matching: Eq.(4.1) can be extended for hypergraph matching with ADJ form by minimizing permuted tensor attribute differences;

$$\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmin}} \sum_k \left|\left|\mathbf{T'}_{ijk} - \mathbf{T}_{\pi(i)\pi(j)\pi(k)}\right|\right|^2_F, \tag{4.5}$$

where the permutation function $\pi(i)$ can be derived from the assignment matrix $\mathbf{X}$ by using the relation $\mathbf{X}_{i,\pi(i)} = 1$. Eq.(4.5) can be re-written by using n-mode tensor multiplication [44].

$$\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmin}} \left|\left|\mathbf{T'} - \mathbf{T} \times_1 \mathbf{X} \times_2 \mathbf{X} \times_3 \mathbf{X}\right|\right|^2_F. \tag{4.6}$$

$$\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmax}} \left|\left|(\mathbf{T} \times_1 \mathbf{X} \times_2 \mathbf{X} \times_3 \mathbf{X}) \odot \mathbf{T'}\right|\right|_F. \tag{4.7}$$

4.3.2 Affinity-based Formulation

Affinity-based formulation maximizes sum of affinity scores between corresponding graph attributes. It casts the graph matching problem into the following optimization problem:

$$\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmax}} \sum_{\substack{(i,j)\in\mathcal{E} \\ (a,b)\in\mathcal{E}'}} \mathbf{X}_{i,a}\mathbf{X}_{j,b} f(\mathbf{A}_{i,j}, \mathbf{A'}_{a,b}) \tag{4.8}$$

where the affinity function $f(\cdot,\cdot)$ measures similarity between two attributes.

For node indices, $i, j \in \mathcal{V}$, we slightly abuse the notation ia to denote $(i+(a-1)n)$-th index, which corresponds to the (i,a)-component of $\mathbf{X}$. Since it is assumed that

$f(\cdot,\cdot)$ is an arbitrary affinity function, affinity-based formulation can solve more general problems than the formulation in Section 4.3.1. We can encode affinity score using the affinity matrix:

$$\mathbf{M}_{ia,jb} = f(\mathbf{A}_{i,j}, \mathbf{A'}_{a,b}) \tag{4.9}$$

Then, Eq.(4.8) is equivalent to the following optimization problem:

$$\begin{aligned} \mathbf{x}^* &= \operatorname*{argmax}_{\mathbf{x}} (\mathbf{x}^\intercal \mathbf{M} \mathbf{x}) \\ s.t.\ \ \mathbf{x} &= \text{vec}(\mathbf{X}), \mathbf{X} \in \Pi \end{aligned} \tag{4.10}$$

AFF formulation for hypergraph matching: In previous hypergraph matching approaches ([20, 35, 36, 2]), the affinity tensor is defined for embedding higher-order similarities between tensor attributes. Eq.(4.10) can be extended as follows;

$$\mathbf{H}_{ia,jb,kc} = f_h(\mathbf{T}_{ijk}, \mathbf{T'}_{abc}) \tag{4.11}$$

$$\begin{aligned} \mathbf{x}^* &= \operatorname*{argmax}_{\mathbf{x}} (\mathbf{H} \times_1 \mathbf{x} \times_2 \mathbf{x} \times_3 \mathbf{x}) \\ s.t.\ \ \mathbf{x} &= \text{vec}(\mathbf{X}), \mathbf{X} \in \Pi \end{aligned} \tag{4.12}$$

Note that $\times_i$ represents tensor multiplication [44] between tensor and vector along with i-th dimension and d represents tensor dimension.

4.3.3 Relation between Two Formulations

We can always convert the affinity-based formulation into an equivalent adjacency-based formulation and vice versa. Zhou *et al.* [23, 25] showed the equivalence by decomposing the affinity matrix $\mathbf{M}$ into two parts related to graph structure and affinity values.

From affinity-based to adjacency-based formulation: For a given graph matching problem in Eq.(4.10), we can always find at least one set of adjacency matrices, $\{\tilde{\mathbf{A}}^c\}_{c=1}^C$ and $\{\tilde{\mathbf{A}}'^c\}_{c=1}^C$, which makes the original problem equivalent to Eq.(4.4). Let us consider a matrix $\tilde{\mathbf{M}} \in \mathbb{R}^{n^2 \times n'^2}$, which is a rearranged version of $\mathbf{M}$ that satisfies $\tilde{\mathbf{M}}_{ij,ab} = \mathbf{M}_{ia,jb}$. Then the objective function in Eq.(4.10) becomes

$$\mathbf{x}^\intercal \mathbf{M} \mathbf{x} = \sum_{ia,jb} \mathbf{x}_{ia} \mathbf{M}_{ia,jb} \mathbf{x}_{jb} = \sum_{ia,jb} \mathbf{x}_{ia} \tilde{\mathbf{M}}_{ij,ab} \mathbf{x}_{jb}. \tag{4.13}$$

According to the singular value decomposition, we can find at least one set of $\mathbf{u}^c \in \mathbb{R}^{n^2 \times 1}$ and $\mathbf{v}^c \in \mathbb{R}^{n'^2 \times 1}$ that satisfies $\tilde{\mathbf{M}} = \sum_{c=1}^C \mathbf{u}^c \cdot \mathbf{v}^{c\intercal}$. Since $\tilde{\mathbf{M}}_{ij,ab} = \sum_{c=1}^C \mathbf{u}_{ij}^c \mathbf{v}_{ab}^c$, if we set $\tilde{\mathbf{A}}_{i,j}^c = \mathbf{u}_{ij}^c$ and $\tilde{\mathbf{A}}'^c_{a,b} = \mathbf{v}_{ab}^c$, the Eq.(4.13) is equal to the following:

$$\begin{aligned} \sum_{c=1}^C \sum_{ia,jb} \mathbf{x}_{ia} \mathbf{u}_{ij}^c \mathbf{v}_{ab}^c \mathbf{x}_{jb} &= \sum_{c=1}^C \sum_{i,a,j,b} \mathbf{X}_{i,a} \tilde{\mathbf{A}}_{i,j}^c \tilde{\mathbf{A}}'^c_{a,b} \mathbf{X}_{j,b} \\ &= \sum_{c=1}^C \operatorname{Tr}\left(\tilde{\mathbf{A}}^c \cdot (\mathbf{X}^\intercal \tilde{\mathbf{A}}'^c \mathbf{X})^\intercal \right). \end{aligned} \tag{4.14}$$

From adjacency-based to affinity-based formulation: For a given graph matching problem (4.2), if we assume the affinity function as

$$f(\mathbf{A}_{i,j}, \mathbf{A}'_{a,b}) = \sum_{c=1}^C \mathbf{A}_{i,j}^c \cdot \mathbf{A}'^c_{a,b}, \tag{4.15}$$

then it can be easily shown that Eq.(4.10) is equivalent to Eq.(4.4).

From affinity-based to adjacency-based formulation of the hypergraph matching: Define a matrix $\tilde{\mathbf{H}} \in \mathbb{R}^{n^3 \times n'^3}$ by re-arranging indices of the tensor $\mathbf{H} \in \mathbb{R}^{nn' \times nn' \times nn'}$. $\tilde{\mathbf{H}}_{ijk,abc} = \mathbf{H}_{ia,jb,kc}$.

$$\begin{aligned}\mathbf{H} \times_1 \mathbf{x} \times_2 \mathbf{x} \times_3 \mathbf{x} &= \sum_{ia,jb,kc} \mathbf{H}_{ia,jb,kc}\mathbf{x}_{ia}\mathbf{x}_{jb}\mathbf{x}_{kc} \\ &= \sum_{ia,jb,kc} \tilde{\mathbf{H}}_{ijk,abc}\mathbf{x}_{ia}\mathbf{x}_{jb}\mathbf{x}_{kc}.\end{aligned} \tag{4.16}$$

Again, we are able to find at least one set of $\mathbf{u}^s \in \mathbb{R}^{n^3\times 1}$ and $\mathbf{v}^s \in \mathbb{R}^{n'^3\times 1}$ that satisfies $\tilde{\mathbf{H}} = \sum_{s=1}^{S} \mathbf{u}^s \cdot \mathbf{v}^{s\intercal}$, where S is the rank of matrix $\tilde{\mathbf{H}} \in \mathbb{R}^{n^3\times n'^3}$. Construct two tensors $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{T}}'$ by $\tilde{\mathbf{T}}_{ijk} = \mathbf{u}_{ijk}$ and $\tilde{\mathbf{T}}'_{abc} = \mathbf{u}_{abc}$. Then Eq.(4.16) becomes as the following:

$$\begin{aligned}&\sum_{s=1}^{S}\sum_{ijk,abc} \mathbf{u}^s_{ijk}\mathbf{v}^s_{abc}\mathbf{x}_{ia}\mathbf{x}_{jb}\mathbf{x}_{kc} \\ =&\sum_{s=1}^{S}\sum_{ijk,abc} \tilde{\mathbf{T}}^s_{ijk}\mathbf{X}_{ia}\mathbf{X}_{jb}\mathbf{X}_{kc}\tilde{\mathbf{T}'}^s_{abc} \\ =&\sum_{s=1}^{S} \left|\left|(\mathbf{T}^s \times_1 \mathbf{X} \times_2 \mathbf{X} \times_3 \mathbf{X}) \odot \mathbf{T}'^s\right|\right|_F\end{aligned} \tag{4.17}$$

From adjacency-based to affinity-based formulation of the hypergraph matching: ADJ formulation can be easily transformed in AFF formulation by extending of Eq.(4.15) as follows;

$$f_h(\mathbf{T}_{ijk}, \mathbf{T}'_{abc}) = \mathbf{T}_{ijk}{}^\intercal\mathbf{T}'_{abc} = \sum_{s=1}^{S} \mathbf{T}^s_{ijk} \cdot \mathbf{T}'^s_{abc}. \tag{4.18}$$

4.4 Affinity Measures

In Section 4.3, we discussed two ADJ and AFF formulations to define graph matching problem. There are two widely used measures to obtain affinity values; one is dot-product (DOT) measure and the other is negative exponential (EXP) measure.

Those formulations and affinity measures generate four combinations and their characteristics are in the following:

ADJ formulation with DOT measure: Adjacency-based formulation aims at minimizing Frobenius norm of corresponding attribute differences as described in section 4.3.1. One can directly set attributes as adjacency matrices. A great advantage using dot-product formulation lies in its computational complexity.

AFF formulation with DOT measure: To encode DOT affinity scores into an affinity matrix, we can set the affinity function as $\mathbf{M}_{ia,jb} = f(\mathbf{A}_{i,j}, \mathbf{A'}_{a,b}) = \mathbf{A}_{i,j}{}^{\intercal}\mathbf{A'}_{a,b}$.

ADJ formulation with EXP measure: EXP measure can be encoded into adjacency matrices by following the technique of Zhou *et al.* [23, 25]. They construct $\mathbf{K}_p$ and $\mathbf{K}_q$ which records pairwise affinity values of node and edge attributes from two graphs respectively. Applying SVD to $\mathbf{K}_p$, $\mathbf{K}_q$ and graph structures, adjacency matrices can be derived (refer [23, 25] for detail).

AFF formulation with EXP measure: To encode EXP affinity scores into an affinity matrix, we can set the affinity function as $\mathbf{M}_{ia,jb} = f(\mathbf{A}_{i,j}, \mathbf{A'}_{a,b}) = \exp(-\|\mathbf{A}_{i,j} - \mathbf{A'}_{a,b}\|/\sigma_s)$. This approach requires additional scale parameter σ_s to be tuned for the best performance.

Discussion: We show that two different objectives yield the same solution under the same affinity measure. Main difference between the two formulations is their computational complexities since adjacency-based formulation requires $\mathcal{O}(n^2)$ memory space while affinity-based formulation requires $\mathcal{O}(n^4)$. In affinity measure aspect, however, main difference is freedom of affinity function one can use. Basically, adjacency-based formulation only covers DOT measure thus DOT measure shows its strength when there is enough attribute dimension. Otherwise EXP measure becomes more robust choice for the graph matching problem.

4.5 Existing Methods & Re-interpretations

Conventional affinity-based methods utilize the affinity matrix $\mathbf{M}$ in two ways; One is for calculating objective score $\mathbf{x}^\intercal\mathbf{M}\mathbf{x}$ and the other way is for gradient direction of objective score $\mathbf{M}\mathbf{x}$. Under the equivalence explained in Section 4.3.3, we can relate Eq.(4.4) and (4.10);

$$\mathbf{x}^\intercal\mathbf{M}\mathbf{x} \leftrightarrow \sum_c \mathrm{Tr}\left(\mathbf{A}^c(\mathbf{X}\mathbf{A}'^c\mathbf{X}^\intercal)^\intercal\right). \tag{4.19}$$

By differentiating Eq.(4.19) with respect to assignment vector $\mathbf{x}$ and matrix $\mathbf{X}$, we can derive counterpart of $\mathbf{M}\mathbf{x}$ in adjacency-based formulation. Using the property of matrix calculus [58] $\frac{\partial}{\partial\mathbf{X}}\mathrm{Tr}\left(\mathbf{A}\mathbf{X}\mathbf{B}\mathbf{X}^\intercal\right) = \mathbf{A}^\intercal\mathbf{X}\mathbf{B}^\intercal + \mathbf{A}\mathbf{X}\mathbf{B}$, we can derive the following;

$$\mathbf{M}\mathbf{x} \leftrightarrow \sum_c \left[(\mathbf{A}^c)^\intercal\mathbf{X}\mathbf{A}'^c + \mathbf{A}^c\mathbf{X}(\mathbf{A}'^c)^\intercal\right] \tag{4.20}$$

In the following subsections, existing representative graph matching techniques and their counterpart interpretation are discussed.

4.5.1 Spectral Matching

Spectral Matching (SM) of [10] is one of the representative affinity-based method. SM solves graph matching problem by simple power iteration and greedy discretization step. The original affinity-based SM is explained in Algorithm 6 and reinterpretation of adjacency-based SM is described in Algorithm 5 by using Eq. (4.20).

Algorithm 5: Spectral Matching with Adjacency Matrices

```
Given $\mathbf{A}$, $\mathbf{A}'$, and an initial $\mathbf{X}_0$
repeat
    $\mathbf{X}_{t+1} = \sum_c [(\mathbf{A}^c)^\intercal \mathbf{X}_t \mathbf{A}'^c + \mathbf{A}^c \mathbf{X}_t (\mathbf{A}'^c)^\intercal]$ $\mathbf{X}_{t+1} = \mathbf{X}_{t+1} / \|\mathbf{X}_{t+1}\|$
until $\mathbf{X}$ converges;
```

Algorithm 6: Spectral Matching with Affinity Matrix

```
Given $\mathbf{M}$ and an initial $\mathbf{x}_0$
repeat
    $\mathbf{x}_{t+1} = \mathbf{M}\mathbf{x}_t$, $\mathbf{x}_{t+1} = \mathbf{x}_{t+1} / \|\mathbf{x}_{t+1}\|$
until $\mathbf{x}$ converges;
```

4.5.2 Integer Projected Fixed Point

Leordeanu and Hebert proposed Integer Projected Fixed Point (IPFP) algorithm [21]. They fixed the solution space as Π defined in Section 4.3.1. IPFP iteratively pushes the current solution by considering both gradient direction and objective score gain. Note that $P_d(\cdot)$ in Algorithm 7 and 8 is a projection function which enforces input matrix/vector to satisfy the matching constraints. Especially, IPFP requires computation of $\mathbf{x}^\intercal \mathbf{M} \mathbf{y}$. We can derive the counterpart of $\mathbf{x}^\intercal \mathbf{M} \mathbf{y}$ for adjacency-based formulation by using the observation that $\mathbf{M}_{ia,jb} = \sum_c \mathbf{A}^c_{i,j} \cdot \mathbf{A}'^c_{a,b}$.

$$\begin{aligned}\mathbf{x}^\intercal \mathbf{M} \mathbf{y} &= \sum_{ia,jb} \mathbf{x}_{ia} \mathbf{M}_{ia,jb} \mathbf{y}_{jb} = \sum_c \sum_{ia,jb} \mathbf{X}_{i,a} \mathbf{A}^c_{i,j} \mathbf{A}'^c_{a,b} \mathbf{Y}_{j,b} \\ &= \sum_c \mathrm{Tr}(\mathbf{A}^c (\mathbf{X} \mathbf{A}'^c \mathbf{Y}^\intercal)^\intercal) \end{aligned} \tag{4.21}$$

$$\mathbf{x}^\intercal \mathbf{M} \mathbf{y} \leftrightarrow \sum_c \mathrm{Tr}\left(\mathbf{A}^c (\mathbf{X} \mathbf{A}'^c \mathbf{Y}^\intercal)^\intercal\right) \tag{4.22}$$

Algorithm 7: Integer Projected Fixed Point with Adjacency Matrices

```
Given $\mathbf{A}$, $\mathbf{A}'$. Set $s = \sum_c \mathrm{Tr}\,(\mathbf{A}^c(\mathbf{X}_0\mathbf{A}'^c(\mathbf{X}_0)^\intercal)^\intercal)$
repeat
    $\mathbf{B} = P_d(\sum_c [(\mathbf{A}^c)^\intercal\mathbf{X}_t\mathbf{A}' + \mathbf{A}^c\mathbf{X}_t(\mathbf{A}'^c)^\intercal])$,
    $c = \sum_c [\mathrm{Tr}\,(\mathbf{A}^c[\mathbf{X}_t\mathbf{A}'^c(\mathbf{B}-\mathbf{X}_t)^\intercal]^\intercal)]$,
    $d = \sum_c [\mathrm{Tr}\,((\mathbf{B}-\mathbf{X}_t)^\intercal\mathbf{A}^c(\mathbf{B}-\mathbf{X}_t)\mathbf{A}'^c)]$
    if $d \geq 0$ then
        $\mathbf{X}_{t+1} = \mathbf{B}$
    else
        $r = \min\{-c/d, 1\}$, $\mathbf{X}_{t+1} = \mathbf{X}_t + r(\mathbf{B}-\mathbf{X}_t)$
    end
    if $\sum_c [\mathrm{Tr}\,(\mathbf{A}^c(\mathbf{B}\mathbf{A}'^c\mathbf{B}^\intercal)^\intercal)] \geq s$ then
        $s = \sum_c [\mathrm{Tr}\,(\mathbf{A}^c(\mathbf{B}\mathbf{A}'^c\mathbf{B}^\intercal)^\intercal)]$, $\mathbf{X}^\star = \mathbf{B}$
    end
until $\mathbf{X}_{t+1} = \mathbf{X}_t$;
```

4.5.3 Reweighted Random Walks Matching

RRWM finds the solution by simulating page-rank style random walks [38]. RRWM can be easily reinterpreted only using Eq.(4.20) and two versions of RRWM is explained in Algorithm 9 and 10. Sinkhorn bistochastic normalization [49, 1] is adopted for RRWM method as the projection $P_d(\cdot)$.

In Algorithm 11, modified RRWM (mRRWM) is introduced for large-size graph matching problem since Sinkhorn nomalization is not robust when graphs are large. For faster computation with large graphs, greedy mapping function of [10] is adopt for $P_d(\cdot)$ and α becomes an adaptive parameter which have high value when $P_d(\mathbf{Y})$ is similar to $\mathbf{Y}$. Thus, the proposed mRRWM focuses on the discretized solution

Algorithm 8: Integer Projected Fixed Point with Affinity Matrix

```
Given $\mathbf{M}$. Set $s = \mathbf{x}_0^\intercal \mathbf{M} \mathbf{x}_0$
repeat
    $\mathbf{b} = P_d(\mathbf{M}\mathbf{x}_t)$, $c = \mathbf{x}_t^\intercal \mathbf{M}(\mathbf{b} - \mathbf{x}_t)$, $d = (\mathbf{b} - \mathbf{x}_t)^\intercal \mathbf{M}(\mathbf{b} - \mathbf{x}_t)$
    if $d \geq 0$ then
        $\mathbf{x}_{t+1} = \mathbf{b}$
    else
        $r = \min\{-c/d, 1\}$, $\mathbf{x}_{t+1} = \mathbf{x}_t + r(\mathbf{b} - \mathbf{x}_t)$
    end
    if $\mathbf{b}^\intercal \mathbf{M} \mathbf{b} \geq s$ then
        $s = \mathbf{b}^\intercal \mathbf{M} \mathbf{b}$, $\mathbf{x}^\star = \mathbf{b}$
    end
until $\mathbf{x}_{t+1} = \mathbf{x}_t$;
```

$P_d(\mathbf{Y})$ only when $P_d(\mathbf{Y})$ is reliable discretized solution.

Algorithm 9: Reweighted Random Walks Matching with Adjacency Matrices

```
Given $\mathbf{A}$, $\mathbf{A}'$, and parameters $\{\alpha, \beta\}$
repeat
    $\mathbf{Y} = \sum_c [\mathbf{A}^c \mathbf{X}_t (\mathbf{A}'^c)^\intercal]$, $\mathbf{Z} = \exp(\beta \mathbf{Y} / \max \mathbf{Y})$, $\mathbf{Z} = P_d(\mathbf{Z})$.
    $\mathbf{X}_{t+1} = \alpha \mathbf{Y} + (1 - \alpha)\mathbf{Z}$.
until $\mathbf{X}$ converges;
```

4.5.4 Factorized Graph Matching

Zhou *et al.* introduced factorized graph matching (FGM) [23, 25] method and showed that FGM outperforms other state-of-the-art methods. FGM calculates the affinity matrix $\tilde{\mathbf{M}}$ (used in Section 4.3.3) then generates attributes $\mathbf{A}$ and $\mathbf{A}'$ by using their

Algorithm 10: Reweighted Random Walks Matching with Affinity Matrix

```
Given $\mathbf{M}$ and parameters $\{\alpha, \beta\}$
repeat
    $\mathbf{y} = \mathbf{M}\mathbf{x}_t$, $\mathbf{z} = \exp(\beta\mathbf{y}/\max\mathbf{y})$, $\mathbf{z} = P_d(\mathbf{z})$. $\mathbf{x}_{t+1} = \alpha\mathbf{y} + (1-\alpha)\mathbf{z}$.
until $\mathbf{x}$ converges;
```

Algorithm 11: Modified Reweighted Random Walks Matching with Adjacency Matrices

```
Given $\mathbf{A}$, $\mathbf{A}'$ and a parameter $\{\beta\}$
repeat
    $\mathbf{Y} = \sum_c [\mathbf{A}^c\mathbf{X}_t(\mathbf{A}'^c)^\intercal]$, $\mathbf{Z} = P_d(\mathbf{Y})$.
    (Element-wise product) $\alpha = ||\mathbf{Z} \odot \mathbf{Y}||_F^\beta$.
    $\mathbf{X}_{t+1} = \alpha\mathbf{Y} + (1-\alpha)\mathbf{Z}$.
until $\mathbf{X}$ converges;
```

proposed factorization technique [23, 25]. FGM relaxes the original objective into convex and concave functions. Concave-Convex Procedure (CCCP) and modified Frank-Wolfe (MFW) method are used for optimizing the relaxed objective function. However, CCCP and MFW require the gradient of the relaxed function which includes $\tilde{\mathbf{M}}$ terms with $\mathcal{O}(n^4)$ memories. Therefore, our proposed conversion rules in Eq.(4.19) and (4.20) cannot interpret ADJ version of FGM. Thus, we briefly explain the original FGM by Algorithm. 12.

4.6 High-order Methods & Reinterpretations

In the section 4.5, we explained two conversion rules for ordinary graph matching algorithms. Our rules transforms calculations of the graph matching objective score

Algorithm 12: Factorized Graph Matching

```
Given $\tilde{\mathbf{M}}$ and parameters $\{\delta, \eta\}$
Factorize $\tilde{\mathbf{M}}$ and find $\tilde{\mathbf{A}}$, $\tilde{\mathbf{A}}'$
for $\alpha = 0 : \delta : 1$ do
    if $\alpha \leq \eta$ then
        Do Concave-Convex Procedure (CCCP)
    else
        Do modified Frank-Wolfe (MFW)
    end
end
```

and gradient of it. Some hypergraph matching approaches, as well, are based on the calculation of those two entities. Therefore, in this section, we will derive another two conversion principles for the hypergraph matching problem. Note that we will discuss the case of third order hypergraph since almost high-order approaches focus on the third order hypergraphs. For simplifying issue, we assume scalar attributes for $\mathbf{T}$, $\mathbf{T}'$, and $\mathbf{H}$, *i.e.*, $C = 1$.

Objective score calculation: In the section 4.3.1 and 4.3.2, the graph matching objective score is already derived in both ADJ and AFF types by Eq.(4.7) and Eq.(4.12). It is straightforward to take them as their counterpart. The first conversion rule becomes;

$$\mathbf{H} \times_1 \mathbf{x} \times_2 \mathbf{x} \times_3 \mathbf{x} \leftrightarrow \left|\left|(\mathbf{T} \times_1 \mathbf{X} \times_2 \mathbf{X} \times_3 \mathbf{X}) \odot \mathbf{T}'\right|\right|_F . \tag{4.23}$$

Objective gradient calculation: The second conversion rule can be derived by taking gradient of both side in Eq.(4.23) *w.r.t.* $\mathbf{x}$ and $\mathbf{X}$, respectively. From the AFF objective score and super-symmetry assumption ($\mathbf{H}_{ijk}$ is identical under any permutation of i, j, and k),

$$S(\mathbf{x}) = \mathbf{H} \times_1 \mathbf{x} \times_2 \mathbf{x} \times_3 \mathbf{x} = \sum_i \sum_j \sum_k \mathbf{H}_{ijk}\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k,$$
$$\frac{\partial}{\partial \mathbf{x}} S(\mathbf{x}) \to \mathbf{H} \times_2 \mathbf{x} \times_3 \mathbf{x}. \tag{4.24}$$

Meanwhile, gradient of ADJ objective score function $S(X)$ and its gradient become as the following:

$$\begin{aligned}
S(\mathbf{X}) &= \left|\left|(\mathbf{T} \times_1 \mathbf{X} \times_2 \mathbf{X} \times_3 \mathbf{X}) \odot \mathbf{T}'\right|\right|_F \\
&= \sum_a \sum_b \sum_c \left[\mathbf{T}'_{abc} \cdot \sum_u \sum_v \sum_w [\mathbf{T}_{uvw}\mathbf{X}_{ua}\mathbf{X}_{vb}\mathbf{X}_{wc}]\right], \\
\frac{\partial}{\partial \mathbf{X}} S(\mathbf{X})_{i,j} &= \sum_b \sum_c \left[\mathbf{T}'_{jbc} \cdot \sum_v \sum_w [\mathbf{T}_{ivw}\mathbf{X}_{vb}\mathbf{X}_{wc}]\right] \\
&+ \sum_a \sum_c \left[\mathbf{T}'_{ajc} \cdot \sum_u \sum_w [\mathbf{T}_{uiw}\mathbf{X}_{ua}\mathbf{X}_{wc}]\right] \\
&+ \sum_a \sum_b \left[\mathbf{T}'_{abj} \cdot \sum_u \sum_v [\mathbf{T}_{uvi}\mathbf{X}_{ua}\mathbf{X}_{vb}]\right], \\
\frac{\partial}{\partial \mathbf{X}} S(\mathbf{X})_{i,j} &\to \left|\left|(\mathbf{T} \times_2 \mathbf{X} \times_3 \mathbf{X})_{i,:,:} \odot \mathbf{T}'_{j,:,:}\right|\right|_F
\end{aligned} \tag{4.25}$$

Note that $\odot$ represents element-wise product, *i.e.*, Hadamard product and $\frac{\partial}{\partial \mathbf{X}} S(\mathbf{X})_{i,j}$ represents (i,j) component $\frac{\partial}{\partial \mathbf{X}} S(\mathbf{X})$ while $\frac{\partial}{\partial \mathbf{X}} S(\mathbf{X}) \in \mathbb{R}^{n \times n'}$. Unlike the second order conversion rule, we are unable to find more simpler notation for representing gradient conversion rule. Again, the super-symmetry assumption of $\mathbf{T}$ and $\mathbf{T}'$ explains the last relation of Eq.(4.25). By combining Eq.(4.24) and Eq.(4.25), we can derive the second conversion rule for the hypergraph matching:

$$(\mathbf{H} \times_2 \mathbf{x} \times_3 \mathbf{x})_{i,j} \leftrightarrow \left|\left|(\mathbf{T} \times_2 \mathbf{X} \times_3 \mathbf{X})_{i,:,:} \odot \mathbf{T}'_{j,:,:}\right|\right|_F \tag{4.26}$$

In the following subsections, four well-known hypergraph matching algorithms are introduced. Hypergraph matching (HGM) by Zass and Shashua [20], singular value decomposition based high order matching (SVD) by Chertok and Keller [36], Tensor power iteration graph matching (TM) by Duchenne *et al.*, Reweighted Random Walks Hypergraph Hatching (RRWHM) [2] by Lee *et al.*, and Discrete Hypergraph Matching (DHM) [37] by Yan *et al.*. Those AFF formulation oriented four techniques are conceptually introduced and then re-interpreted if possible.

4.6.1 Hypergraph Matching by Zass and Shashua

In [20], Zass and Shashua interpret the graph matching problem in probabilistic sense and proposed an order-free graph matching algorithm. They relax the assignment $\mathbf{X} \in \{0,1\}^{n\times n'}$ into $\mathbf{X} \in (0,1)^{n\times n'}$ by regarding $\mathbf{X}_{i,a}$ as the confidence score of $\mathcal{V}_i$ being matched to $\mathcal{V}'_a$. In ideal case, $\mathbf{H}_{ia,jb,kc} = 1$ only if $\mathbf{X}_{ia} = \mathbf{X}_{jb} = \mathbf{X}_{kc} = 1$. This observation leads to a principle that $\mathbf{X}\otimes\mathbf{X}\otimes\mathbf{X}$ should be close to $\mathbf{H}$. Thus, their objective for solving hypergraph matching problem becomes as the following: $\mathbf{X}^* = \text{argmin}_{\mathbf{X}} \left|\left|\mathbf{H} - \otimes^3\mathbf{X}\right|\right|$. [1] In Algorithm 13, HGM is conceptually introduced, however, we are unable to find HGM's ADJ counterpart since our conversion rule works only when an algorithm contains calculations of objective or gradient of it.

Algorithm 13: Hypergraph Matching by Zass and Shashua with Affinity Tensor

```
Given $\mathbf{H}$
Marginalize $\mathbf{H}$ by $\mathbf{x} = \sum_{j,k} \mathbf{H}_{:,i,j}$.
Apply successive projection on $\mathbf{x}$ for satisfying
$\mathbf{x1} = \mathbf{1}$ and $\mathbf{x}^\intercal\mathbf{1} = \mathbf{1}$.
```

[1] $\otimes^3\mathbf{X}$ is used for abbreviating $\mathbf{X}\otimes\mathbf{X}\otimes\mathbf{X}$ while $\otimes$ represents Kronecker product.

4.6.2 SVD-based Hypergraph Matching

Chertok and Keller [36] introduced SVD-based hypergraph matching algorithm. They interpreted the affinity tensor $\mathbf{H}$ and the assignment vector $\mathbf{x}$ in probabilistic manner, similar to HGM [20]. Then, they calculates rank-1 approximation of $\mathbf{H}$ by SVD as an approximation. The tensor unfolding technique is introduced for applying SVD onto $\mathbf{H}$. Further gradient steps are applied for making their solution nonnegative. A brief algorithm is explained in Algorithm 14. The SVD step in their method is unable to be converted by our conversion rule thus we only describe AFF version of the method.

Algorithm 14: Singular Value Decomposition for Hypergraph Matching with Affinity Tensor

```
Given $\mathbf{H}$
Unfold $\mathbf{H} \in \mathbb{R}^{nn' \times nn' \times nn'}$ into $\mathbf{H}_u \in \mathbb{R}^{(nn')^2 \times nn'}$.
Calculate the right-leading singular vector $\mathbf{x}$ by eigen-decomposition onto
$\mathbf{H}_u \mathbf{H}_u^\intercal$.
repeat
    $\mathbf{x}_{t+1} = \mathbf{H} \times_2 \mathbf{x}_t \times_3 \mathbf{x}_t$, $\mathbf{x}_{t+1} = \mathbf{x}_{t+1} / ||\mathbf{x}_{t+1}||$
until $\mathbf{x}$ converges;
```

4.6.3 Tensor Power Iteration based Hypergraph Matching

Duchenne *et al.* proposed the tensor power iteration algorithm for hypergraph matching [35]. They extend the concept of power iteration to high order tensor for rank-1 approximation of $\mathbf{H}$. Their simple but powerful technique produces state-of-the-art performance in hypergraph matching problems. The key concept of TM method is explained in Algorithm 16 and its ADJ counterpart is in Algorithm 15. The high order power iteration step is mainly based on the gradient calculation of the hypergraph objective function thus we can easily derive ADJ counterpart of TM.

Algorithm 15: Tensor Power Iteration with Adjacency Tensor

```
Given $\mathbf{T}$, $\mathbf{T}'$
repeat
    $\mathbf{Y}_{i,j} = ||(\mathbf{T} \times_2 \mathbf{X} \times_3 \mathbf{X})_{i,:,:} \odot \mathbf{T}'_{j,:,:}||_F$,
    $\mathbf{X} = \mathbf{Y}/||\mathbf{Y}||$
until $\mathbf{X}$ converges;
```

Algorithm 16: Tensor Power Iteration with Affinity Tensor

```
Given $\mathbf{H}$
repeat
    $\mathbf{x}_{t+1} = \mathbf{H} \times_2 \mathbf{x}_t \times_3 \mathbf{x}_t$, $\mathbf{x}_{t+1} = \mathbf{x}_{t+1}/||\mathbf{x}_{t+1}||$
until $\mathbf{x}$ converges;
```

4.6.4 Reweighted Random Walks for Hypergraph Matching

Lee *et al.* proposed high order random walks interpretation for hypergraph matching [2]. By defining random walker's behavior on hypergraph, they successfully extend their previous RRWM [1] algorithm to hypergraph matching problems. Their affinity-preserving random walk transition can be calculated by gradient of the objective function thus their method can be transformed into ADJ counterpart. The original RRWHM is explained briefly in Algorithm 17 and our new interpretation is explained in Algorithm 18.

Algorithm 17: Reweighted Random Walks Hypergraph Matching with Adjacency Tensor

```
Given $\mathbf{T}$, $\mathbf{T}'$ and parameters $\{\alpha, \beta\}$
repeat
    $\mathbf{Y}_{i,j} = ||(\mathbf{T} \times_2 \mathbf{X} \times_3 \mathbf{X})_{i,:,:} \odot \mathbf{T}'_{j,:,:}||_F$, $\mathbf{Z} = \exp(\beta \mathbf{Y} / \max \mathbf{Y})$, $\mathbf{Z} = P_d(\mathbf{Z})$
    $\mathbf{X} = \alpha \mathbf{Y} + (1 - \alpha)\mathbf{Z}$
until $\mathbf{X}$ converges;
```

Algorithm 18: Reweighted Random Walks Hypergraph Matching with Affinity Tensor

```
Given $\mathbf{H}$ and parameters $\{\alpha, \beta\}$
repeat
    $\mathbf{y} = \mathbf{H} \times_2 \mathbf{x} \times_3 \mathbf{x}$, $\mathbf{z} = \exp(\beta \mathbf{y} / \max \mathbf{y})$, $\mathbf{z} = P_d(\mathbf{z})$
    $\mathbf{x} = \alpha \mathbf{y} + (1 - \alpha)\mathbf{z}$
until $\mathbf{x}$ converges;
```

4.6.5 Discrete Hypergraph Matching

Yan *et al.* proposed discrete hypergraph matching [37]. In DHM, an adaptive linearized update scheme on discrete domain is proposed. By designing a updating scheme similar to mth-order Markov Chain and iterating on discrete domain, DHM achieves robust performance on hypegraph matching problems. The iteration procedure of DHM is summarized in Algorithm 19, however, DHM is not possible to be converted into ADJ version since our conversion rule is not able to cover tensor product with histories of $\mathbf{X}$.

Algorithm 19: Discrete Hypergraph Matching

```
Given $\mathbf{H}$ and a parameter $\{m\}$
repeat
    $\mathbf{x}_{k+1} = P_d(\mathbf{H} \times_1 \mathbf{x}_{k+2-m} \times_2 \cdots \times_{m-1} \mathbf{x}_k)$
until $\mathbf{x}$ converges;
```

4.7 Experiments & Comparison

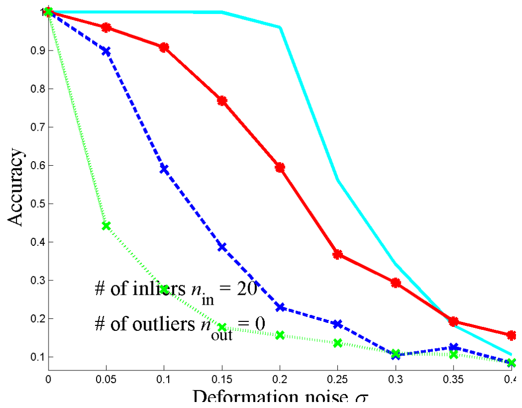
In experiment section, we compare existing representative graph and hypergraph matching approaches. In addition, we demonstrate the equivalence of the two graph matching formulation as well. For ordinary graph matching problem (or second order), previously explained SM, IPFP, RRWM, and FGM are compared in various settings. Meanwhile, the performances of HGM, SVD, TM, RRWHM, and DHM are demonstrated on hypergraph matching problems. We conduct our experiments on MATLAB using author provided codes (some methods are modified for fare comparison). Our novel interpretation to conventional algorithms enable them to solve large-size (over 1000 nodes) graph matching problem. Every experiments are performed 30 to 100 times and their average values are reported.

Synthetic random graph generation: Random graphs are synthetically generated to compare performances of previously mentioned graph and hypergraph matching methods. Point-set matching problems are simulated in some literatures such as [10] or [2]. However, we generated graphs by assigning random attributes both on nodes and edges, thus we synthesize an unbiased situation [1].

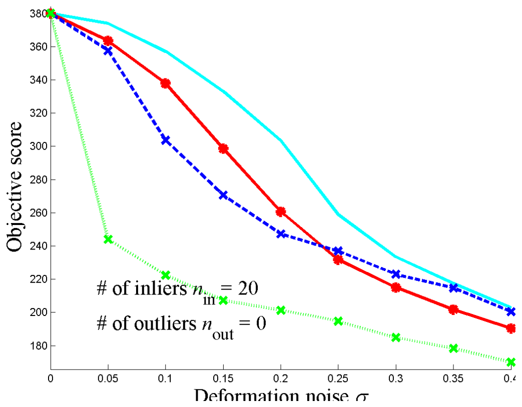
Oridinary random graph: We generate graphs $\mathcal{G}$, $\mathcal{G}'$ by the following procedure. At first, the graph attribute $\mathbf{A}$ is randomly generated using uniform distribution $\mathcal{U}(0,1)$; *i.e.*, $\mathbf{A} \sim \mathcal{U}(0,1)^{n \times n}$. Note that $n = n' = n_{in} + n_{out}$, where n_{in} and n_{out} are inlier and outlier numbers, respectively. To synthesize more difficult matching problem, we assume that there is no attribute values for nodes so we set $\mathbf{A}_{i,i} = 0$. Then, $\mathbf{A}'$ is generated by following $\mathbf{A}' = \mathbf{A} + \mathbf{N}$ and $\mathbf{N} \sim \mathcal{N}(0,\sigma)^{n' \times n'}$ for simulating deformation noises using Gaussian distribution $\mathcal{N}$. Finally, outlier noises are imposed by overwriting noises to $\mathbf{A}$ and $\mathbf{A}'$ with $\mathbf{A}_{1:n_{out},1:n_{out}} \sim \mathcal{U}(0,1)^{n_{out} \times n_{out}}$ and $\mathbf{A}'_{1:n_{out},1:n_{out}} \sim \mathcal{U}(0,1)^{n_{out} \times n_{out}}$. $\mathbf{A}$ and $\mathbf{A}'$ are randomly permuted for preventing one method accidentally achieving high score.

High-order random graph: Random hypergraphs are synthesized in similar manner with ordinary random graphs. First, attribute tensors are generated by following $\mathbf{T} \sim \mathcal{U}(0,1)^{n \times n \times n}$ and $\mathbf{T}' \sim \mathcal{U}(0,1)^{n' \times n' \times n'}$. Our conversion rules are based on super-symmetry thus tensor generation should follow $\mathbf{T}_{ijk} = \mathbf{T}_{\pi(i)\pi(j)\pi(k)}$ while π is any permutation of $\{1,2,3\}$. Next, deformation noises are generated by $\mathbf{N} \sim \mathcal{N}(0,\sigma)^{n_{in} \times n_{in} \times n_{in}}$ ($\mathbf{N}$ is also super-symmetric). We overwrite $\mathbf{T}'$ by $\mathbf{T}'_{1:n_{in},1:n_{in},1:n_{in}} = \mathbf{T}_{1:n_{in},1:n_{in},1:n_{in}} + \mathbf{N}$ and then permute indices of $\mathbf{T}'$ in the same manner as ordinary graph case.

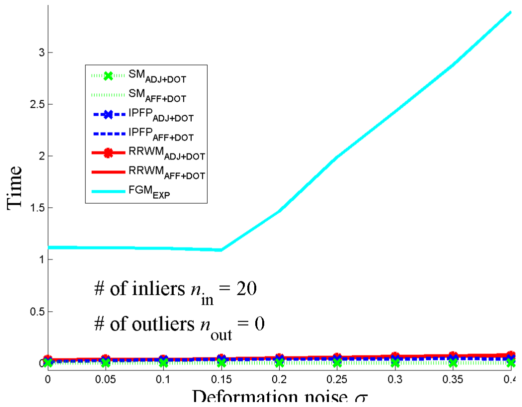
Equivalence between two formulations: We theoretically (Section 4.3.3) and



(a) matching accuracy

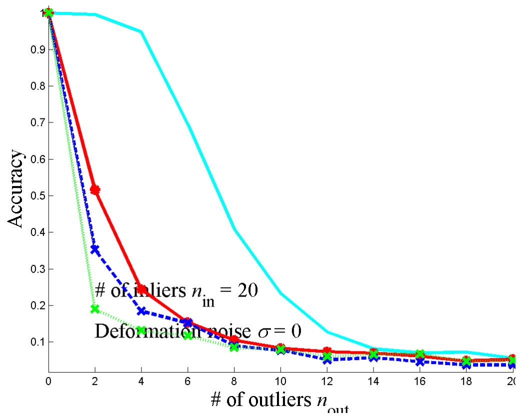


(b) objective score

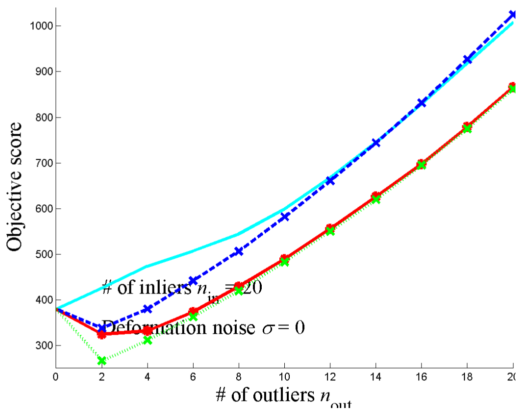


(c) execution time

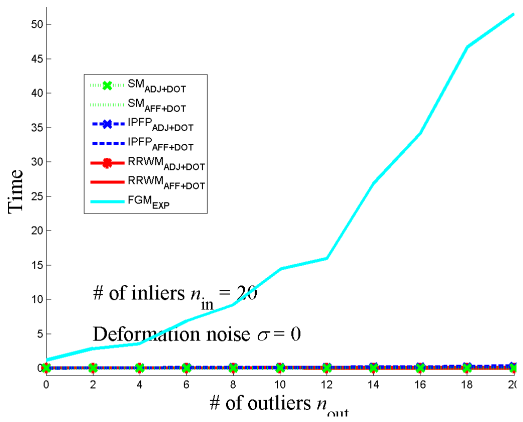
Figure 4.1: Performance of DOT measure according to varying deformation noise. Performances of SM, IPFP, and RRWM are reported in two formulation types. ADJ form is expressed with marker and AFF form is expressed without marker. Inlier number n_{in} is fixed to 20, while deformation scale σ varies from 0 to 0.4.



(a) matching accuracy

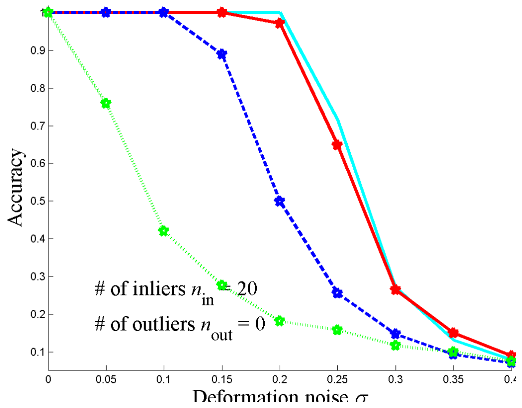


(b) objective score

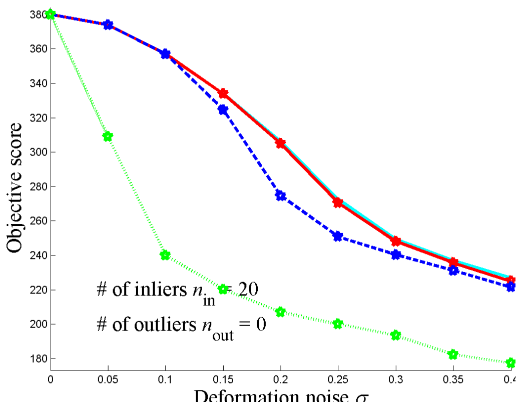


(c) execution time

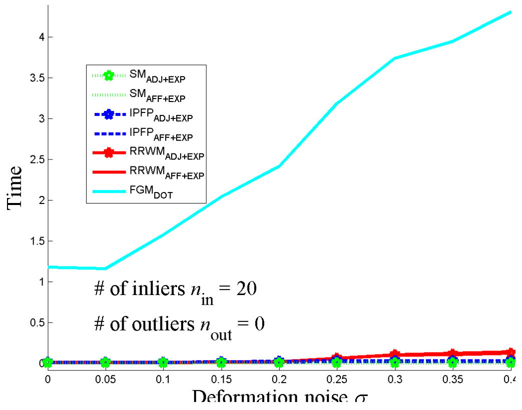
Figure 4.2: Performance of DOT measure according to varying outlier noise. Performances of SM, IPFP, and RRWM are reported in two formulation types. ADJ form is expressed with marker and AFF form is expressed without marker. Inlier number n_{in} is fixed to 20, while outlier number n_{out} varies from 0 to 20.



(a) matching accuracy

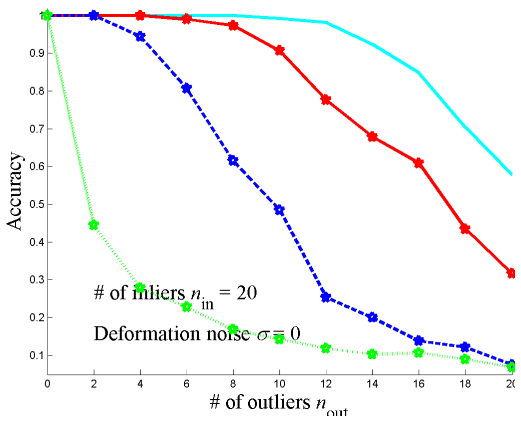


(b) objective score

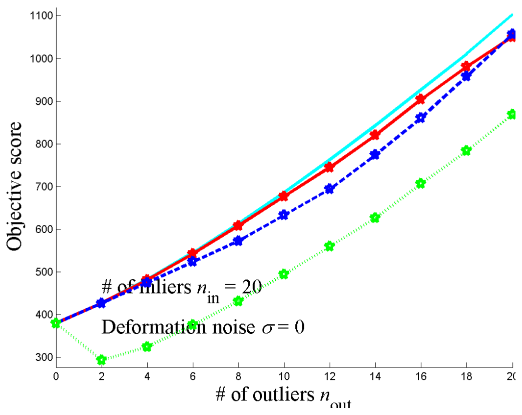


(c) execution time

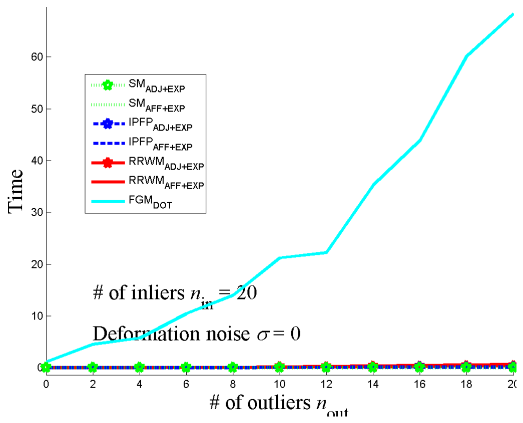
Figure 4.3: Performance of EXP affinity measure according to deformation noise. Performances of SM, IPFP, and RRWM are reported in two formulation types. ADJ form is expressed with marker and AFF form is expressed without marker. Inlier number n_{in} is fixed to 20, while deformation scale σ varies from 0 to 0.4.



(a) matching accuracy

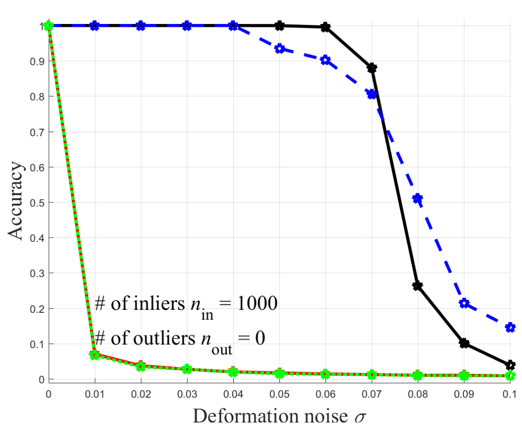


(b) objective score

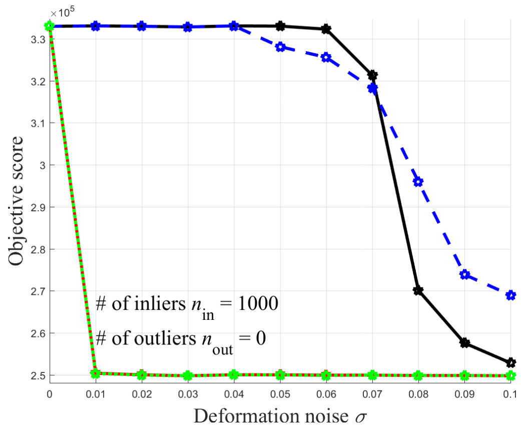


(c) execution time

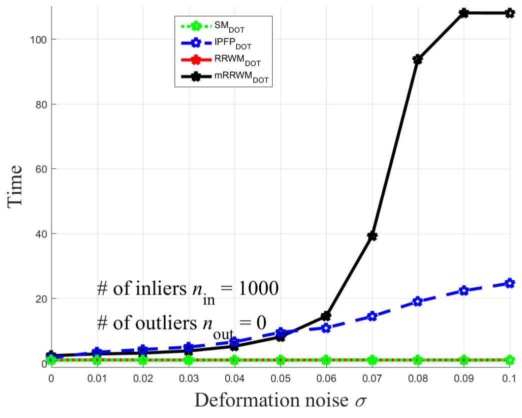
Figure 4.4: Performance of EXP affinity measure according to outlier noise. Performances of SM, IPFP, and RRWM are reported in two formulation types. ADJ form is expressed with marker and AFF form is expressed without marker. Inlier number n_{in} is fixed to 20, while outlier number n_{out} varies from 0 to 20.



(a) matching accuracy

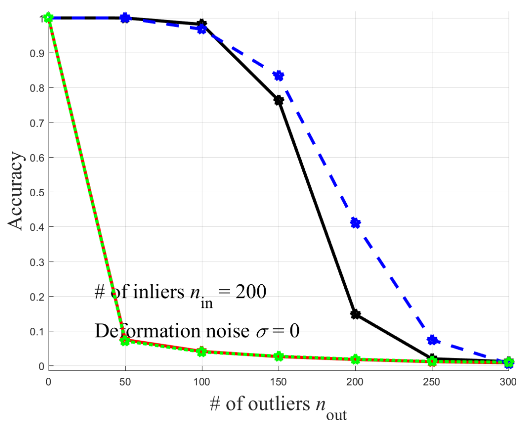


(b) objective score

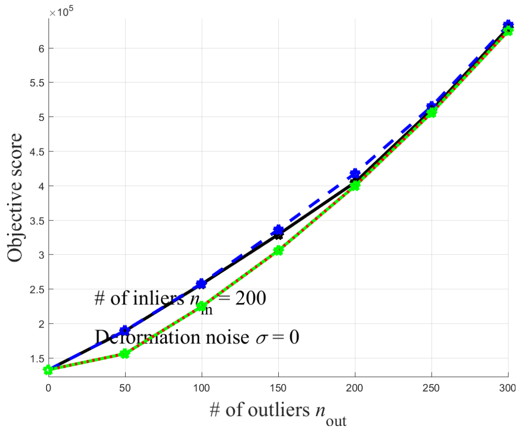


(c) execution time

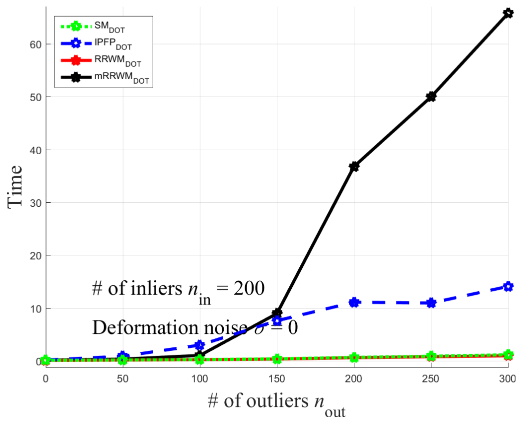
Figure 4.5: Performance on large-size graph matching problem according to deformation noise. Large-size graph matching problem can be solved only with adjacency-based formulation with dot-product affinity measure. Performances of SM, IPFP, RRWM, and mRRWM are reported. Inlier number n_{in} is fixed to 1000, while deformation scale σ various from 0 to 0.1.



(a) matching accuracy

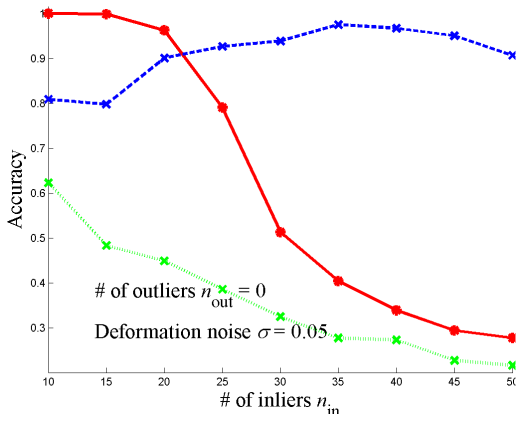


(b) objective score

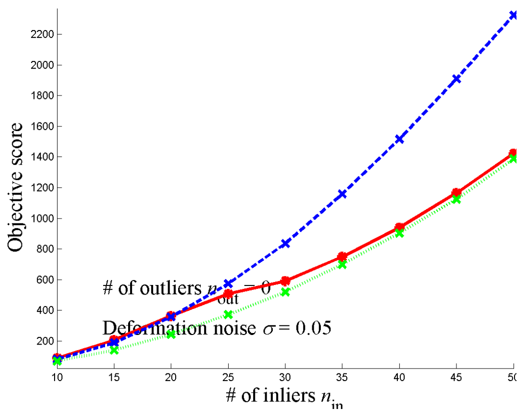


(c) execution time

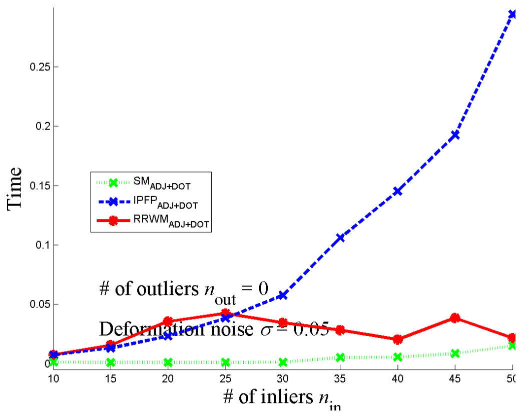
Figure 4.6: Performance on large-size graph matching problem according to outlier noise. Large-size graph matching problem can be solved only with adjacency-based formulation with dot-product affinity measure. Performances of SM, IPFP, RRWM, and mRRWM are reported. Inlier number n_{in} is fixed to 200, while outlier number n_{out} varies from 0 to 300.



(a) matching accuracy

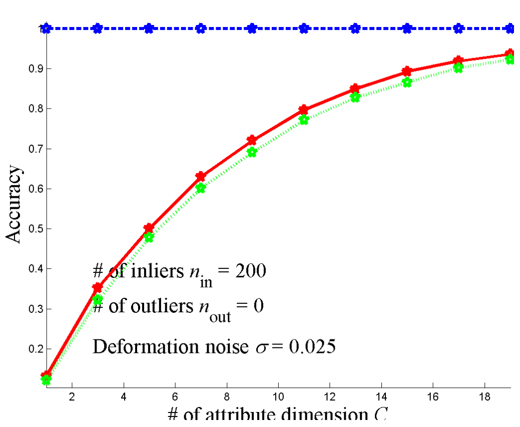


(b) objective score

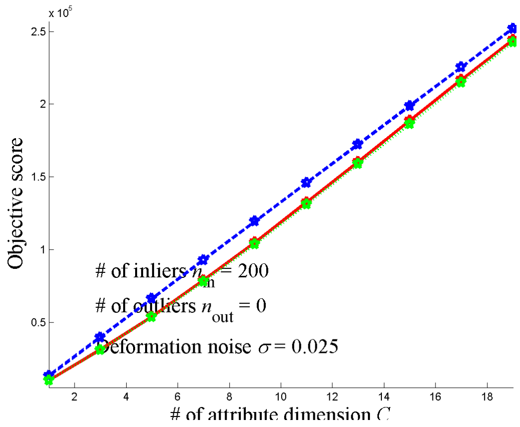


(c) execution time

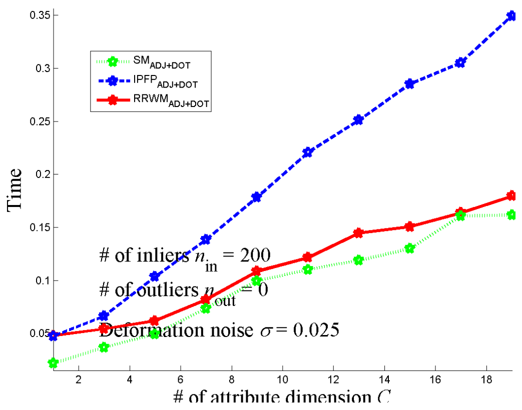
Figure 4.7: Performance according to the graph size. Adjacency-based formulation with dot product is used. Inlier number n_{in} varies from 10 to 50, while outlier number n_{out} and deformation noise σ is fixed to 0 and 0.05, respectively.



(a) matching accuracy



(b) objective score



(c) execution time

Figure 4.8: Performance according to the attribute dimension. Adjacency-based formulation with dot product is used. Inlier number n_{in}, outlier number n_{out}, and deformation scale σ is fixed to 200, 0, and 0.025, respectively while the attribute dimension varies from 1 to 19.

experimentally (Figs. 4.1, 4.2, 4.3, and 4.4) demonstrate that both adjacency-based formulation and affinity-based formulation produces the same solutions. For SM, IPFP, and RRWM, users can notice that each method shows the same performance with both ADJ (with marker) and AFF (without marker) form. As mentioned in Section 4.5.4, FGM solves the problem with its original form (AFF form).

Affinity measure comparison: DOT and EXP measures solve the same graph matching problem but produce different results. Comparing Figs. 4.1 and 4.3 reveals that EXP measure is more robust than DOT measure under deformation noise. FGM, surprisingly, shows almost the same performance on both measures under deformation noises. Similarly, we can compare the performances of the two measures under outlier noises by checking Figs. 4.2 and 4.4. With outlier noises, performance gap between the two measures is more severe than deformation noises.

Synthetic problem with large graphs: Applying graph matching techniques in computer vision fields inevitably handles image features which are usually more than hundreds or thousands. To our best knowledge, there is no reported work handling fully connected graph matching problem with node size n more than 1000.

We tested previously mentioned graph matching algorithms on huge graphs. FGM is omitted in large-size graph matching problem since we are unable to get adjacency-based FGM method. IPFP method originally performs Hungarian algorithm [48] in every iteration for satisfying one-to-one constraints. However the computational complexity of Hungarian algorithm is $\mathcal{O}(n^3)$ which makes runtime of IPFP impractical. Instead, we modify IPFP to perform greedy manner projection for speed-up. In addition, the proposed mRRWM is also compared in large-size graph matching experiment.

Performances of the three methods (SM, IPFP, RRWM, mRRWM) on the large-size graph matching problem are reported in Figs. 4.5 and 4.5. Note that, for all methods, parameters are fixed throughout all experiments. For outlier tests in Fig. 4.6, we set the attribute dimension C as 10 since outlier noises dramatically decrease performances. Thus, we increase attribute dimension C which makes attributes more informative and robust to noises. Scalar attribute is not informative for large-size graph matching, thus we increase the dimension of attributes to 10. It is remarkable that IPFP and mRRWM show reasonable performances on the large-size graph while SM and RRWM fail to find correct matches. IPFP and mRRWM methods (among the four methods discussed) which restrict their solution space satisfying the matching constraints during iterations. Other methods relax their solution space on real numbers and we analyze that this is the reason why IPFP shows the best performances with large graphs.

We also examine when the graph matching problem becomes large one. In Fig. 4.7, RRWM loses its strength when the inlier number n_{in} exceeds 20. However, IPFP shows reasonable performances throughout various sizes of graphs. This performance loss can be prevented by using more robust affinity measure such as exponential measure, or increasing the attribute dimension C.

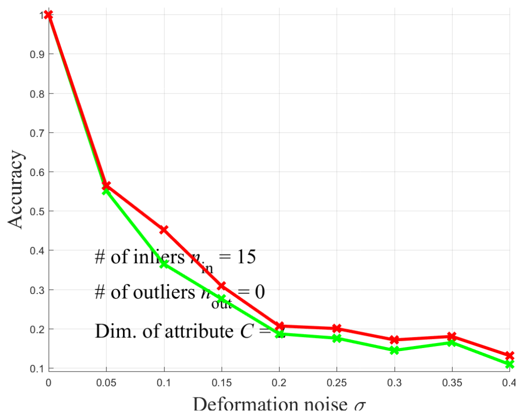
Effect of the attribute dimension: We tested effect of the attribute dimension C in Fig. 4.8. As attribute dimension increases, attributes become more informative which results better performances. IPFP surely outperform others and it is consistent with Figs. 4.5 and 4.6. We can observe that increasing attribute dimension is recommended strategy for overcoming deformation noises. For outlier noises, however, higher attribute dimension is not appropriate solution for SM and RRWM. We can conclude that IPFP is the only solution for solving large-size graph matching problem under various noise conditions.

Performance on the hypergraph matching problem: Finally, performances of hypergraph matching algorithms [20, 35, 36, 2, 37] are compared and reported in Figs. 4.9, 4.10, 4.11, and 4.12. As mentioned in Section 4.6, HGM, SVD, DHM methods are unable to solve ADJ formulation, thus, only TM and RRWHM are compared for ADJ formulation which is reported in Figs. 4.9 and 4.10. In memory aspect, ADJ formulation surely has advantage than AFF formulation, however actual execution times of ADJ are reported longer than AFF formulation since we optimize our implementation of AFF formulation part by using C-MEX. We assume that ADJ form is faster and requires less memories in the same implementation environment. In performance aspect, AFF formulation clearly outperforms ADJ formulation. Among four representative algorithms, TM and RRWHM show state-of-the-art performances.

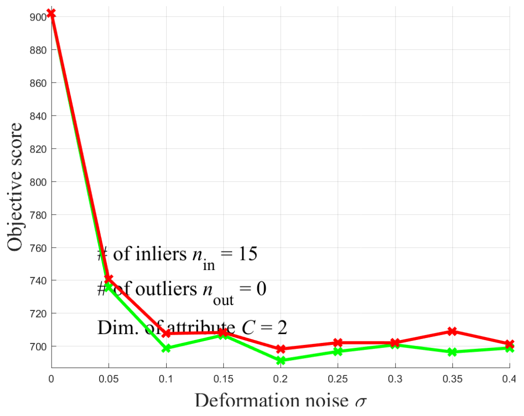
Performance on image feature correspondence problem: Performances of both graph and hypergraph algorithms for real image experiments are also evaluated and report. For constructing graphs and hypergraphs for feature matching experiment, Willow object class dataset from Cho *et al.* [59] is utilized. In Willow dataset, about 300 images from 5 different categories are manually annotated with 10 distinct points. In total, there are more than 10,000 images pairs for the same category. 5 graph matching and 5 hypergraph matching algorithms conduct image feature correspondence tasks and their average performances are reported in Table 4.1 and Table 4.2. For calculating 3rd-order affinity tensor $\mathbf{H}$, as illustrated in Fig. 2.8, a sum of sine value differences between corresponding angles [20, 35, 2] was adopted:

$$d_{ia} = |\sin(\theta_i) - \sin(\theta'_a)|,$$

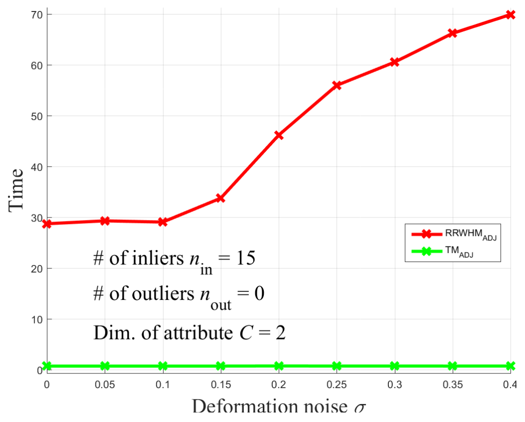
$$\mathbf{H}_{ia,jb,kc} = \exp\Big(-(d_{ia} + d_{jb} + d_{kc})/\sigma_s \Big), \tag{4.27}$$



(a) matching accuracy

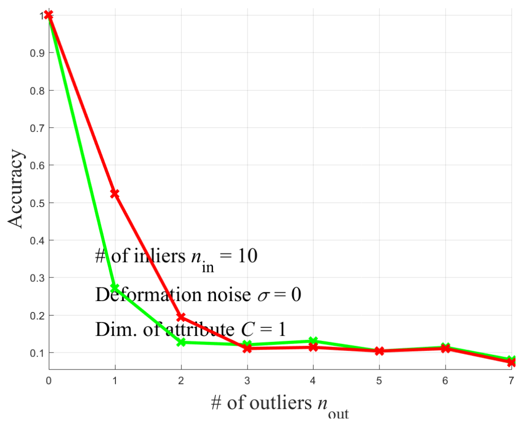


(b) objective score

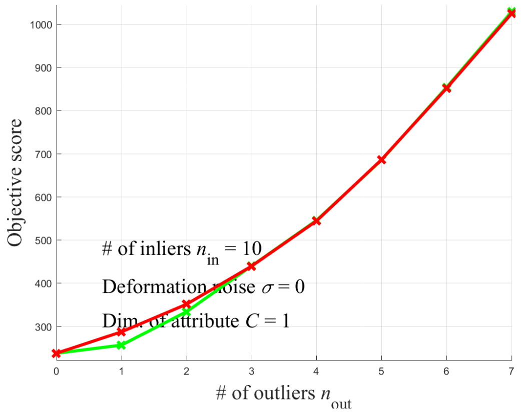


(c) execution time

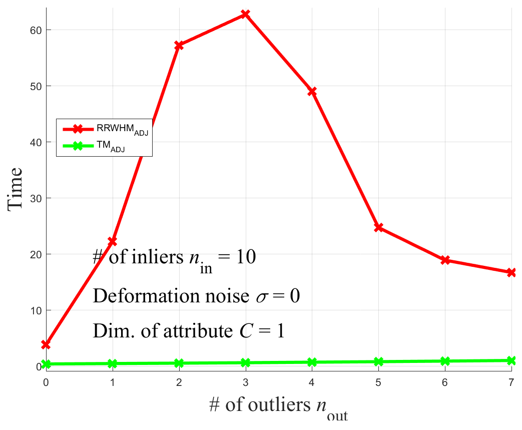
Figure 4.9: Performance of DOT measure on hypergraph matching problem according to deformation noise. Performances of RRWHM and TM are reported. Inlier number n_{in} and attribute dimension C are fixed to 15 and 2, respectively, while deformation scale σ varies from 0 to 0.4.



(a) matching accuracy

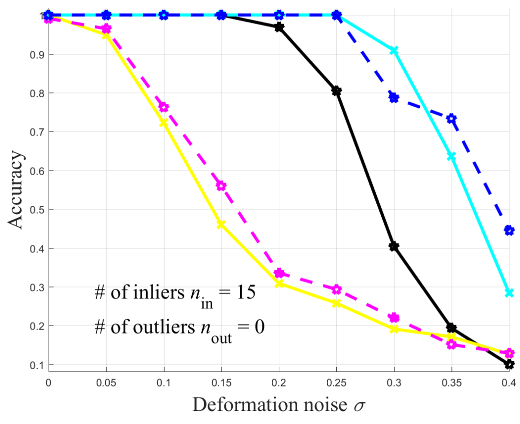


(b) objective score

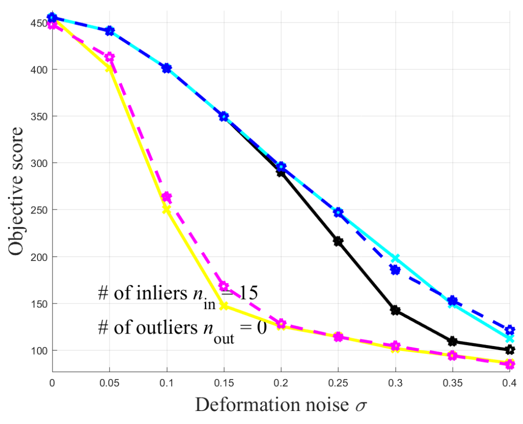


(c) execution time

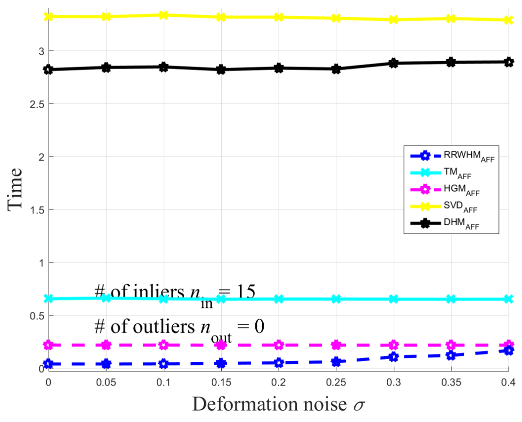
Figure 4.10: Performance of DOT measure on hypergraph matching problem according to outlier noise. Performances of RRWHM and TM are reported. Inlier number n_{in} and attribute dimension C are fixed to 10 and 1, respectively, while outlier number n_{out} varies from 0 to 7.



(a) matching accuracy

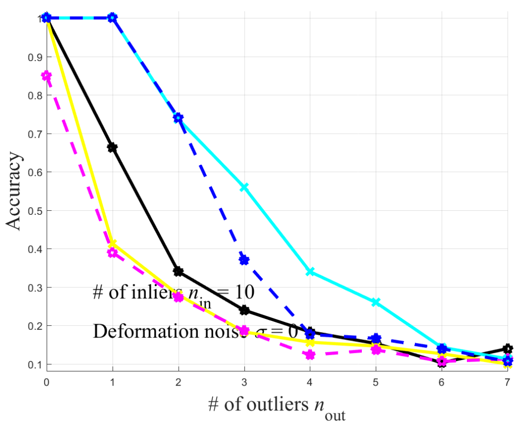


(b) objective score

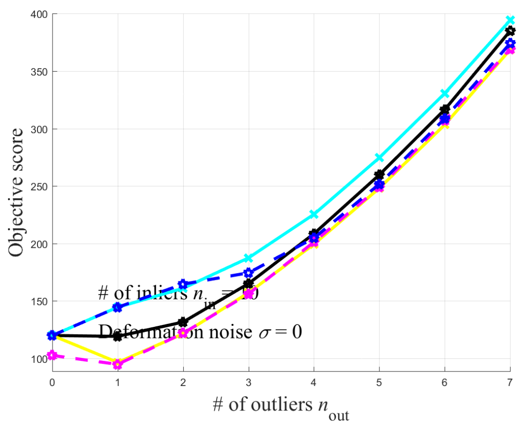


(c) execution time

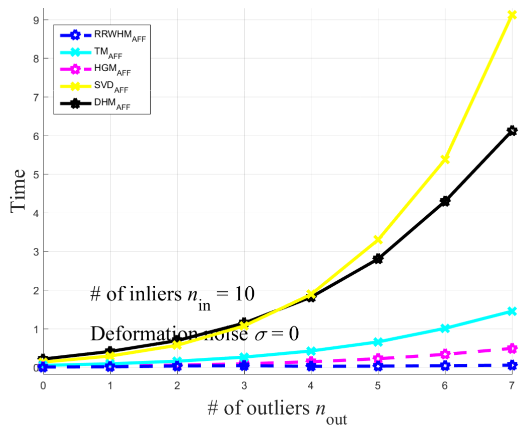
Figure 4.11: Performance of EXP affinity measure on hypegraph matching problem according to deformation changes. Performances of RRWHM, TM, HGM, SVD, and DHM are reported. Inlier number n_{in} and attribute dimension C are fixed to 15 and 2, respectively, while deformation scale σ varies from 0 to 0.4.



(a) matching accuracy



(b) objective score



(c) execution time

Figure 4.12: Performance of EXP affinity measure on hypegraph matching problem according to outlier numbers. Performances of RRWHM, TM, HGM, SVD, and DHM are reported. Inlier number n_{in} and attribute dimension C are fixed to 10 and 1, respectively, while outlier number n_{out} varies from 0 to 7.

Table 4.1: Performances of various graph matching algorithms according to different affinity measures on image feature matching problem.

	DOT measure	EXP measure
SM [10]	0.4133	0.4030
IPFP [21]	0.4405	0.5296
RRWM [1]	0.5100	0.6279
FGM [23]	**0.6263**	**0.6699**
DDMCMC	0.5237	0.6612

where θ_i and θ'_a denote the node angles of candidate correspondence (i, a). The scale parameter σ_s was set to 0.5 for the best performance.

4.8 Conclusion

In this chapter, we investigate two conventional graph matching formulations and their transformational relation, where users are recommended to adopt adjacency-based algorithms since it has computational advantages without performance loss. Note that adjacency and affinity based formulations show different performances. Two affinity measures and their combinations with the formulations are explained. When users have large graphs to be matched, adjacency-based formulation with dot-product measure is essential since only adjacency formulation allows us $\mathcal{O}(n^3)$ computational complexity while others require $\mathcal{O}(n^4)$. In other scenarios, however, negative exponential measure is more robust choice under both deformation and outlier noises. Converting AFF algorithm into ADJ version decreases complexity of

Table 4.2: Performances of various hypergraph matching algorithms according to different affinity measures on image feature matching problem.

	DOT measure	EXP measure
HGM [20]	0.2003	0.1801
SVD [36]	0.2409	0.2324
TM [35]	0.2489	0.3248
RRWHM [2]	**0.3427**	**0.3659**
DHM [37]	0.2574	0.1985

optimizing iterations, however, converting AFF formulation into ADJ version requires singular value decomposition which might be computationally heavy. We introduced reinterpretation of famous graph matching approaches according to the two formulations. It is revealed that IPFP and mRRWM show their strength on large-size graph matching problem compared to ordinary size problem. For hypergraph matching problem, we also introduce two different (affinity-based and adjacency-based) formulations and relate them together. Four representative hypergraph matching algorithms are reinterpreted and their performances are thoroughly compared in synthetic experiments. Once again, adjacency-based formulation takes advantages in computational complexity sense while affinity-based formulation shows robust performance on severe noise conditions.

Chapter 5

Conclusion

5.1 Summary and Contribution of Dissertation

In Chapter 2, a general graph matching formulation for graph matching and hypergraph matching is introduced and a novel algorithm from an Random Walks view is proposed. The proposed algorithm effectively solve graph matching by Random Walks with reweighting jumps, and achieves noise-robust (hyper)graph matching by updating and exploiting the confidences of candidate correspondences simultaneously. Extensive experiments on both synthetic and real problems demonstrated that the proposed algorithm outperforms current state-of-the-art methods for graph matching [10, 16, 21, 18, 19] and hypergraph matching [20, 35, 36] in the presence of outliers and deformation. The proposed method could be useful for real-world matching problems in a wide range of fields.

In Chapter 3, a novel stochastic Markov Chain Monte Carlo based graph matching algorithm which can avoid local minima problem of deterministic approaches is proposed. It adopts Markov Chain Monte Carlo framework with the data-driven state transition proposals, which efficiently explores the solution space of graph matching. The proposed Markov Chain Monte Carlo graph matching algorithm is extended to hypergraph matching problem by re-defining the energy function using hypergraph matching objective score function. Experiments show that our graph matching algorithm is robust to deformation and outliers arising from the practical correspondence problems, and outperforms the state-of-the-art graph matching algorithms.

In Chapter 4, two conventional graph matching formulations and their transformational relation are investigated. Two representative affinity measures (dot-product and negative-exponential) and their combinations with two formulations are explained. When users have large graphs to be matched, adjacency-based formulation with dot-product measure is essential since only adjacency formulation allows us $\mathcal{O}(n^3)$ computational complexity while others require $\mathcal{O}(n^4)$. In other scenarios, however, negative exponential measure is more robust choice under both deformation and outlier noises. We introduced reinterpretation of famous graph matching approaches according to the two formulations. It is revealed that IPFP shows its strength on large size graph matching problem compared to ordinary size problem, and modified RRWM is proposed for solving large graph matching problem. For hypergraph matching problem, we also introduce two different (affinity-based and adjacency-based) formulations and relate them together. Four representative hypergraph matching algorithms are reinterpreted and their performances are thoroughly compared in synthetic experiments. Once again, adjacency-based formulation takes advantages in computational complexity sense while affinity-based formulation shows robust performance on severe noise conditions.

5.2 Future Works

First, the proposed conversion rule between *adjacency* formulation and *affinity* formulation is limited since there are only two rules are derived; objective score and gradient of objective score. If one algorithm requires computation of another quantity except those two, it is impossible to find counterpart of this algorithm. SVD based hypergraph matching algorithm of Chertok and Keller [36], for example, it is required to find *adjacency* version of SVD operation for convert this algorithm into *adjacency* formulation. Thus, additional conversion principles should be derived for our equivalence statement being more general.

Second, in Chapter 2, IPFP of Leordeanu and Herbert [21] shows outstanding robust performance on large size (over 100 nodes) graph matching problem. In fact, IPFP is the only algorithm which can be applied to large size problem. However, there are much of graph matching algorithms [18, 1, 23, 25, 24] which show superior performances on adequate size (under 100 nodes) of graph matching problems. Considering these facts, therefore, it is natural to expect that the performance on large graph matching is not saturated and previous state-of-the-art algorithms can be adjust, tuned, or modified to be robust to large graph matching.

Third, outlier handling graph matching algorithm is necessary. Usually, in real world situation, there exists outlier nodes on both graph $\mathcal{G}$ and $\mathcal{G}'$. Conventional graph matching formulations, however, does not account the presence of outliers nodes in objective score function sense. For example, Suh *et al.* [24] proposed a novel objective function which finds a trade-off between matching size and matching quality by using the following objective function:

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}} \mathbf{x}^\intercal \mathbf{M} \mathbf{x} - \lambda \|\mathbf{x}\|. \tag{5.1}$$

Therefore, expanding or modifying the proposed algorithms to theoretically cover outliers might be a reasonable future work.

Bibliography

[1] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *European Conference on Computer Vision.* Springer, 2010, pp. 492–505.

[2] J. Lee, M. Cho, and K. M. Lee, "Hyper-graph matching via reweighted random walks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1633–1640.

[3] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 03, pp. 265–298, 2004.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the IEEE International Conference on Computer vision*, vol. 2, 1999, pp. 1150–1157.

[5] M. Leordeanu, M. Hebert, and R. Sukthankar, "Beyond local appearance: Category recognition from pairwise interactions of simple features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[6] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and vision computing*, vol. 22, no. 10, pp. 761–767, 2004.

[7] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." in *IJCAI*, vol. 81, no. 1, 1981, pp. 674–679.

[8] M. Cho, J. Lee, and K. M. Lee, "Feature correspondence and deformable object matching via agglomerative correspondence clustering." in *Proceedings of the IEEE International Conference on Computer Vision*, 2009, pp. 1280–1287.

[9] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 26–33.

[10] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, 2005, pp. 1482–1489.

[11] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[12] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Robotics-DL tentative.* International Society for Optics and Photonics, 1992, pp. 586–606.

[13] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman, "Discovering object categories in image collections," 2005.

[14] K. Grauman and T. Darrell, “Pyramid match hashing: Sub-linear time indexing over partial correspondences,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[15] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[16] T. Cour, P. Srinivasan, and J. Shi, “Balanced graph matching,” *Advances in Neural Information Processing Systems*, vol. 19, p. 313, 2007.

[17] L. Torresani, V. Kolmogorov, and C. Rother, “Feature correspondence via graph matching: Models and global optimization,” in *European Conference on Computer Vision.* Springer, 2008, pp. 596–609.

[18] S. Gold and A. Rangarajan, “A graduated assignment algorithm for graph matching,” *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 377–388, 1996.

[19] B. J. van Wyk and M. A. van Wyk, “A pocs-based graph matching algorithm,” *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1526–1530, 2004.

[20] R. Zass and A. Shashua, “Probabilistic graph and hypergraph matching,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[21] M. Leordeanu, M. Hebert, and R. Sukthankar, “An integer projected fixed point method for graph matching and map inference,” in *Advances in Neural Information Processing Systems*, 2009, pp. 1114–1122.

[22] J. Lee, M. Cho, and K. M. Lee, "A graph matching algorithm using data-driven markov chain monte carlo sampling," in *Proceedings of the IEEE International Conference on Pattern Recognition*, 2010, pp. 2816–2819.

[23] F. Zhou and F. De la Torre, "Factorized graph matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 127–134.

[24] Y. Suh, M. Cho, and K. M. Lee, "Graph matching via sequential monte carlo," in *European Conference on Computer Vision.* Springer, 2012, pp. 624–637.

[25] F. Zhou and F. De la Torre, "Deformable graph matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2922–2929.

[26] M. Cho, J. Sun, O. Duchenne, and J. Ponce, "Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2083–2090.

[27] K. Adamczewski, Y. Suh, and K. Mu Lee, "Discrete tabu search for graph matching," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 109–117.

[28] L. Livi and A. Rizzi, "The graph matching problem," *Pattern Analysis and Applications*, vol. 16, no. 3, pp. 253–283, 2013.

[29] P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last 10 years," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 28, no. 01, 2014.

[30] A. Egozi, Y. Keller, and H. Guterman, "A probabilistic approach to spectral graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 18–27, 2013.

[31] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 695–703, 1988.

[32] M. Zaslavskiy, F. Bach, and J.-P. Vert, "A path following algorithm for the graph matching problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2227–2242, 2009.

[33] W. Brendel and S. Todorovic, "Learning spatiotemporal graphs of human activities," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 778–785.

[34] Z.-Y. Liu, H. Qiao, and L. Xu, "An extended path following algorithm for graph-matching problem," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1451–1456, 2012.

[35] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce, "A tensor-based algorithm for high-order graph matching," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 12, pp. 2383–2395, 2011.

[36] M. Chertok and Y. Keller, "Efficient high order matching," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2205–2215, 2010.

[37] J. Yan, C. Zhang, H. Zha, W. Liu, X. Yang, and S. M. Chu, "Discrete hypergraph matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1520–1528.

[38] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proceedings of the 11th international conference on World Wide Web.* ACM, 2002, pp. 517–526.

[39] J. Maciel and J. P. Costeira, "A global solution to sparse correspondence problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 187–199, 2003.

[40] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[41] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web." 1999.

[42] M. Gori, M. Maggini, and L. Sarti, "Exact and approximate graph matching using random walks," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 7, pp. 1100–1111, 2005.

[43] A. Robles-Kelly and E. R. Hancock, "String edit distance, random walks and graph matching," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 03, pp. 315–327, 2004.

[44] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[45] R. Diestel, *Graph theory.* Springer, 2000.

[46] A. N. Langville and C. D. Meyer, "Deeper inside pagerank," *Internet Mathematics*, vol. 1, no. 3, pp. 335–380, 2004.

[47] E. Seneta, *Non-negative matrices and Markov chains.* Springer Science & Business Media, 2006.

[48] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[49] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *The annals of mathematical statistics*, vol. 35, no. 2, pp. 876–879, 1964.

[50] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Advances in Neural Information Processing Systems*, 2006, pp. 1601–1608.

[51] P. A. Regalia and E. Kofidis, "The higher-order power method revisited: convergence proofs and effective initialization," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, 2000, pp. 2709–2712.

[52] Z. Tu and S.-C. Zhu, "Image segmentation by data-driven markov chain monte carlo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 657–673, 2002.

[53] F. Dellaert, S. M. Seitz, S. Thrun, and C. Thorpe, "Feature correspondence: A markov chain monte carlo approach," in *Advances in Neural Information Processing Systems*, 2000, pp. 852–858.

[54] L. Lin, X. Liu, and S.-C. Zhu, “Layered graph matching with composite cluster sampling,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1426–1442, 2010.

[55] J. S. Liu, *Monte Carlo strategies in scientific computing.* Springer Science & Business Media, 2008.

[56] M. Leordeanu, R. Sukthankar, and M. Hebert, “Unsupervised learning for graph matching,” *International Journal of Computer Vision*, vol. 96, no. 1, pp. 28–45, 2012.

[57] A. Albarelli, S. R. Bulo, A. Torsello, and M. Pelillo, “Matching as a non-cooperative game,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2009, pp. 1319–1326.

[58] J. Dattorro, *Convex optimization & Euclidean distance geometry.* Lulu. com, 2010.

[59] M. Cho, K. Alahari, and J. Ponce, “Learning graphs to match,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 25–32.

국문초록

그래프정합은 컴퓨터비전 및 패턴인식 분야의 근본적인 문제인 두 집합간의 대응관계를 도출하는 대표적인 기법이다. 많은 알고리즘들이 그래프의 꼭짓점으로 특징점을 표현하고 변으로 꼭짓점간의 관계를 나타내는 것으로 대응관계 문제를 그래프정합으로 다루었지만, 실제 발생하는 여러 가지 노이즈로 인하여 그래프정합은 도전적인 문제로 남아있다. 본 학위논문에서는 노이즈에 강인한 그래프정합 알고리즘들을 제안하고 컴퓨터비전에 있어서 효과적인 그래프정합 전략을 조사하도록 한다.

이를 위하여 첫째로, 시뮬레이션을 기반으로 한 강인한 두 그래프정합 알고리즘들을 제안한다. 하나의 알고리즘은 랜덤워크 시뮬레이션을 기반으로 하였으며, 다른 하나의 알고리즘은 마르코프체인몬테카를로를 기반으로 한다. 둘째로, 그래프정합을 위한 두 수식과 그래프정합 분야에서 분석이 제대로 이루어지지 않은 두 수식간의 상호호환 관계에 대하여 분석한다. 본 학위논문에서 제안하는 변환공식을 통하여 기존의 그래프정합 알고리즘들이 다른 형태의 수식을 기반으로 한 그래프정합 문제를 풀어낼 수 있음을 보인다. 마지막으로, 주장하는 내용을 고차그래프정합 문제로 확장하여, 각각 랜덤워크와 마르코프체인몬테카를로를 기반으로 하는 두 고차그래프정합 알고리즘들을 제안하고, 두 고차그래프정합 수식의 상호호완 관계에 대하여 분석하며, 기존의 고차그래프정합 알고리즘들이 다른 형태의 고차그래프정합 수식을 다룰 수 있도록 변환이 가능함을 보인다. 본 학위논문 각각의 장을 통하여 철저한 비교실험을 수행하였고, 이를 통하여 그래프정합 수식, 상호호환관계, 알고리즘들의 특성을 분석하였다. 합성그래프 실험과 실제 이미지 특징점 정합 실험을 통하여 다양하고 강도 높은 노이즈에 대한

특성을 분석하였다.

주요어: 그래프정합, 고차그래프정합, 그래프정합 수식화, 마르코프체인몬테카를로, 데이터기반, 랜덤워크

학번: 2008-20943