Toward Reliable Graph Matching: from Deterministic Optimization to Combinatorial Learning

by

Tianshu Yu

A Dissertation Presented in Partial Fulfillment
of the Requirement for the Degree
Doctor of Philosophy

Approved July 2021 by the
Graduate Supervisory Committee:

Baoxin Li, Chair
Yalin Wang
Yezhou Yang
Yingzhen Yang

ARIZONA STATE UNIVERSITY

December 2021

ABSTRACT

Graph matching is a fundamental but notoriously difficult problem due to its NP-hard nature, and serves as a cornerstone for a series of applications in machine learning and computer vision, such as image matching, dynamic routing, drug design, to name a few. Although there has been massive previous investigation on high-performance graph matching solvers, it still remains a challenging task to tackle the matching problem under real-world scenarios with severe graph uncertainty (e.g., noise, outlier, misleading or ambiguous link).

In this dissertation, a main focus is to investigate the essence and propose solutions to graph matching with higher reliability under such uncertainty. To this end, the proposed research was conducted taking into account three perspectives related to reliable graph matching: modeling, optimization and learning. For modeling, graph matching is extended from typical quadratic assignment problem to a more generic mathematical model by introducing a specific family of separable function, achieving higher capacity and reliability. In terms of optimization, a novel high gradient-efficient determinant-based regularization technique is proposed in this research, showing high robustness against outliers. Then learning paradigm for graph matching under intrinsic combinatorial characteristics is explored. First, a study is conducted on the way of filling the gap between discrete problem and its continuous approximation under a deep learning framework. Then this dissertation continues to investigate the necessity of more reliable latent topology of graphs for matching, and propose an effective and flexible framework to obtain it. Coherent findings in this dissertation include theoretical study and several novel algorithms, with rich experiments demonstrating the effectiveness.

To my wife Shuang Chen.

ACKNOWLEDGEMENTS

I would give my first and deepest thanks to my advisor, professor Baoxin Li. Baoxin not only has influenced me with his curiosity, persistence and passion for science, but also has taught me how I should be facing any obstacle in my life and much more beyond. Without his guidance, I will not be able to complete my Ph.D. study.

I appreciate professor Yalin Wang, professor Yezhou Yang and professor Yingzhen Yang for serving on my dissertation committee and providing many helpful suggestions. I hope we can have the opportunity to conduct research together in my future career.

Special thank is for my mentor and friend, professor Junchi Yan at Shanghai Jiao Tong University, whom I worked with since 2017 when I was interning at IBM research Shanghai. It was my pleasure to collaborate with Junchi for all these years. His experience also encouraged me to pursue my career at academia after my graduation.

I spent four wonder years at Arizona State University, working with great friends: Yikang Li, Kevin Pak Lun Ding, Yuzhen Ding, Yaoxin Zhuo, Riti Paul, Nupur Thakur, Sachin Chhabra, Jiuxu Chen, Zhiyuan Fang, Liang Mi and Yilin Wang, from whom I learned so much.

After so many years, I finally get to understand the unconditional love from my parents, for encouraging me when I was lost, for trusting me when I failed, and for pushing me when I was hesitating. They gave me so much support during my Ph.D. career and my life.

Most importantly, I give my thank to my wife, Shuang Chen, for giving me love, tolerance and care. I am so lucky to have you being with me, and to have our cats Meiqiu and Erdan in our life. You are my valor and strength every moment. You are my shining star in the sky.

My Ph.D. study was support in part by grants from ONR, which is greatly appreciated.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Background

While being a long standing and NP-hard combinatorial problem, graph matching (**GM**) has been persistently investigated over decades (Loiola *et al.*, 2007; Yan *et al.*, 2020). A graph can be defined as a collection of nodes and associated edges as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $i \in \mathcal{V}$ refers to node index i and $(i, j) \in \mathcal{E}$ corresponds to an edge linking node i and j. Let $n = |\mathcal{V}|$ be the number of nodes. In general, given source and target graphs, GM seeks to find node-to-node correspondence between them which maximizes the overall similarity measurement for both nodes and edges. Several modeling methods for GM have been proposed such as linear assignment (Swoboda *et al.*, 2017, 2019), semi-definite programming (Schellewald and Schnörr, 2005; Yu and Wang, 2016) or higher-order tensor expression (Duchenne *et al.*, 2011; Shi *et al.*, 2016). In this dissertation, we focus on a more general mathematical model of GM termed as Quadratic Assignment Problem (**QAP**) which can be formalized as [1] :

$$\max_{\mathbf{z}} \mathbf{z}^\top \mathbf{M} \mathbf{z} \qquad \text{s.t.} \quad \mathbf{Z} \in \{0,1\}^{n\times n}, \quad \mathbf{H}\mathbf{z} = \mathbf{1} \tag{1.1}$$

where the affinity matrix $\mathbf{M} \in \mathbb{R}_{+}^{n^2\times n^2}$ encodes node (diagonal elements) and edge (off-diagonal) affinities/similarities and $\mathbf{z}$ is the column-wise vectorization form of the permutation matrix $\mathbf{Z}$. $\mathbf{H}$ is a selection matrix ensuring each row and column of $\mathbf{Z}$

[1]Without loss of generality, we discuss graph matching under the setting of equal number of nodes without outliers. The unequal case can be readily handled by introducing extra constraints or dummy nodes. Bipartite matching and graph isomorphism are subsets of this quadratic formulation (Loiola *et al.*, 2007).

summing to 1. $\mathbf{1}$ is a column vector filled with 1. A large body of arts in the category of traditional deterministic optimization framework have been proposed attempting to tackle GM problem (Eq. (1.1)) under QAP relaxation (Rangarajan *et al.*, 1999a; Nowak *et al.*, 2018; Yu *et al.*, 2018b). More recently, investigation following QAP has extended from deterministic optimization to deep-learning-based methods, which significantly enhance the capacity and performance on several real-world benchmarks (Wang *et al.*, 2019a; Zhang and Lee, 2019; Yu *et al.*, 2020a; Fey *et al.*, 2020; Rolínek *et al.*, 2020).

Although GM problem, especially its QAP counterpart, is more like a theoretical practice, it has a wide range of applications under real scenarios. A series of practical applications can benefit from the research on GM, such as image matching (Zhou *et al.*, 2015b), dynamic routing (Hsu *et al.*, 2020), protein matching (Krissinel and Henrick, 2004), social network mining (Chiasserini *et al.*, 2018) and metric for comparing graphs (Bai *et al.*, 2019). Here we outline two typical examples where GM serves as the fundamental cornerstone:

- **(Multi-)Image Matching.** This is an essential task and building block of several computer vision pipelines, where establishing keypoint matching between two or more images is vital for the subsequent task (Zhou *et al.*, 2015b). For example, in the application of stereo, the most difficult and time-consuming step is to find the dense point matching between images, then the 3D geometry of objects in the scene can be estimated from matching of the pixels. This can be viewed as a special case of GM, since typical stereo matching algorithms also seek to maximize an overall similarity on nodes under some smoothness constraints on edges, analogously to the objective in GM. While GM covers the model of stereo, it is capable of handling more complex geometry since it incorporates higher level similarity. However, one major problem is that traditional

GM methods may suffer from severe noise and mis-detected keypoints due to complex 3D real-world environments. Besides, a grid-like structure indicates isotropic local interaction for each pixel and its neighbors, which may cause misleading to a GM solver.

- **Gene Regulatory Network Inference.** It is instrumental baseline to explore genetic mechanisms which potentially drive diverse diseases, including cancer (Weighill *et al.*, 2020). The essential part of this task can be reduced to QAP under a subgraph matching context [2] . While some efficient solvers to this problem regard it as a bipartite matching problem, this modeling does not accommodate clique-level (higher-level) similarities, thus can produce ambiguous matching in some cases (Alberich *et al.*, 2019). Thus we can infer that encoding clique-level similarities can further help to improve the performance. However, an essential issue in this task is that the length of a piece of gene can easily exceeds mega with a large amount of gene segments that cannot be aligned intrinsically, which greatly hinders the applicability of traditional QAP solvers since they are very likely to be interfered by mass outliers. Since this task is related to key research on improving human lives and can bring about much economic benefit, we think it is demanding to devise more efficient and reliable solvers for such problems.

Significance: A series of practical applications can in consequence benefit from this research, such as image matching (Zhou *et al.*, 2015b), dynamic routing (Hsu *et al.*, 2020), protein matching (Krissinel and Henrick, 2004), social network mining (Chiasserini *et al.*, 2018) and metric for comparing graphs (Bai *et al.*, 2019). In either of the applications, GM can serve as a building block or even the cornerstone from

[2]Subgraph matching implies a small graph is similar to a subgraph of a larger one. It sometimes called inexact graph matching.

the mathematical modeling perspective. Therefore, this research is anticipated to boost or at least provide insight to relevant topics. Aside from application benefits, investigation on theoretical aspects of combinatorial GM may also help to further understand the intrinsic structure of related combinatorial problems.

1.2 Overview

Having discussed the relevant real-world problems and identified some key issues, this dissertation will focus on the topic of *learning to solve graph matching with higher reliability*. Concretely, we put our focus on three essential aspects of GM related to the reliability issue: modeling, optimization and learning:

- **Modeling**. As a common modeling of GM, QAP has been investigated for years. However, it is clear that the model capacity of QAP is limited since QAP only consists of one linear part and a quadratic part. This fact can potentially hinder the expressive power of GM and sometimes lead to difficulties in optimization (Yan *et al.*, 2020). The first aspect in the dissertation is about finding a more generic and reliable modeling method as studied in Chapter 3, which can increase the model capacity as well as ease in optimization, compared to QAP.

- **Optimization**. Since GM is intrinsically a discrete combinatorial problem, a series of optimization methods for GM heavily rely on regularization or function deformation techniques to reach discrete solutions. However, there has no previous work on investigating the basis of validity or effectiveness of such techniques. In this part, we study the functional behavior of such regularizers and propose a novel method to improve the reliability under massive outliers as in Chapter 4.

- **Learning**. It has been proved that deep learning paradigm can greatly help to improve the matching accuracy compared to traditional solvers under severe degradation (Zanfir and Sminchisescu, 2018; Wang *et al.*, 2019a; Zhang and Lee, 2019). Based on this fact, we further investigate more effective graph neural mechanism as well as the gap between discrete GM problem and its approximation in Chapter 5. Besides, we also study the way of explore more effective latent topology for graph matching under a learning paradigm in Chapter 6.

We will briefly review recent related works for graph matching in Chapter 2. In what follows (from Chapter 3-6), we will summarize our progress towards each perspective as stated above. In Chapter 3, we summarize our effort to generalize graph matching problem from QAP to a more broader function family, denoted as Separable Function, achieving higher model capacity and graph degradation tolerance. In Chapter 4, we introduce a specific regularization technique using matrix determinant, with stronger performance on outlier. We further develop a optimization method derived from geometric inequality. We outline our deep learning based graph matching solver which seeks to fill the gap between disrete problem and its approximation in Chapter 5. A way of learning more effective topology to avoid useless and misleading edges is investigated in Chapter 6, building upon the hypothesis on the existence of latent topology that is favorable for the GM solvers. We finally summarize our work and point some future research directions in Chapter 7.

Chapter 2

RELATED WORK

2.1 Prior Art

Mathematically, graph matching has been treated as a pure optimization problem (Loiola *et al.*, 2007) where the objective to find node-to-node level correspondence between two graphs, in which the topological structure as well as the inter-graph similarities are predefined. As such, traditional solvers to graph matching do not incorporate any learning paradigm. While this problem is NP-hard due to its discrete nature, some methods attempt to incorporate discrete combinatorial heuristics for solving it (Zhao *et al.*, 2014; Adamczewski *et al.*, 2015). A broader series of works alter to introduce continuous relaxation to make the optimization tractable, including: 1) Quadratic Assignment Problem (QAP) (Gold and Rangarajan, 1996; Cho *et al.*, 2010; Yu *et al.*, 2018b); 2) Semi-Definite Programming (SDP) relaxation (Schellewald and Schnörr, 2003; Torr, 2003); 3) Linearization (Swoboda *et al.*, 2017, 2019). While both SDP and linearization relaxations scale up the number of variables to be optimized, most existing traditional graph matching solvers are devised based on QAP. Solvers based on such relaxed problem generally fall into the categories of iterative update (Cho *et al.*, 2010; Jiang *et al.*, 2017a) or numerical continuation (Zhou and Torre, 2016; Yu *et al.*, 2018b). On one hand, iterative update generally assumes that the affinity/similarity matrix between two input graphs has some nice properties, which enable more efficient calculation or update for each round of iteration; on the other hand, numerical continuation does not make any assumption on affinity but seeks to find the optima purely using gradient. In either of these two categories, the solvers are developed under two key assumptions: 1) Affinity matrix is pre-computed with some non-negative metrics, e.g. Gaussian kernel, L^2-distance or Manhattan distance; 2) Graph topology is pre-defined as input either in dense (Schellewald and Schnörr, 2005) or sparse (Zhou and Torre, 2016) fashion.

It is a natural extension from deterministic optimization to learning paradigm, since the model capacity of QAP is somewhat limited while more high-quality human labelled datasets become available. Early non-deep learning-based methods seek to learn effective metric (e.g. weighted Euclid distance) for node and edge features or affinity kernel (e.g. Gaussian kernel) in a parametric fashion (Caetano *et al.*, 2009; Cho *et al.*, 2013). These attempts, unfortunately, cannot be readily integrated into deep networks mostly due to their undifferentiable nature. Recent deep graph matching methods have shown how to extract more dedicated feature representation. In 2018, the seminal work (Zanfir and Sminchisescu, 2018) adopts VGG16 (Simonyan and Zisserman, 2014) as the backbone for feature extraction on images. Other efforts have been witnessed in developing more advanced pipelines, where graph embedding (Wang *et al.*, 2019a; Yu *et al.*, 2020a; Fey *et al.*, 2020) and geometric learning (Zhang and Lee, 2019; Fey *et al.*, 2020) are involved. Rolínek *et al.* (2020) studies the way of incorporating traditional non-differentiable combinatorial solvers, by introducing a differentiatiable blackbox GM solver (Pogancic *et al.*, 2020). Recent works in tackling combinatorial problem with deep learning (Huang *et al.*, 2019; Kool and Welling, 2018) also inspire developing combinatorial deep solvers, for GM problems formulated by both Koopmans-Beckmann's QAP (Nowak *et al.*, 2018; Wang *et al.*, 2019a) and Lawler's QAP (Wang *et al.*, 2019b). Specifically, Wang *et al.* (2019a) devise a permutation loss for supervised learning, with an improvement in Yu *et al.* (2020a) via Hungarian attention. Wang *et al.* (2019b) solve the most general Lawler's QAP with graph embedding technique. We note, to our best knowledge, there is no previous work explicitly seek to systematically address the issues of reliability. Most existing algorithms are designed for up to hundreds of nodes (Wang *et al.*, 2019a; Zhang and Lee, 2019; Yu *et al.*, 2020a; Fey *et al.*, 2020; Rolínek *et al.*, 2020), under naive Gaussian noise or less outliers.

Since we will handle the case where graph topology is generated or sampled, we also briefly review related arts in the context of graph generative model. Early generative models for graph can date back to 1950s (Erdos and Renyi, 1959), in which edges are generated with fixed probability. Recently, Kipf and Welling (2016) presented a graph generative model by re-parameterizing the edge probability from Gaussian noise. Johnson (2017) proposed to generate graph in an incremental fashion, and in each iteration a portion of the graph is produced. Gómez-Bombarelli *et al.* (2018) utilized recurrent neural network to generate graph from a sequence of molecule representation. Adversarial graph generation is considered in Pan *et al.* (2018); Wang *et al.* (2018a); Bojchevski *et al.* (2018). Specifically, Wang *et al.* (2018a); Bojchevski *et al.* (2018) seeked to unify graph generative model and generative adversarial networks. In parallel, reinforcement learning has been adopted to generate discrete graphs (De Cao and Kipf, 2018). In either of the aforementioned methods, the generative model is dedicated to generate graph as a whole sufficing some observable data distribution. For generating partial graph, there is barely any previous research. Though Ibarrola *et al.* (2020) is an initial attempt towards partial generation, it can only handle partial categorical conditioning vector which can only be applied to limited scenarios.

2.2 My Related Work

This dissertation is based on our previous investigation of graph matching problem from both deterministic and learning perspectives. On tradition modeling and optimization of graph matching, we extended traditional Quadratic Assignment Problem (QAP) modeling to a much broader domain by developing Separable Functions (Yu *et al.*, 2018b). This extension can significantly improve the model capacity with theoretical guarantee. In Yu *et al.* (2020b), we discussed the effectiveness of several regularization techniques and proposed a novel one using determinant. It achieves much better gradient-effectiveness compared to previous regularizers such as entropy and L^2-norm. Noticing the issues of overfitting and approximation gap to discrete problem, we proposed Channel-Independent Embedding and Hungarian Attention in Yu *et al.* (2020a) to respectively address them. This deep learning based GM algorithm achieved state-of-the-art performance on several public benchmarks. Taking into account the uncertainty of heuristically created topology, we further propose a way of performing graph matching on the latent topology in a learning-based fashion (Yu *et al.*, 2021). This framework can be readily extended to a family of graph matching solvers and ensures the convergence under an Expectation-Maximization interpretation.

Aside from pairwise GM, we proposed the first work to conduct incremental multi-graph matching by grouping set of graphs under diversity with randomness (Yu *et al.*, 2018a). This method was an attempt to reduce hyper-topology from dense to sparse, aligning our plan on investigating sparse hyper-topology to handle scalability issue. In Yu *et al.* (2018c), we defined a novel problem to jointly perform graph cuts and graph matching, with a novel optimization method.

Chapter 3

GENERALIZED MODELING OF GRAPH MATCHING

3.1 Problem Statement

Graph matching seeks the solution to the quadratic assignment problem (QAP):

$$\max_{\mathbf{Z}} \mathrm{vec}(\mathbf{Z})^\top \mathbf{A} \mathrm{vec}(\mathbf{Z}) \tag{3.1}$$

where $\mathrm{vec}(\mathbf{Z}) \in \{0,1\}^{n^2}$ is the column-wise vectorized version of the binary (partial) assignment matrix $\mathbf{Z} \in \{0,1\}^{n\times n}$ and the so-called affinity matrix $\mathbf{A} \in \mathbb{R}^{n^2\times n^2}$ in the real domain consists of the affinity score measuring how one edge in one graph is similar to another from the other graph. Traditionally, the common practice is relaxing $\mathrm{vec}(\mathbf{Z})$ into the continuous real domain $\mathrm{vec}(\mathbf{Z}) \in \mathbb{R}^{n^2}$ (Gold and Rangarajan, 1996; Leordeanu and Hebert, 2005; Cho *et al.*, 2010).

In this part, we show that a large family of functions, defined as **Separable Functions**, can asymptotically approximate the discrete matching problem by varying the approximation controlling parameters. With this function family, there exist infinite modelings of graph matching problem, thereby providing the feasibility of adapting different practical problems with different models. This provides a new perspective of considering graph matching. We also give analysis on the conditions based on which these approximations have good properties. Novel solvers on instances of Separable Functions are proposed based on the path-following and multiplicative techniques respectively.

Notations We use bold lower-case $\mathbf{x}$ and upper-case $\mathbf{A}$ to represent vector and matrix, respectively. Function $\mathrm{vec}(\cdot)$ transforms a matrix to its column-wise vectorized replica. Conversely, function $\mathrm{mat}(\cdot)$ transfers a vector back to its matrix form.

Denote $\mathbb{R}_+$, $\mathbb{S}$ as non-negative real numbers and symmetric matrices respectively. Function $\mathbf{K} = \text{diag}(\mathbf{k})$ transforms a vector $\mathbf{k}$ into a diagonal matrix $\mathbf{K}$ such that $\mathbf{K}_{ij} = \mathbf{k}_i$ if $i = j$, and $\mathbf{K}_{ij} = 0$ otherwise.

3.2 Generalizing QAP for Graph Matching

We re-visit the graph matching problem in this section. We propose an equivalent model to the discrete one over continuous domain $[0, 1]$, provided the relaxation gap is 0. This gives rise to the possibility to relax graph matching with much tighter ways. Mathematically, graph matching can be formulated as the following quadratic assignment problem which is also called Lawler's QAP [1] (Lawler, 1963):

$$\begin{aligned} &\max_{\mathbf{Z}} \text{vec}(\mathbf{Z})^\top \mathbf{A} \text{vec}(\mathbf{Z}) \\ &\text{s.t. } \mathbf{Z1} = \mathbf{1}, \mathbf{Z}^\top \mathbf{1} = \mathbf{1}, \mathbf{x}_{ia} \in \{0, 1\} \end{aligned} \tag{3.2}$$

where $\mathbf{A} \in \mathbb{R}_+^{n^2 \times n^2}$ is a non-negative affinity matrix, which encodes node similarities on diagonal elements and edge similarities on the rest. Note $\mathbf{x}_{ia}$ denotes the element of $\mathbf{Z}$ indexed by row i and column a indicating the matching status of node i to node a from the other graph. If we break down problem (3.2) into element-wise product, it becomes:

$$\begin{aligned} &\max_{\mathbf{x}} \sum_{i,j,a,b} \mathbf{A}_{ij:ab} \mathbf{z}_{ia} \mathbf{z}_{jb} \\ &\text{s.t. } \mathbf{Hz} = \mathbf{1}, \mathbf{z} \in \{0, 1\}^{n^2} \end{aligned} \tag{3.3}$$

where $\mathbf{A}_{ij:ab}$ corresponds to the edge similarity between edge $(i, j) \in \mathcal{G}_1$ and $(a, b) \in \mathcal{G}_2$. Here $\mathbf{H} \in \{0, 1\}^{2n \times n^2}$ is a selection matrix over the elements of $\mathbf{x}$ sufficing assignment constraints according to (3.2).

[1]Here the number of nodes in two graphs are assumed the same. In case $m \neq n$ one can add dummy nodes as a standard technique as in literature Cho *et al.* (2010); Zhou and Torre (2012).

In particular, we relax $\mathbf{z}$ into continuous domain and let $f_{\text{prod}}(\mathbf{z}_{ia}, \mathbf{z}_{jb}) = \mathbf{z}_{ia}\mathbf{z}_{jb}$:

$$\begin{aligned} &\max_{\mathbf{z}} \sum_{i,j,a,b} \mathbf{A}_{ij:ab} f_{\text{prod}}(\mathbf{z}_{ia}, \mathbf{z}_{jb}) \\ &\text{s.t. } \mathbf{Hz} = \mathbf{1}, \mathbf{z} \in [0,1]^{n^2} \end{aligned} \tag{3.4}$$

We generalize problem (3.4) by replacing f_{prod} with f_δ:

$$\begin{aligned} &\max_{\mathbf{z}} \sum_{i,j,a,b} \mathbf{A}_{ij:ab} f_{\delta}(\mathbf{z}_{ia}, \mathbf{z}_{jb}) \\ &\text{s.t. } \mathbf{Hz} = \mathbf{1}, \mathbf{z} \in [0,1]^{n^2} \end{aligned} \tag{3.5}$$

where f_δ is a 2D quasi-delta function in the continuous domain ($f_\delta(x, y) = 1$ if $x = 1$ and $y = 1$, and $f_\delta = 0$ otherwise). We have the following theorem that establishes the connection between (3.3) and (3.5):

Theorem 1. *The optimal objective p^* to problem (3.3) is equal to the optimal objective p^*_δ to problem (3.5).*

Proof. As problem (3.5) is the relaxed version of problem (3.3), we must have $p^*_\delta \geq p^*$.

Suppose $\mathbf{z}^* = \text{vec}(\mathbf{Z}^*)$ is the optimal solution to problem (3.5). We recursively implement the following procedure until there is no 1 in $\mathbf{z}^*$. If $\mathbf{z}^*_{ia} = 1$, according to the doubly stochastic property, the ith row and ath column elements other than (i, a) element would all be 0. We then remove all the elements in $\mathbf{A}$ corresponding to node i in $\mathcal{G}_1$ and node a in $\mathcal{G}_2$. Finally we can reach a subset of $\mathbf{z}$ and $\mathbf{A}$ such that each element in $\mathbf{z}$ is in the range $[0, 1)$. Figure 3.1 schematically shows how this procedure works from left to right.

However, due to the definition of function f_δ, the affinity score over the remaining nodes becomes 0. As $\mathbf{A}$ is non-negative, any 1 value assignment would result in affinity score no less than 0. Denote the objective value of such assignment p^{assign}, then we have $p^*_\delta \leq p^{\text{assign}}$. On the other hand, p^{assign} is discrete, then we must have $p^{\text{assign}} \leq p^*$.

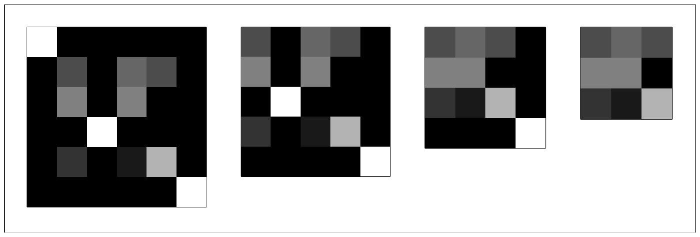


Figure 3.1: Procedure to Remove 1 Elements. Here the Manipulation on a 6×6 Matrix Is Demonstrated Schematically. From Left to Right, We Remove a 1 Element and Corresponding Column and Row in Each Step. The Rightmost Matrix Is $\mathrm{mat}(Z^{\dagger})$ with All Elements in $[0, 1)$.

In summary, we have $p^* = p^*_{\delta}$. QED. $\square$

Remark. Based on Theorem 1, one can devise a sampling procedure to find the optimal solution to problem (3.3) from the solution to problem (3.5): Given optimal $\mathbf{z}^*_{\delta}$ to problem (3.5), if all the elements are in the set $\{0, 1\}$, then $\mathbf{z}^*_{\delta}$ is automatically optimal to problem (3.3). If not, we first remove all 1 elements and corresponding columns and rows, yielding a subvector (submatrix) $\mathbf{z}^{\dagger}$ with all elements in range $[0, 1)$. Then any sampling subject to one-to-one constraint on $\mathbf{z}^{\dagger}$, together with the removed discrete values, forms the optimal solution to problem (3.3).

For the time being, a discrete assignment problem (3.3) is relaxed into (3.5) with continuous feasible domain. However, function f_{δ} is not continuous as there is a jump at value $(1, 1)$, ending up with much difficulty to solve (recall (3.5) is equivalent to (3.3)). In the next section, we will show some approximate techniques to tackle problem (3.5).

3.3 Separable Functions

3.3.1 Separable Approximation Function Family

It is important to find an appropriate approximate function for f_δ, otherwise it may lead to intractable models to solve. To avoid high computational cost, we narrow our focus on a specific family of functions, called Separable Functions.

Definition 1. *A function $f_\theta(x, y)$ is called Separable Function (**SF**)*[2] *if it satisfies the following properties:*

1. $f_\theta(x, y) = h_\theta(x) \times h_\theta(y)$ where h_θ is defined on $[0, 1]$.

2. $h_\theta(0) = 0$ and $h_\theta(1) = 1$. $h_\theta \in \mathbb{C}^1$.

3. h_θ is non-decreasing and $\lim_{\theta\to 0} h_\theta(x) - h_\delta(x) = 0$ for any $x \in [0, 1]$, where h_δ is defined on $[0, 1]$, $h_\delta(x) = 1$ if $x = 1$ and $h_\delta(x) = 0$ otherwise.

We also call such a function h_θ **univariate SF**, where θ is a controlling parameter. Being seemingly a simple formulation, SF has three fine properties for computation.

Firstly, SF shows similar behavior as a probabilistic distribution on two independent variables. That is, if two nodes are impossible to match, then any pair of edges containing the two nodes will never match neither. Mathematically, assuming the matching score of node pair $\langle i, a\rangle$ is $h_\theta(\mathbf{z}_{ia})$, we have $f_\theta(\mathbf{z}_{ia}, \mathbf{z}_{jb}) = 0$ for any $\langle j, b\rangle$ if $h_\theta(\mathbf{z}_{ia}) = 0$.

Secondly, the definition of SF eases gradient computing. For a given SF $f_\theta(x, y) = h_\theta(x)h_\theta(y)$, the approximate version of problem (3.5) can be expressed in matrix form as:

$$\begin{aligned} &\max_{\mathbf{z}} \mathbf{h}_\theta^\top \mathbf{A}\mathbf{h}_\theta \\ &\text{s.t. } \mathbf{H}\mathbf{z} = \mathbf{1}, \mathbf{z} \in [0, 1]^{n^2} \end{aligned} \tag{3.6}$$

[2]In fact separable function has its traditional meanings in mathematics, we re-define it in the graph matching context.

where $\mathbf{h}_\theta = [h_\theta(\mathbf{z}_1), ..., h_\theta(\mathbf{z}_{n^2})]^\top$. The gradient of objective (3.6) with respect to $\mathbf{z}$ is $\nabla\mathbf{z} = 2\mathbf{GAh}$, where $\mathbf{G}$ is a diagonal matrix with the ith element $\partial h_\theta(\mathbf{z}_i)/\partial\mathbf{z}_i$.

The third advantage of SF is that we can construct new approximation functions via reweighted summation and multiplication of existing ones, e.g. if h_1 and h_2 are two univariate SFs, it can be trivially verified that $\alpha h_1 + (1-\alpha)h_2$ for $0 \leq \alpha \leq 1$ and $h_1 \times h_2$ are also univariate SFs.

If we keep the constraints on $\mathbf{z}$ intact as in problem (3.6), and let

$$p_\theta^* = \max_{\mathbf{z}} \mathbf{h}_\theta(\mathbf{z})^\top \mathbf{A}\mathbf{h}_\theta(\mathbf{z}) \tag{3.7}$$

where $\mathbf{h}_\theta(\mathbf{z}) = [h_\theta(\mathbf{z}_1), ..., h_\theta(\mathbf{x_n})]^\top$, we have the following theorem:

Theorem 2. $\lim_{\theta\to 0} p_\theta^* = p_\delta^*$

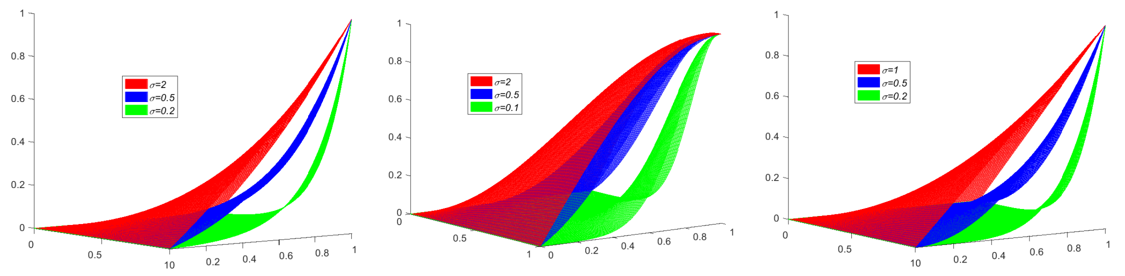
Proof. First we define two sets: $\mathcal{C}_1 = \{\mathbf{z}|\mathbf{Hz} = \mathbf{1}, \mathbf{z} \in [0,1]^{n^2}\}$, $\mathcal{C}_2 = \{\mathbf{z}|\mathbf{z} \in [0,1]^{n^2}\}$. It's easy to observe that $|p_\theta^* - p_\delta^*| \leq p_1$, where $p_1 = \arg\max_{\mathbf{z}} |\mathbf{h}_\theta^\top \mathbf{A}\mathbf{h}_\theta - \mathbf{h}_\delta^\top \mathbf{A}\mathbf{h}_\delta|$ subject to $\mathcal{C}_1$. This observation is true because the gap between two separable optimal objectives must be no larger than the maximal gap between the objectives.

We further define $p_2 = \arg\max_{\mathbf{x}} |\mathbf{h}_\theta^\top \mathbf{A}\mathbf{h}_\theta - \mathbf{h}_\delta^\top \mathbf{A}\mathbf{h}_\delta|$ subject to $\mathcal{C}_2$. As $\mathcal{C}_1 \subset \mathcal{C}_2$, we must have $p_1 \leq p_2$. By rewriting the objective corresponding to p_2 in the following way:

$$\begin{aligned}&\left|\sum_{i,j} \mathbf{A}_{ij} h_\theta(\mathbf{z}_i) h_\theta(\mathbf{z}_j) - \sum_{i,j} \mathbf{A}_{ij} h_\delta(\mathbf{z}_i) h_\delta(\mathbf{z}_j)\right| \\ =& \left|\sum_{i,j} \mathbf{A}_{ij} \left[((h_\theta(\mathbf{z}_i) - h_\delta(\mathbf{z}_i))h_\theta(\mathbf{z}_j) + (h_\theta(\mathbf{z}_j) - h_\delta(\mathbf{z}_j))h_\delta(\mathbf{z}_i)\right]\right|\end{aligned}$$

Note $\mathbf{A}$, h_θ and h_δ are all bounded. Additionally, $h_\theta(\mathbf{z}_i) \to h_\delta(\mathbf{z}_i)$ and $h_\theta(\mathbf{z}_j) \to h_\delta(\mathbf{z}_j)$ when $\theta \to 0$ by the third property. Thus $|p_\theta^* - p_\delta^*| \leq p_1 \leq p_2 \to 0$. QED. □

The above theorem guarantees that, if we approximate the quasi-delta function by



(a) h_{Lap} (b) h_{Gauss} (c) h_{Poly}

Figure 3.2: Three Examples of Approximations (Laplacian, Gaussian, Polynomial) to Function f_δ with Varying θ. The Closer for $\theta \to 0$ (from Red to Green), the Better Approximation to f_δ.

letting $\theta \to 0$, problem (3.5) can also be approximated asymptotically. As $h_\theta \in \mathbb{C}^1$, gradient-base algorithms can be applied to such approximations.

3.3.2 Approximations to Function f_δ

Though we have proved that using f_δ can derive an equivalent problem i.e. (3.5), finding its optimal solution is still notoriously difficult. Instead of solving (3.5) directly, based on the analysis in Sec. 3.3.1, we introduce approximation functions to f_δ. To simplify the expression, we only present the univariate SF h, and the SF f can be obtained using definition (1). It is trivial to show that the SFs derived from the following functions approximate f_δ when $\theta \to 0_+$ under the properties in definition (1):

$$h_{\text{Lap}}(x) = \frac{1}{m}\left\{\exp\left(\frac{x-1}{\theta}\right) - d\right\} \tag{3.8a}$$

$$h_{\text{Gauss}}(x) = \frac{1}{m}\left\{\exp\left(-\frac{(x-1)^2}{\theta}\right) - d\right\} \tag{3.8b}$$

$$h_{\text{Poly}}(x) = x^{\frac{1}{\theta}} \tag{3.8c}$$

where $d = \exp(-\frac{1}{\theta})$ and $m = 1 - d$. The usage of m and d is to normalize the SFs to satisfy the second property. Figure 3.2 shows some examples of such functions with varying θ values. Note that traditional quadratic graph matching model in fact is a special case of our model, which seeks to optimize a model where the SF is derived from h_{Poly} and $\theta = 1$. Specifically, for the univariate SFs (3.8a) and (3.8c), we also have the following proposition.

Proposition 1. *For univariate SF h_{Lap}, h_{Poly}, suppose p_1^* and p_2^* are the optimal objectives for (3.6) with θ_1 and θ_2, respectively. Then we have $p_1^* \geq p_2^*$ if $0 < \theta_2 < \theta_1$.*

Proof. This can be easily proved by showing $h_{\text{Lap}}(x; \theta_2) < h_{\text{Lap}}(x; \theta_1)$ and $h_{\text{Poly}}(x; \theta_2) < h_{\text{Poly}}(x; \theta_1)$ when $\theta_2 < \theta_1$. QED. □

Together with Theorem 2, this claim means that, given univariate SF h_{Lap} or h_{Poly}, the optimal objective of (3.6) will converge as $\theta \to 0_+$ monotonically.

3.3.3 Convexity/Concavity Analysis

Section 3.3.1 and 3.3.2 show that original problem (3.5) can be asymptotically approximated using SFs as $\theta \to 0$. In this section, we analyze the properties of convexity/concavity under such approximations. We believe this effort is worthwhile as one can employ techniques e.g. self-amplification (Rangarajan *et al.*, 1999b), to convert non-convex/non-concave problems into convex/concave ones with the beneficial properties of convexity/concavity. We first show the equivalence of problem (3.4) and (3.6) under global convexity.

Theorem 3. *Assume that affinity $\mathbf{A}$ is positive definite. If the univariate SF $h_\theta(x) \leq x$ on $[0, 1]$, then the global maxima of problem (3.3), which is discrete, must also be the global maxima of problem (3.6).*

Proof. As shown in Yuille and Kosowsky (1994), whenever affinity $\mathbf{A}$ is positive definite, the global maximum of problem (3.4) is a permutation. In this case, the optimum to (3.4) is also optimum to (3.3). Denote $\mathbf{y}^*$ the optimal permutation to (3.4). As $\mathbf{y}^*$ is doubly stochastic, it must also satisfy the same constraints in problem (3.6). Let p_1 be the objective of problem (3.6) w.r.t. $\mathbf{y}^*$ – Note p_1 is the optimal objective of problem (3.4). Assume there exists an optima $\mathbf{z}_\theta^* \neq \mathbf{y}^*$ to problem (3.6) with corresponding objective p_2. As p_2 is optimal, we have $p_2 \geq p_1$. Let $\mathbf{y}_\theta = \mathbf{h}_\theta(\mathbf{z}_\theta^*)$. As $h_\theta(z) \leq x$, we must have $\mathbf{z}_\theta^* \geq \mathbf{y}_\theta \geq \mathbf{0}$. Denote p_3 the objective score of (3.4) by substituting $\mathbf{z}_\theta^*$. Since $\mathbf{A}$ is non-negative, $\mathbf{z}_\theta^* \geq \mathbf{y}_\theta$ and $\mathbf{z}_\theta^*, \mathbf{y}_\theta \geq \mathbf{0}$, we have $p_3 \geq p_2$. In summary, $p_3 \geq p_1$. However, p_1 is the global optimal objective of (3.4). Thus the inequality leads to contradiction. The equality exists only when the global optimum of (3.6) is $\mathbf{y}^*$. QED. □

The above theorem builds up a link from problem (3.3) to problem (3.6) when $\mathbf{A}$ is positive definite. In this case, we first conclude that the optimum to problem (3.4) is discrete, hence also optimal to (3.3). Then as long as $h_\theta(x) < x$ on $[0, 1]$ and h_θ satisfies the second property in Definition 1, this solution is also optimal to problem (3.6). In this case the optimal objective gap of these three problems becomes 0. We give the following proposition showing under mild conditions, the generalized problem (3.6) is convexity/concavity-preserving.

Proposition 2. *Assume affinity maxtrix* $\mathbf{A}$ *is positive/negative semi-definite, then as long as the univariate SF* h_θ *is convex, the objective of (3.6) is convex/concave.*

Proof. Consider problem (3.6), we prove this proposition by checking the property of the Hessian with respect to $\mathbf{z}$. As we have obtained the gradient $2\mathbf{GAh}_\theta$ of the objective in (3.6) with respect to $\mathbf{z}$, we calculate the Hessian by taking the derivative once again. After some mathematical manipulations, we have $\nabla^2 \mathbf{z} = 2\mathbf{AK}$, where

$$\mathbf{K} = \operatorname{diag}\left(\left[\left(\frac{\partial h_\theta}{\partial \mathbf{z}_1}\right)^2 + h_\theta(\mathbf{z}_1)\frac{\partial^2 h_\theta}{\partial \mathbf{z}_1^2}, \cdots, \left(\frac{\partial h_\theta}{\partial \mathbf{z}_{n^2}}\right)^2 + h_\theta(\mathbf{z}_{n^2})\frac{\partial^2 h_\theta}{\partial \mathbf{z}_{n^2}^2}\right]^\top\right) \tag{3.9}$$

It is easy to show that $(\partial h_\theta / \partial \mathbf{z}_i)^2$ and $h_\theta(\mathbf{z}_i)$ are non-negative according to Definition 1. As h_θ is convex, its second order derivative must also be non-negative. Matrix $\mathbf{K}$ is positive semi-definite. Thus the convexity/concavity of $\mathbf{A}$ is preserved after multiplying $\mathbf{K}$. QED. $\square$

Any matrix $\mathbf{A}$ can be transformed to positive definite by adding up a diagonal matrix $\lambda\mathbf{I}$. The lower bound of λ is $\lambda \geq |\lambda^\dagger|$, where $\lambda^\dagger$ is the smallest eigenvalue of $\mathbf{A}$ below 0. We define $\mathbf{A}^\dagger = \mathbf{A} + \lambda\mathbf{I}$.

Proposition 3. *Assume affinity matrix $\mathbf{A}$ is positive definite and univariate SF h_θ is convex. The optimal value to the following problem is:*

$$E_{conv} = \max_{\mathbf{z}} \mathbf{h}_\theta^\top \mathbf{A}^\dagger \mathbf{h}_\theta \tag{3.10}$$

Then there exists a permutation $\mathbf{z}^*$, s.t. $E_{conv} - E(\mathbf{z}^*) \leq n\lambda$ where $E(\mathbf{z}^*)$ is the objective value w.r.t. problem (3.6).

Proof. First for any convex univariate SF h_θ in range $[0, 1]$, we have $h_\theta(z) \leq z$. Under the assumption in the theorem, given $\hat{\mathbf{z}}$ the optima to problem (3.6), we can obtain an optimal discrete $\mathbf{y}$ according to the sampling procedure in Theorem 1. The optimal objective of (3.6) can be written as:

$$E_{\text{conv}}(\mathbf{y}) = \sum_{i \neq j, a \neq b} \mathbf{A}_{ij:ab} h_\theta(\mathbf{y}_{ia}) h_\theta(\mathbf{y}_{jb}) + \sum_{i,a} \left(\mathbf{A}_{ii:aa} + \lambda\right) h_\theta^2(\mathbf{y}_{ia}) \tag{3.11}$$

Besides, by substituting $\mathbf{y}$ into problem (3.6) we obtain:

$$E(\mathbf{y}) = \sum_{i,j,a,b} \mathbf{A}_{ij:ab} h_\theta(\mathbf{y}_{ia}) h_\theta(\mathbf{y}_{jb}) \tag{3.12}$$

By subtracting Equation (3.12) from (3.11) we have:

$$E_{\text{conv}}(\mathbf{y}) - E(\mathbf{y}) = \lambda \sum_{i,a} h_\theta^2(\mathbf{y}_{ia}) \tag{3.13}$$

As $\text{mat}(\mathbf{y}) \in \{0,1\}^{n^2}$ is a permutation hence $h_\theta(\mathbf{y}_{ia}) = \mathbf{y}_{ia}$, we have $\lambda \sum_{i,a} h_\theta^2(\mathbf{y}_{ia}) = n\lambda$. Then there exists at least one permutation $\mathbf{z}^*$ satisfying the condition. QED. □

3.4 Two Optimization Strategies for Generalized GM

Algorithm 1 Path following for GGM

```
Input: $\mathbf{A}$, $h_\theta$, $\theta_0$, $0 < \alpha < 1$, initial $\mathbf{z}_0$, $k$
Output: $\mathbf{z}$
$\mathbf{z} \leftarrow \mathbf{z}_0$, $\theta \leftarrow \theta_0$
repeat
    make problem according to (3.6) with $\theta$
    repeat
        compute $\mathbf{V}$ using Eq. (3.14)
        $\mathbf{z} = \mathbf{z} + \epsilon \text{vec}(\mathbf{V})$
    until Converge
    $\theta \leftarrow \alpha\theta$
until $\theta < k$
```

Algorithm 2 Multiplicative strategy for GGM

```
Input: $\mathbf{A}$, $h_\theta$, initial $\mathbf{z}_0$
Output: $\mathbf{z}$
$\mathbf{z} \leftarrow \mathbf{z}_0$
repeat
    $\mathbf{h} \leftarrow h_\theta(\mathbf{z})$
    $\mathbf{h} \leftarrow \mathbf{A}\mathbf{h}$
    $\mathbf{z} \leftarrow h_\theta^{-1}(\mathbf{h})$
until Converge
```

3.4.1 Path Following Strategy

It is observed that solving the problem when θ is too close to 0 is highly non-convex, suggesting the existence of many local optima. Instead, moderate smoothness is desired when we initiate the optimization. This naturally leads to the path following strategy. Such optimization is involved in (Gold and Rangarajan, 1996; Zhou and Torre, 2012; **?**). In our implementation, we start by obtaining a local optimum $\mathbf{z}_1^*$ from a relatively tractable problem $\mathcal{P}_{\theta_1}$, then we shrink the value of θ_1 by letting $\theta_2 = \alpha\theta_1$ where $0 < \alpha < 1$. Let the starting point for next iteration be $\mathbf{z}_1^*$, we solve the updated problem $\mathcal{P}_{\theta_2}$. The iteration continues until convergence condition is satisfied. To verify the convergence, we calculate the energy gap between two consecutive iterations. Formally, for current $\mathbf{z}^{(t)}$ at iteration t, we calculate the corresponding energy $\mathcal{E}^{(t)} = \mathbf{z}^{(t)\top}\mathbf{A}\mathbf{z}^{(t)}$. The energy at previous iteration $t-1$ is analogously calculated as $\mathcal{E}^{(t-1)} = \mathbf{z}^{(t-1)\top}\mathbf{A}\mathbf{z}^{(t-1)}$. Then if $\left|\mathcal{E}^{(t)} - \mathcal{E}^{(t-1)}\right| < \eta$, where η is a small positive value, we identify the convergence of the iteration. If there is no such t, the algorithm stops when reaching the pre-defined maximal iteration number. In all the following experiments, we let $\eta = 10^{-8}$.

Note the problem $\mathcal{P}_\theta$ is a general objective with affine constraints. For any gradient-based strategy, projection is necessary to mapping the current solution back to the feasible set. As discussed in Jiang *et al.* (2017a), projection in variable domain may lead to weak optima. Instead, we use Iterative Bregmann Gradient Projection (IBGP), which is performed in the gradient domain and the convergence is guaranteed (Yu *et al.*, 2018c). Given current gradient $\mathbf{U} = \text{mat}(\nabla\mathbf{z})$, previous matching $\mathbf{Z}$ and step length ϵ, IBGP performs the following calculations iteratively to obtain $\mathbf{V}$ until convergence:

$$\mathbf{V} = \mathbf{U} - \frac{1}{n}\mathbf{U}\mathbf{1}\mathbf{1}^T - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\mathbf{U} + \frac{2}{n^2}\mathbf{1}\mathbf{1}^\top\mathbf{U}\mathbf{1}\mathbf{1}^\top \tag{3.14a}$$

$$\mathbf{V}_{ij} = -\mathbf{Z}_{ij}/\epsilon \quad \text{if} \quad \mathbf{V}_{ij} < -\mathbf{Z}_{ij}/\epsilon \tag{3.14b}$$

$$\mathbf{V}_{ij} = (1-\mathbf{Z}_{ij})/\epsilon \quad \text{if} \quad \mathbf{V}_{ij} > (1-\mathbf{Z}_{ij})/\epsilon \tag{3.14c}$$

where $\mathbf{V}$ is the update direction within the feasible set. Note the iterative procedure in the above equation is a projection. As the constraint set is convex (affinity set), the projection convergence is ensured. Thus in each iteration of update, the algorithm seeks a direction $\mathbf{V}$ with ascending guarantee and proceeds a fixed length ϵ. This procedure iterates until convergence or maximal step number. The path following method is summarized in Algorithm 1.

3.4.2 Multiplication Strategy

Multiplicative strategy on optimizing quadratic objective proved to be convergent under the assumption that $\mathbf{A}$ is positive semi-definite (Rangarajan *et al.*, 1999b). In this strategy, each step amounts to a multiplication $\mathbf{z}^{(t+1)} = \mathbf{A}\mathbf{z}^{(t)}$ and the objective score over the solution path is non-decreasing. There are works (Cho *et al.*, 2010; Gold and Rangarajan, 1996; Tian *et al.*, 2012) falling into this category. However, in general affinity $\mathbf{A}$ is barely positive semi-definite. While some methods handle this circumstance by adding reweighted identity matrix to $\mathbf{A}$ (Jiang *et al.*, 2017b), others simply neglect the non-decreasing constraint including some popular algorithms (Cho *et al.*, 2010; Gold and Rangarajan, 1996). The empirical success of such methods suggests pursuing objective ascending and enhancing matching accuracy sometimes are paradox. Moreover, the recent study (Yan *et al.*, 2016) further shows due to noise and the parametric modeling limitation of the affinity function, high accuracy may even corresponds to lower affinity score. Inspired by these observations, we devise a simple yet effective multiplicative strategy by omitting the non-decreasing check. The procedure is shown in Algorithm 2. In this strategy, the update rule involves calculating inverse function of h_θ. While it is found the multiplicative method converges much faster and hence the overall run time is less compared with the path following method.

3.5 Experiments

Three popular benchmarks are used including Random Graph Matching (Cho *et al.*, 2010), CMU house sequence (Caetano *et al.*, 2006) and Caltech-101/MSRC object matching (Cho *et al.*, 2010). *accuracy*, *score* and *ratio* are evaluated, where *accuracy* measures the portion of correctly matched nodes with respect to all nodes, *score* represents the value of the objective function and *ratio* emphasizes the ratio between current objective value and the maximal one. The algorithms for comparison include Spectral Matching (**SM**) (Leordeanu and Hebert, 2005), Integer Projected Fixed Point (**IPFP**) (Leordeanu *et al.*, 2009), Graduated Assignment (**GAGM**) (Gold and Rangarajan, 1996), Reweighted Random Walk (**RRWM**) (Cho *et al.*, 2010), Soft-restricted Graduated Assignment (**SRGA**) (Tian *et al.*, 2012), Factorized Graph Matching (**FGM**) (Zhou and Torre, 2012) and Branching Path Following Matching (**BPM**) (**?**). We term our algorithm Generalized Graph Matching (**GGM**) with a subscript indicating the corresponding Separable Function and optimization strategy. Namely, $\mathbf{GGM}_{\mathrm{xy}}$ represents the method with Separable Function $\mathrm{x} \in \{\mathrm{l} : h_{\mathrm{Lap}}; \mathrm{p} : h_{\mathrm{Poly}}\}$ and optimizing strategy $\mathrm{y} \in \{\mathrm{p} : \text{Path following}; \mathrm{m} : \text{Multiplication}\}$. In all the experiments, the algorithms with any updating rules are initialized with a uniform matching. For path following strategy of GGM , we set $\theta_0 = 2$, $\alpha = 0.5$, $k = 0.2$.

3.5.1 Results on Random Graph

This test is performed following the protocol in Cho *et al.* (2010). For each trial, source graph $\mathcal{G}_S$ and destination graph $\mathcal{G}_D$ with n_{in} inlier nodes are generated, consisting of vector attributes $\mathbf{a}_{ij}^S$ and $\mathbf{a}_{ij}^D$ for both nodes and edges (note $\mathbf{a}_{ii}$ is a node attribute and $\mathbf{a}_{ij}$ is an edge attribute when $i \neq j$.). In the initial setting, $\mathcal{G}_D$ is

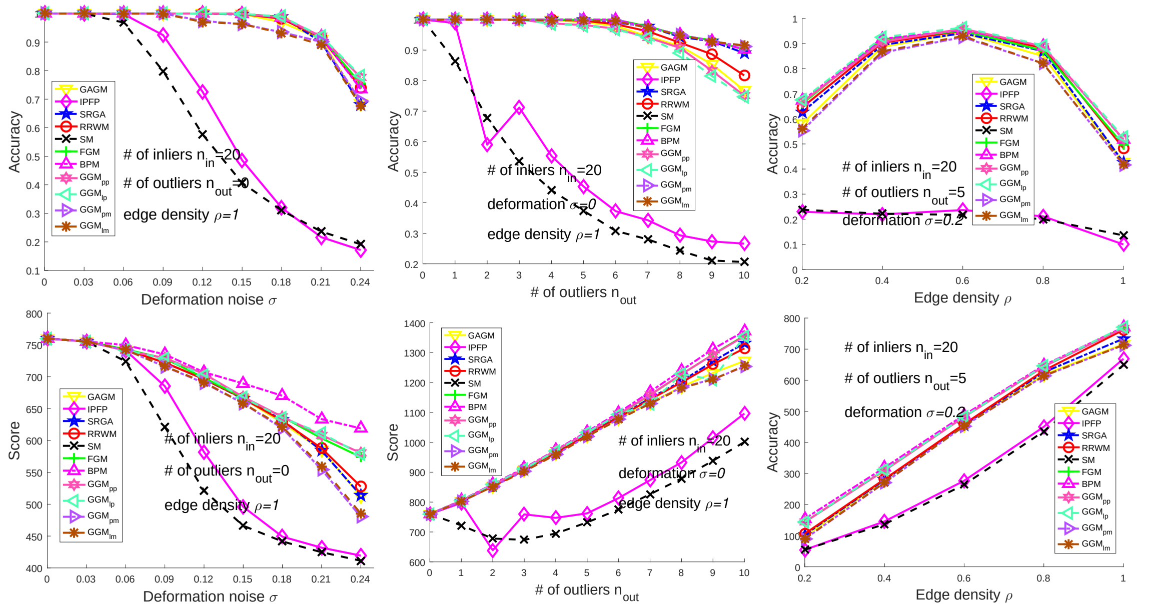


Figure 3.3: Performance on Random Graphs. Note BPM (Wang *et al.*, 2016)'s Runtime Is Significantly More Expensive than Other Methods (Empirically an Order Higher than Ours Using the Public Source Code) as It Simultaneously Seeks Multiple Paths for the Best Score (Though Accuracy Is Similar to Ours). In Contrast, Our Method Focus on One Path No Matter the Path Following or Multiplicative Strategy Is Used.

the replica of $\mathcal{G}_S$. Three types of sub-experiments are conducted with varying graph deformation σ, number of outliers n_{out} and edge density ρ. To deform a graph, we add an independent Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma)$ to attribute $\mathbf{a}_{ij}^D$ such that $\mathbf{a}_{ij}^D = \mathbf{a}_{ij}^S + \varepsilon$. Thus the resulting affinity is calculated by $\mathbf{A}_{ij:ab} = \exp(-|\mathbf{a}_{ij}^S - \mathbf{a}_{ab}^D|^2/\sigma_s^2)$. The parameter σ_s is empirically set to be 0.15. In outlier test, we generate the same number of outlier nodes for both graph. In edge density test, we randomly sample ρ portion of corresponding edges from two fully connected graphs. Each type of sub-experiment is independently carried out 500 times, while average *accuracy* and *score* are calculated.

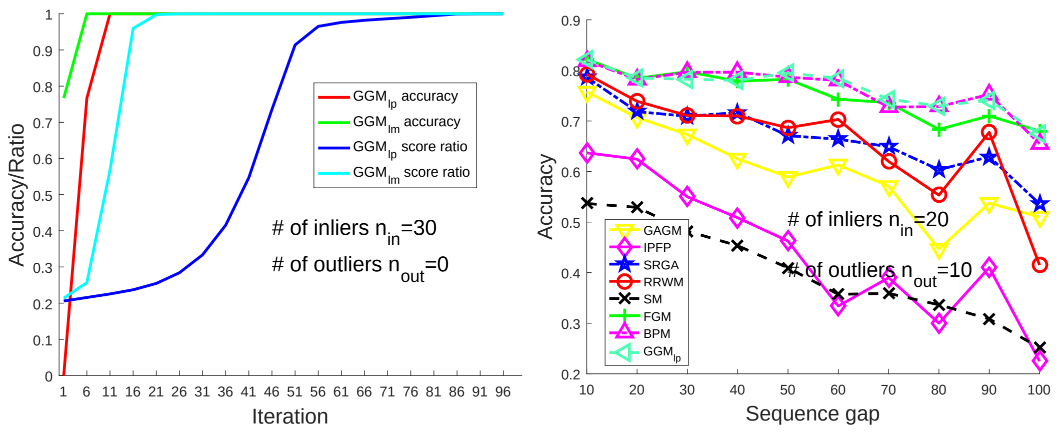


Figure 3.4: Results on CMU House. Left: Convergence Speed VS Iteration. Right: Accuracy by Frame Gap.

Results are shown in Fig 3.3. In the deformation and the edge density tests, GGM$_{pp}$ and GGM$_{lp}$ achieve competitive performance compared to state-of-the-art algorithms. Especially when there is combination of severe deformation and edge density is low, GGM$_{pp}$ and GGM$_{lp}$ outperform the selected counterparts. On the other hand, GGM$_{pm}$ and GGM$_{lm}$ reach significant performance close to state-of-the-art e.g. BPM **?**. Though multiplicative strategies cannot guarantee ascending objective in each iteration, GGM$_{pm}$ and GGM$_{lm}$ are still effective. This supports the discussion of the paradox between matching accuracy and objective score in Section 3.4.2. We only show results of GGM$_{\mathrm{lp}}$ in the following experiments, as we see no notable performance gap compared to the other settings.

To examine the algorithm sensitivity to the parameters, we also conduct an extra Random Graph Matching experiment with SFs h_{Poly} and h_{Lap} on Algorithm 1. In this test, we let deformation noise 0.15 and edge density 0.8, 20 inliers and 5 outliers. Test is carried out independently for 20 times and the average accuracy is reported. For both the SFs, we observe that $k = 0.2$ is sufficient to produce satisfying matching accuracy. Thus we conduct the test by varying the values of θ_0 and α. The results are

Table 3.1: Sensitivity Test on h_{Poly} and h_{Lap}

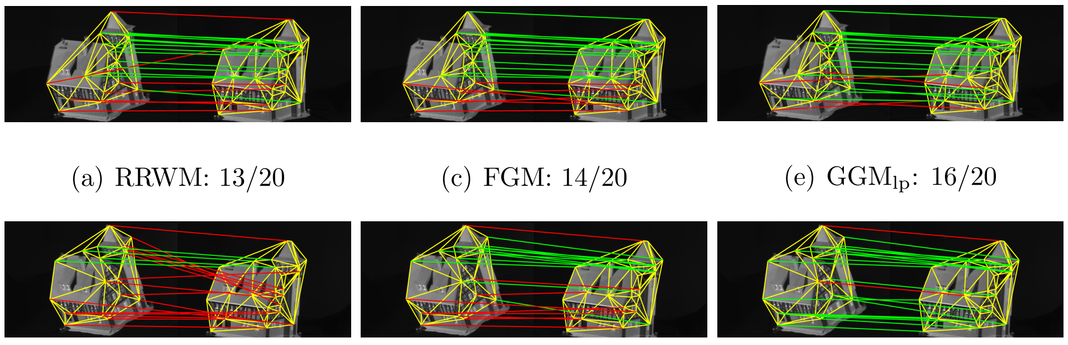
h_{Poly}		α				
		0.7	0.6	0.5	0.4	0.3
θ_0	3	0.842	0.839	0.841	0.721	0.610
	2	0.905	0.905	0.904	0.848	0.725
	1	0.910	0.905	0.908	0.851	0.717
	0.5	0.823	0.814	0.770	0.652	0.422

h_{Lap}		α				
		0.7	0.6	0.5	0.4	0.3
θ_0	4	0.912	0.909	0.910	0.872	0.685
	3	0.911	0.907	0.903	0.836	0.672
	2	0.904	0.904	0.906	0.811	0.567
	1	0.853	0.844	0.810	0.728	0.472

Table 3.2: Performance on Natural Images from Caltech-101 and MSRC Datasets.

Method	GAGM	IPFP	SRGA	RRWM	SM	FGM	BPM	GGM_{lp}
accuracy (%)	73.66	75.77	72.86	72.95	65.78	76.35	75.14	**76.69**
score ratio	0.933	0.942	0.940	0.946	0.735	0.969	1	0.972

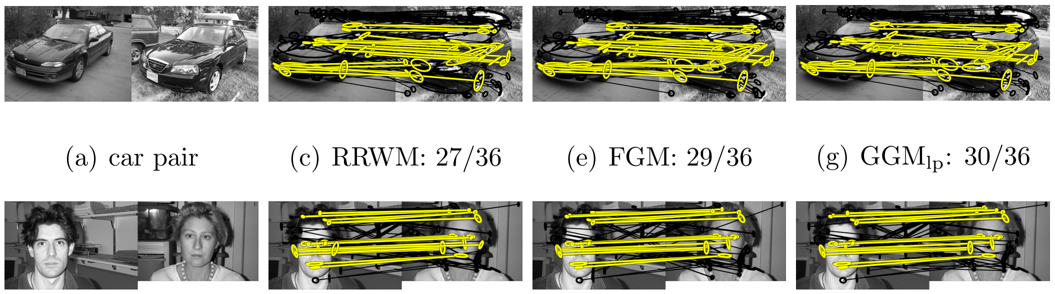
demonstrated in Table 3.1. As larger θ_0 and α indicate more iterations, and $\theta_0 < 2$ and $\alpha < 0.5$ result in decreasing behavior, we employ the setting $\theta_0 = 2$ and $\alpha = 0.5$ throughout all experiments.



(a) RRWM: 13/20 (c) FGM: 14/20 (e) GGM_{lp}: 16/20

(b) RRWM: 4/20 (d) FGM: 11/20 (f) GGM_{lp}: 18/20

Figure 3.5: Top and Bottom Row Shows Examples on CMU House Sequence with Gap 20 and 80 Respectively, by Setting $(n^S = 30, n^D = 20)$.



(a) car pair (c) RRWM: 27/36 (e) FGM: 29/36 (g) GGM_{lp}: 30/36

(b) face pair (d) RRWM: 32/40 (f) FGM: 33/40 (h) GGM_{lp}: 35/40

Figure 3.6: Examples of Matchings on Selected Caltech-101 and MSRC Datasets.

3.5.2 Results on CMU House Sequence

We perform feature point matching on widely used CMU house sequence dataset following the settings in Caetano *et al.* (2006); Cho *et al.* (2010). The dataset consists of 111 house images with gradually changing view points. There are 30 landmark points in each frame. Following the protocol in Cho *et al.* (2010); **?**, matching test is conducted on totally 560 pairs of images, spaced by varying frame gaps $(10, 20, ..., 100)$. We use 2 settings of nodes $(n^S, n^D) = (30, 30)$ and $(20, 30)$. In case $n^S < 30$, n^S nodes are randomly sampled from the source graph. The affinity is conducted by $\mathbf{A}_{ij:ab} = \exp(-|\mathbf{a}_{ij}^S - \mathbf{a}_{ab}^D|^2/\sigma_s^2)$, where $\mathbf{a}_{ij}^S$ measures the Euclidean distance between point i and j, and $\sigma_s^2 = 2500$. The edge density is set by $\rho = 1$. One can see when there is no outlier, all methods except for IPFP and SM achieve perfect matching on any gap setting, and we only show the results with outliers. Figure 3.5 and Figure 3.4 depict the matching samples and performance curve, respectively. We also show typical converging behavior of $\mathrm{GGM}_{\mathrm{lp}}$ and $\mathrm{GGM}_{\mathrm{lm}}$ on the upper of Figure 3.4. We note our path following strategy (Alg. 1) converges slower than multiplicative one (Alg. 2) and they obtain similar final accuracy. One can see when there exist outlier points, GAGM and RRWM suffer notable degraded performance. Our algorithm, on the other hand, achieves competitive performance to state-of-the-arts and behaves stably even under severe degradations.

3.5.3 Results on Natural Image Matching

This is a challenging dataset as it includes natural images in arbitrary backgrounds. In line with the protocol in Cho *et al.* (2010), 30 pairs of images are included in this test collected from Caltech-101 (Fei-Fei *et al.*, 2007) and MSRC [3] . In each pair of images, MSER detector (Matas *et al.*, 2004) and SIFT descriptor (Lowe, 1999) are used to obtain the key points and the corresponding node feature. Mutual projection error function (Cho *et al.*, 2009) is further adopted to calculate the edge affinity. The ground-truth are manually labeled. The results are shown in Table 4.1 and matching examples are shown in Fig. 3.6. Our method outperforms selected algorithms w.r.t. accuracy regardless of objective score. This also suggests the paradox between accuracy and score under complex affinity modeling as discussed in Yan *et al.* (2016).

[3]http://research.microsoft.com/vision/cambridge/recognition/

3.6 Conclusion

By using Separable Functions, we present a family of continuous approximations to the vanilla QAP formulation widely used in graph matching. We explore the relation of such approximations to the original discrete matching problem, and show convergence properties under mild conditions. Based on the theoretical anslysis, we propose a novel solver GGM, which achieves remarkable performance in both synthetic and real-world image tests. This gives rise to the possibility of solving graph matching with many alternative approximations with different solution paths.

Chapter 4

GRADIENT-EFFICIENT REGULARIZATION

4.1 Problem Statement

Over the decades, extensive works have been developed on graph matching. Traditionally, most methods are focused on pairwise graph matching, i.e., each time only two graphs are involved for matching. Due to its NP-hardness, most methods seek approximation techniques to pursuit the trade-off between accuracy and efficiency. A popular conversion treatment from continuous to discrete solution is applying greedy algorithm or Hungarian method as projection. However such a conversion is likely to bring about arbitrary accuracy degeneration to the final solution.

Recently, a few regularizers have been developed and become an important way to graduated discretization along the solution path. Examples include entropy (Tian *et al.*, 2012), factorization based convex-concave relaxation (Zhou and Torre, 2013), and ℓ_2 norm (Jiang *et al.*, 2017b)). However, there still lacks a clear investigation on the gradient behavior during the optimization: how does the regularized gradient impact the solution path and what can we do to improve?

This part is focused on investigating and improving the effectiveness of gradient provided by regularization, by providing more reliable gradient direction along the continuous solution path (being a numerical continuation method). We also develop a new regularization technique combined with the simple gradient based continuous optimization. It explores the determinant of the matching matrix which is relaxed in the continuous domain, and achieves superior performance among several gradient-based solvers, while existing regularized methods (Gold and Rangarajan, 1996; Tian *et al.*, 2012; Jiang *et al.*, 2017b) often perform less competitive and also are algorithmically more complex. The main contributions are summarized as:

i) To enable gradient-efficient continuation optimization for graph matching, we propose a novel graduated discretization technique. Specifically a determinant regularization technique is devised on the matching solution. We analytically show the geometric property of our method compared to existing regularizers;

ii) We develop two types of sub-gradient updating rules to address the issue of irregular solution matrix, which have been proved effective and moderately efficient;

iii) Our approach shows promising experimental results on public benchmarks. Thanks to the clear theoretical foundation, the algorithm procedure is in general simple and insensitive to hyperparameters.

Notations are used as follows. Bold lower case $\mathbf{x}$ and upper case $\mathbf{X}$ refer to vector and matrix, respectively. While $\det(\cdot)$ and $\text{sign}(\cdot)$ calculate matrix determinant and sign of a real value, respectively, and $\text{diag}(\cdot)$ maps a vector to a diagonal matrix, and vice versa. $\mathbf{1}$ is a vector with all 1.

4.2 GM via Determinant Regularization

4.2.1 Theorem and Derived Objectives

Graph matching can be relaxed as following:

$$\max_{\mathbf{z}} \mathbf{z}^\top \mathbf{K} \mathbf{z}, \quad \text{s.t. } \mathbf{H}\mathbf{z} = \mathbf{1}, \quad \mathbf{z} \in [0,1]^{n^2} \tag{4.1}$$

where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is the so-called affinity matrix which is assumed pre-given in this chapter, and $\mathbf{H}$ is a selection matrix encoding the one-to-one constraint for node correspondence. Here n is the number of nodes in each graph. Now we first present the following proposition.

Proposition 4. *For any doubly stochastic matrix* $\mathbf{Z}$*, we have* $|\det(\mathbf{Z})| \leq 1$*. Equality holds iff* $\mathbf{Z}$ *is a permutation.*

Proof. According to Hadamard's inequality (Fink, 2000), we have $|\det(\mathbf{Z})| \leq \prod_{j=1}^{n} \|\mathbf{Z}_j\|$, where $\mathbf{Z}_j$ refers to the jth column of $\mathbf{Z}$. As $\mathbf{Z}$ is doubly stochastic, we have $\mathbf{Z}_{ij} \in [0,1]$ and $\sum_i \mathbf{Z}_{ij} = 1$. Thus $\|\mathbf{Z}_j\| \leq 1$ and equality holds iff there is only a 1 element in $\mathbf{Z}_j$ and all the resting elements are 0s. Therefore, for any column j that's not in 0-1 mode, we must have $\|\mathbf{Z}_j\| < 1$. This observation claims that, if $\mathbf{Z}$ is not a permutation (in this case non-0/non-1 element exists), $|\det(\mathbf{Z})| \leq \prod_{j=1}^{n} \|\mathbf{Z}_j\| < 1$.

On the other hand, for any permutation $\mathbf{Z}$, we must have $|\det(\mathbf{Z})| = 1$ (As the columns of a doubly stochastic matrix are orthogonal). QED. □

This proposition gives an upper bound for the absolute value of the determinant. Together with the trivial lower bound we have $0 \leq |\det(\mathbf{Z})| \leq 1$. The equality of the 0 side holds when the columns/rows of $\mathbf{Z}$ are linearly dependent. Based on the discussion above, we construct the following objective involving determinant regularization:

$$\max_{\mathbf{z}} \mathbf{z}^\top \mathbf{K} \mathbf{z} + \lambda |\det(\mathbf{Z})|, \quad \text{s.t. } \mathbf{H}\mathbf{z} = \mathbf{1} \tag{4.2}$$

We refer to objective (4.2) as $\mathbb{P}_\lambda$. Given objective (4.2) with a regularization term, two types of optimization strategies, namely multiplication and gradient, are widely applied. The multiplicative strategy is established on certain convergence guarantee where in the current iteration the updated affinity value is greater than the previous one under some mild conditions (Jiang *et al.*, 2017a; Gold and Rangarajan, 1996). In our case, however, it is difficult to devise a multiplication-based updating paradigm since the convergence condition involving matrix inversion cannot be easily found. Instead, we develop a gradient-based optimization algorithm under the framework of path-following (Zhou and Torre, 2012). Though we still face the difficulty of calculating matrix inversion in this setting, we provide an effective and efficient way to approximate the gradient of low-rank solution. These will be discussed in Section 4.3. Although it was reported that the multiplicative updating rule is more computationally efficient, we have found that our gradient-based algorithm can achieve remarkable improvement compared to multiplication-based ones in a reasonable time cost.

4.2.2 Geometric Property of Gradients

We show some geometric properties of the proposed models in this section and analyze the gradient behavior compared to two regularization counterparts. To this end, we unfold our analysis in Euclidean space, which is more intuitive. We also choose regularization model with ℓ_2 (Jiang *et al.*, 2017b) ($\sum_i \mathbf{z}_i^2$) and entropy (Gold and Rangarajan, 1996) ($-\sum_i \mathbf{z}_i \log(\mathbf{z}_i)$) for comparison on the polytope. As the underlying contours are visualized, we demonstrate the property of determinant by showing its bound vs ℓ_2 norm. Let us consider the following two sets:

$$\mathcal{S}_1 = \{\mathbf{Z} | \|\mathbf{Z}\|_F^2 = n\} \quad \mathcal{S}_2 = \{\mathbf{Z} | |\det(\mathbf{Z})| = 1\} \tag{4.3}$$

where $\mathcal{S}_1$ is the regularization introduced in Jiang *et al.* (2017b). For any $\mathbf{Z} \in \mathcal{S}_1$, we have the following formula according to the geometric inequality and Hadamard's inequality:

$$\|\mathbf{Z}\|_F^2/n = \sum_{j=1}^{n} \|\mathbf{Z}_j\|_2^2/n \geq \sqrt[n]{\prod_{j=1}^{n} \|\mathbf{Z}_j\|_2^2} \geq \sqrt[n]{|\det(\mathbf{Z})|^2} \tag{4.4}$$

In general, we have $|\det(\mathbf{Z})| \leq 1$ if $\mathbf{Z} \in \mathcal{S}_1$. The above analysis implies that, the set of $\{\mathbf{Z} | \|\mathbf{Z}\|_F^2 \leq n\}$ is a proper subset of $\{\mathbf{Z} | |\det(\mathbf{Z})| \leq 1\}$. To show the geometric relation among these regularizers in 2D space, we present an example by projecting the 2-permutation polytope onto the 2-dimensional Euclidean space, with two permutation:

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{A}_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{4.5}$$

Any point on this polytope can be expressed as:

$$\alpha \mathbf{A}_1 + (1-\alpha)\mathbf{A}_2 \tag{4.6}$$

We first give a 2-dimensional expression of determinant regularizer. Given Eq. (4.6), the corresponding absolute determinant value given α is:

$$\left| \det \left\{ \begin{pmatrix} \alpha & 1-\alpha \\ 1-\alpha & \alpha \end{pmatrix} \right\} \right| = |2\alpha - 1| \tag{4.7}$$

which is merely dependent on variable α. Upon this juncture, we linearly map permutation vertices $\mathbf{A}_1$, $\mathbf{A}_2$ and original point $(0,0;0,0)$ onto $(0,1)$, $(1,0)$ and $(0,0)$ in regular 2D space, respectively. This can be done with a naive projection matrix from vectorized 4D to 2D space:

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0.5 & 0 \end{pmatrix} \tag{4.8}$$

In this case the 2-permutation polytope in a regular 2D Euclidean space is the interval between points $(0,1)$ and $(1,0)$. If we rescale any point on the polytope by $b > 0$, then the absolute value of determinant is $b^2|2\alpha - 1|$. This implies that the absolute value of determinant is the product of rescaled point $b|2\alpha - 1|$ and b. Since b corresponds to an axis orthogonal to the polytope, we conclude that the contour of absolute value of determinant is a hyperbola rotated by $45°$. The contours of ℓ_2 and entropy can be analogously obtained.

The contours of three types of regularizers can be found in Figure 4.1. Serving as another regulizer besides ℓ_2 and entropy, we note that absolute determinant shows much different behavior than the other two. When the current solution is in the interior of the polytope and near to the point $\frac{1}{n}\mathbf{1}\mathbf{1}^\top$, the gradients of ℓ_2 and entropy are almost orthogonal to the polytope. In this case, the gradients of regularizers have *no contribution* to the update. However, absolute determinant has more effective gradient (almost in the polytope subspace) in the interior. This effectiveness also holds along with the solution path until polytope vertices, yielding contributing gradient in each iteration.

Furthermore, it is a fact that $|\det(\mathbf{Z})| > 0$ when $\mathbf{X}$ is strictly diagonal dominant (i.e. there exist an i for each j, such that $|\mathbf{Z}_{ij}| \geq \sum_{k \neq i} \mathbf{Z}_{kj}$). Thus for any (not necessarily doubly) stochastic matrix $\mathbf{Y}$, strictly diagonal dominance implies there exist an i such that $\mathbf{Y}_{ij} > 0.5$ for any j. As for doubly stochastic matrix, there will be only one element larger than 0.5 in each column/row, which means there's no ambiguity. In this case, there exists a permutation $\mathbf{P}$ such that $\mathbf{P}\mathbf{X}\mathbf{P}^\top$ is diagonally dominant. We call such $\mathbf{Z}$ **doubly diagonally dominant**. Conversely, if for any continuous solution $\det(\mathbf{Z}) = 0$, $\mathbf{Z}$ cannot be doubly diagonally dominance, which implies ambiguity and is not of our interest. In general, larger determinant value of doubly stochastic solution implies less ambiguity, and a absolute determinant larger than 0 implies strong discriminativity to some extent ($\mathbf{Z}_{ij} > 0.5$ exists).

4.3 Optimization Methods

4.3.1 Path-following for solving Objective (4.2)

Given objective (4.2), we devise a path-following-based method. Path-following, sometimes also called continuation method, is widely adopted in previous works on graph matching e.g. Zhou and Torre (2012). While the details differ, the common advantage of such strategy includes a guarantee on increasing objective score, as

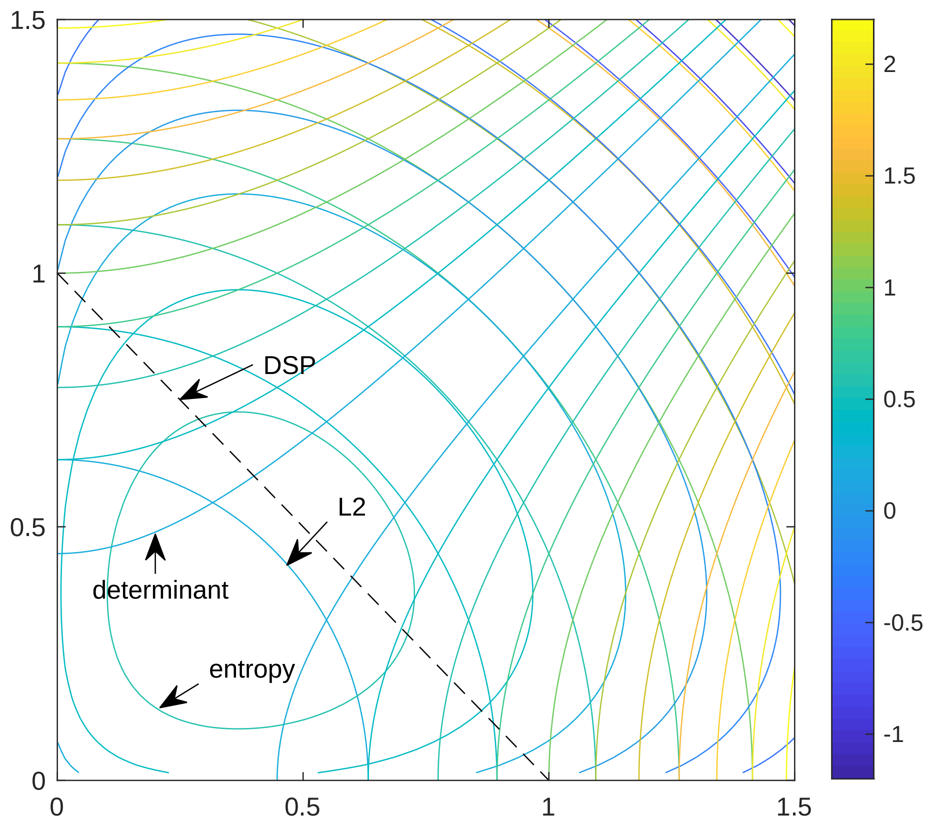


Figure 4.1: Contours of Three Regularization Techniques on 2D Euclidean Space. Black Arrows Indicate the Contours of the Regularization Terms: Entropy, ℓ_2 Norm (ℓ_2) and Determinant (DET). The Black Dashed Line Refers to a 2-dimensional Doubly Stochastic Polytope (DSP), Where the Discrete Solutions Lie on Coordinates $(0, 1)$ or $(1, 0)$. We See along with Emphasizing the Regularization Weights, Either Term Will Push the Solution to the Discrete Ones. However, in the Deep Interior of the DSP, Determinant Has Much Different Behavior Against the Other Two.

well as the optimum being a permutation, since a discrete solution is what we seek for the original problem. Another reason that we employ path-following rather than multiplication is that it is extremely difficult to find a theoretical ascending guarantee for the determinant-regularized model under multiplicative strategy.

Such an algorithm requires the calculation of gradient of $\det(\mathbf{Z})$ w.r.t. $\mathbf{Z}$ involve a matrix inversion. However in practice, $\mathbf{Z}$ is not necessarily invertible, especially in the early stage of optimization when $\mathbf{Z}$ is deep in the interior of the permutation polytope. For the time being, we assume $\mathbf{Z}$ is invertible, hence $\mathbf{Z}^{-1}$ exists. We will discuss how to handle the case of low-rank $\mathbf{Z}$ in the next section.

Our path-following starts from an initial $\lambda_0 > 0$. After a gradient-based algorithm finds the local optimum $\mathbf{Z}_0^{\mathcal{O}}$ corresponding to λ_0, it proceeds by amplifying the value with $\lambda_{t+1} > \lambda_t$ and solves problem $\mathbb{P}_{\lambda_{t+1}}$ for the next stage. In all the experiments we employ the amplifying procedure $\lambda_{t+1} = 2\lambda_t$. The gradient of (4.2) w.r.t. $\mathbf{Z}$ is:

$$\hat{\mathbf{Z}} = 2\mathbf{Kz} + \lambda \text{sign}(\det(\mathbf{Z}))\mathbf{Z}^{-\top} \tag{4.9}$$

Then the algorithm iterates by $\mathbf{Z} = \mathbf{Z} + \Delta\hat{\mathbf{Z}}$, where $\Delta > 0$ is a small increment. Likewise we need to project $\mathbf{Z}$ back to the permutation polytope. Rather than Sinkhorn's algorithm (Sinkhorn, 1964), we employ a slightly revised projection method from Yu *et al.* (2018c) which manipulates columns and rows simultaneously. To this end, we first let $\mathbf{Z}_{ij} = 0$ if $\mathbf{Z}_{ij} < 0$, then repeat the following projection:

$$\mathbf{Z} = \mathbf{Z} + \frac{1}{n}\mathbf{1}\mathbf{1}^\top - \frac{1}{n}\mathbf{Z}\mathbf{1}\mathbf{1}^\top - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\mathbf{Z} + \frac{1}{n^2}\mathbf{1}\mathbf{1}^\top\mathbf{Z}\mathbf{1}\mathbf{1}^\top \tag{4.10}$$

which theoretically ensures to converge to a point in the permutation polytope. This algorithm has fast convergence in practice (less than 10 iterations to an acceptable precision).

4.3.2 Spectral Sub-gradient

As discussed earlier, $\mathbf{Z}$ is not necessarily invertible. In this case, the assumption of updating rule (4.9) no longer holds. This issue possibly happens when the current solution is close to the center of the polytope interior. One may imagine a naive replacement of matrix inversion with the pseudo-inversion, which can be applied on singular matrices. However, pseudo-inversion will keep the zero eigenvalues intact, and will not fulfill the purpose of maximizing the absolute determinant.

To address this issue, we first diverge to look into the partial objective $\max |\det(\mathbf{Z})|$. Since this partial term seeks to maximize the absolute determinant, any small value above 0 will be a proper increment on $|\det(\mathbf{Z})|$ if the current determinant is 0. Thus any incremental direction in terms of absolute determinant implies a proper "sub-gradient", and we can adopt such sub-gradient for update. To this end, we first perform eigen-decomposition on $\mathbf{Z}$:

$$\mathbf{Z} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} \tag{4.11}$$

where $\mathbf{U}$ is an orthogonal matrix containing both the basis of linear and null spaces and $\mathbf{\Lambda}$ is the diagonal matrix with eigen-values in the magnitude descending order $(\sigma_1, ..., \sigma_q, 0, ..., 0)$, $|\sigma_{t-1}| \geq |\sigma_t|$ for any t. Now we present two types of proper sub-gradients.

Method Design by Increment Update: DetGM$_1$

We employ a simple yet effective amplification procedure by letting any eigenvalue with magnitude smaller than a tolerance $\epsilon > 0$ to ϵ. The amplified eigenvalues become $(\sigma_1, ..., \sigma_s, \text{sign}(\sigma_{s+1})\sigma_{s+1}, ..., \text{sign}(\sigma_r)\sigma_r, \epsilon, ..., \epsilon)$, where $s \leq q$, $0 < |\sigma_{s+1}| < \epsilon$, $|\sigma_s| \geq \epsilon$, $\sigma_r \neq 0$ and $\sigma_{r+1} = 0$ in the original eigen-values and q is the index of the first element which is negative value but with magnitude larger than ϵ. We add a tolerance ϵ to accommodate the calculating precision of float numbers. As such, each eigen-value is above 0, thus the determinant will not vanish. Let the diagonalized amplified eigenvalue matrix be $\hat{\mathbf{\Lambda}}$, then the modified matrix with small non-zero determinant can be written as:

$$\mathbf{Z}_u = \mathbf{U}\hat{\mathbf{\Lambda}}\mathbf{U}^{-1} \tag{4.12}$$

Algorithm 3 Spectral sub-gradient for DetGM$_1$.

```
Input: $\mathbf{Z}$; tolerance $\epsilon$
Output:sub-gradient $\bar{\mathbf{Z}}$
$\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1} \leftarrow \mathbf{Z}$ (eigen-decomposition)
$\hat{\boldsymbol{\Lambda}} \leftarrow \boldsymbol{\Lambda}$
for $\forall i, |\boldsymbol{\Lambda}_{ii}| < \epsilon$ do
    if $\boldsymbol{\Lambda}_{ii} = 0$ then
        $\hat{\boldsymbol{\Lambda}}_{ii} = \epsilon$
    end if
    $\hat{\boldsymbol{\Lambda}}_{ii} = \text{sign}(\boldsymbol{\Lambda}_{ii})\epsilon$
end for
$\mathbf{Z}_u = \mathbf{U}\hat{\boldsymbol{\Lambda}}\mathbf{U}^{-1}$
return $\bar{\mathbf{Z}} = \mathbf{Z}_u - \mathbf{Z}$
```

Then the difference $\bar{\mathbf{Z}} = \mathbf{Z}_u - \mathbf{Z} \approx \frac{\partial|\det(\mathbf{Z})|}{\mathbf{Z}}$ can be viewed as a proper ascending direction w.r.t. $\mathbf{Z}$, as by adding $\bar{\mathbf{Z}}$, $|\det(\mathbf{Z})|$ becomes above 0. This procedure is summarized in Algorithm 3.

Method Design by Geometric Update: DetGM$_2$

This line of sub-gradient is motivated by the geometric inequality $\frac{1}{n}\sum_i a_i \geq \sqrt[n]{\prod_i a_i}$ for $a_i \geq 0$. It is easy to show that $\prod_i a_i$ is concave in the affine subset $\sum_i a_i = d$, where $a_i \geq 0$ and $d > 0$ is a constant. The maximal value is reached if and only if $a_i = d/n$ for each i. According to the concavity and the reachable maximum, it is easy to conclude that for any $0 < \theta < 1$ and the reweighted point $\mathbf{b} = (1-\theta)\mathbf{a}+\theta\mathbf{m}$, it have to be $\prod_i b_i \geq \prod_i a_i$. Here $\mathbf{a} = (a_1, ..., a_n)$ and $\mathbf{m} = (n/d, ..., n/d)$. Motivated by this observation, we devise another type of valid sub-gradient for absolute determinant in graph matching.

Algorithm 4 Spectral sub-gradient for DetGM$_2$.

```
Input: $\mathbf{Z}$; tolerance $\epsilon$; step $\theta$
Output: sub-gradient $\bar{\mathbf{Z}}$
$\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} \leftarrow \mathbf{Z}$ (eigen-decomposition)
$g \leftarrow \sum_i |\mathbf{\Lambda}_{ii}|$
for all $i$ do
    if $\mathbf{\Lambda}_{ii} \neq 0$ then
        $\mathbf{\Gamma}_i \leftarrow \frac{g}{n}\text{sign}(\mathbf{\Lambda}_{ii}) - \mathbf{\Lambda}_{ii}$
    else
        $\mathbf{\Gamma}_i \leftarrow \frac{g}{n}$
    end if
end for
return $\bar{\mathbf{Z}} \leftarrow \theta\mathbf{U}\text{diag}(\mathbf{\Gamma})\mathbf{U}^{-1}$
```

To this end, we first calculate the summation of all the absolute eigen-values in $\mathbf{\Lambda}$ as $g = \sum_i |\mathbf{\Lambda}_{ii}|$. g is regarded as a constant. Then under fixed summation g, the maximal determinant value can be achieved iff each eigen-value is equal to g/n. Then we can calculate the gap $\mathbf{\Gamma} = \mathbf{m} - \mathbf{a}$ between $\mathbf{m} = (g/n, ..., g/n)$ and $\mathbf{a} = (\mathbf{\Lambda}_{11}, ..., \mathbf{\Lambda}_{nn})$. To avoid the gradient to be too aggressive, we assign a small step θ to $\mathbf{\Gamma}$. Thus the sub-gradient is written as:

$$\bar{\mathbf{Z}} = \theta\mathbf{U}\text{diag}(\mathbf{\Gamma})\mathbf{U}^{-1} \tag{4.13}$$

This procedure is given in Algorithm 4. The main overhead for both algorithms lies in the eigen-decomposition, resulting in similar computational efficiency.

4.4 Experiments

All the experiments are conducted on a laptop with 3.0GHz CPU and 16GB memory.

Datasets and metrics. Experiments involve both synthetic data and real-world images. The synthetic dataset is from the popular Random Graph Matching (Cho *et al.*, 2010). The real images include the CMU house sequence (Caetano *et al.*, 2006), Caltech-101/MSRC object matching (Cho *et al.*, 2010) and Pascal dataset (Leordeanu *et al.*, 2012). For evaluation, *accuracy*, *score* and *ratio* are evaluated, where *accuracy* measures the portion of correctly matched nodes with respect to all nodes, *score* represents the value of the objective function and *ratio* emphasizes the ratio between current objective value and the maximal one.

Compared Methods. Compared methods include Integer Projected Fixed Point (**IPFP**) (Leordeanu *et al.*, 2009), Graduated Assignment (**GAGM**) (Gold and Rangarajan, 1996), Reweighted Random Walk (**RRWM**) (Cho *et al.*, 2010), Binary-preserving Graph Matching (**BGM**) (Jiang *et al.*, 2017b), and two very recent state-of-the-art solvers: Branching Path Following Matching (**BPF**) (**??**) and Generalized Graph Matching (**GGM**) (Yu *et al.*, 2018b). For **GGM**, we select the setting of $\mathbf{GGM_{lp}}$ (refer (Yu *et al.*, 2018b) for more details). We term our method $\mathbf{DetGM}_1$ and $\mathbf{DetGM}_2$ for Sec. 4.3.2 and Sec. 4.3.2, respectively. In all experiments, all algorithms are initialized with a uniform matching.

4.4.1 Results on Synthetic Data

For each trial of the matching, a pair of graphs G^S and G^D is generated with n_{in} inlier nodes on both, where we let $n_{in} = 40$ in all tests ($n_{in} = 40$ is more challenging than a usual setting $n_{in} = 20$ such as in Cho *et al.* (2010)). The attribute $\mathbf{a}_{ij}^k$ with

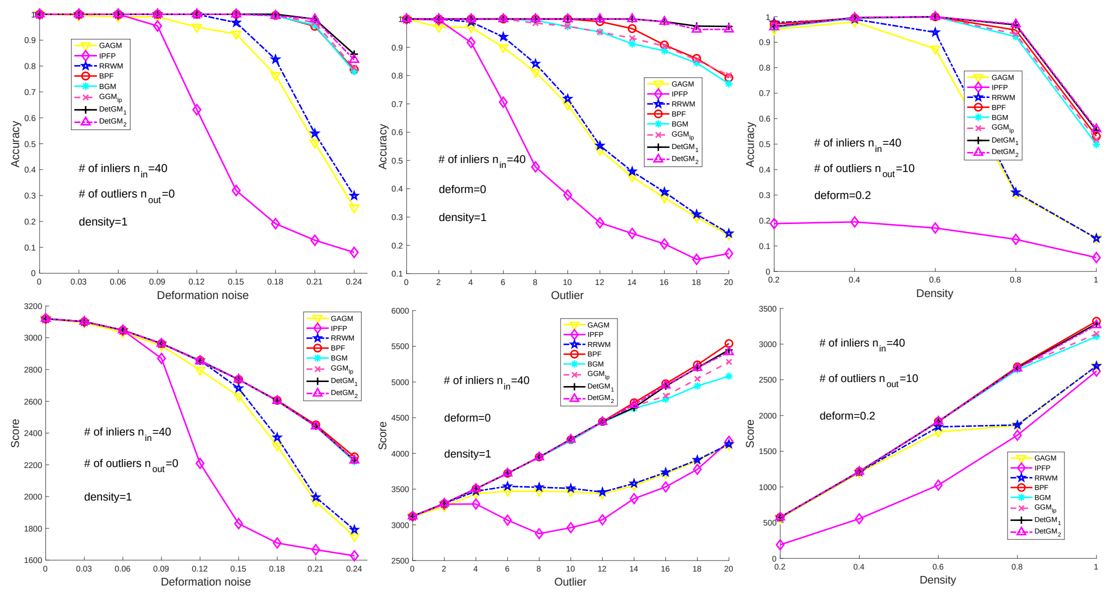


Figure 4.2: Accuracy and Objective Score on Synthetic Data by Varying Deformation, Outlier Count and Edge Density. Zoom-in for Better View.

Table 4.1: Accuracy (%) and Ratio on Caltech-101 Natural Images.

Method	GAGM	BGM	RRWM	BPF	GGM_{lp}	DetGM
Acc.	73.66	76.56	72.95	75.14	76.69	**77.44**
Ratio	0.933	0.970	0.946	1	0.972	0.985

$k \in \{S, D\}$ on edge is randomly generated from a multivariate uniform distribution ($i = j$ and $i \neq j$ correspond to node and edge attributes, respectively). Attributes on graph G^D are the copies of those on graph G^S with Gaussian noise $\epsilon \sim \mathcal{N}(0, \delta)$, namely $\mathbf{a}^D_{ij} = \mathbf{a}^D_{ij} + \epsilon$. For a pair of edges (i, j) in G^S and (a, b) in G^D, the corresponding affinity is calculated as $\mathbf{A}_{ij:ab} = \exp(-|\mathbf{a}^S_{ij} - \mathbf{a}^D_{ab}|^2/\sigma_s^2)$, where σ_s is a parameter and set to be 0.15 during all the synthetic tests. We conduct three types of empirical experiments with varying deformation noise ϵ, outlier count n_{out} and edge density ρ

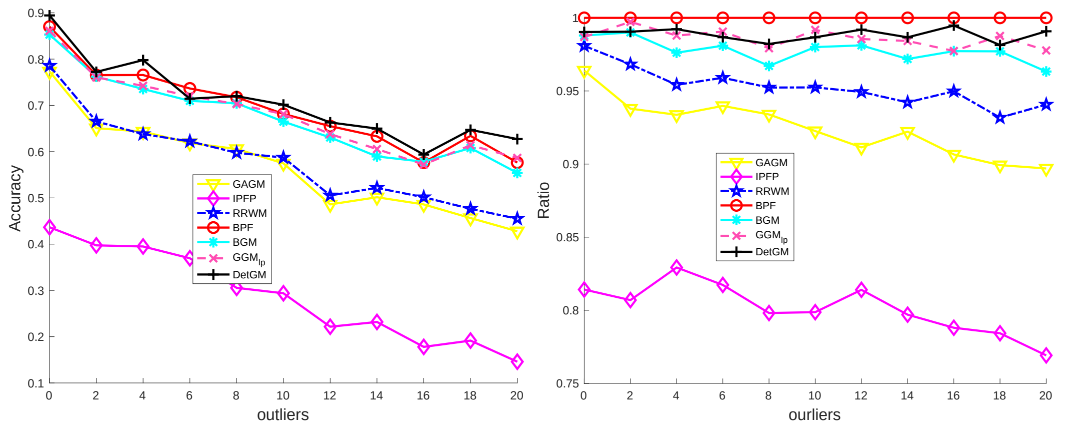


Figure 4.3: Accuracy and Score Ratio on Pascal Dataset by Varying the Number of Outliers.

Cho *et al.* (2010). In each experiment, we independently conduct 200 times of the single trial and report the average *accuracy* and *score* (objective value).

Figure 5.4 summarizes the synthetic experimental results. We see that DetGM$_1$ and DetGM$_2$ outperform all the selected algorithms in all settings. Particularly in the outlier test, the algorithms of DetGM significantly surpass both BPF and GGM$_{\mathrm{lp}}$ when there are massive outliers. We also observe that, though the objective *scores* of DetGM$_{1,2}$, BFP and GGM$_{\mathrm{lp}}$ are similar, there is a significant gap w.r.t. *accuracy*. This fact indicates again that the score may not necessarily reflect the true matching, which has also been pointed out in Yan *et al.* (2016); Yu *et al.* (2018b). DetGM$_{1,2}$, BPF and GGM$_{\mathrm{lp}}$ show similar performance in edge density test.

Remarks IPFP, GAGM and RRWM are in the line of *multiplication-based* updating strategy, where in iteration the product between the affinity and the previous solution contributes most to the current solution. Instead, DetGM$_{1,2}$, BGM, BPF and GGM$_{\mathrm{lp}}$ are *gradient-based*, which implies these algorithms employ a much more cautious update in each iteration other than multiplication-based ones. While the performance of the two categories of updating strategies hardly differentiates when

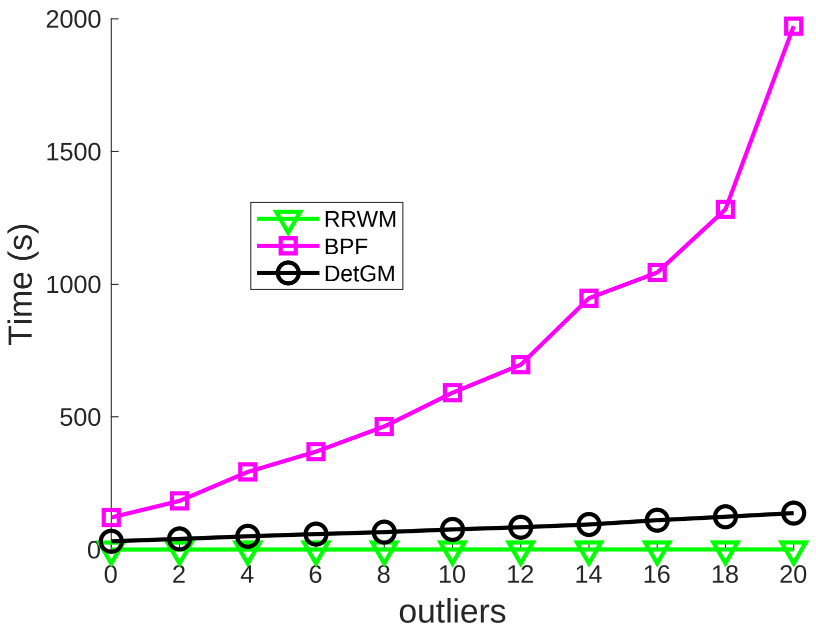


Figure 4.4: Average Running Time on Pascal Dataset by Varying the Number of Outliers.

the number of nodes is small (up to 20 for inliers and 10 for outliers), which has been verified by multiple previous works (Cho *et al.*, 2010; Jiang *et al.*, 2017b; Yu *et al.*, 2018b), it can be concluded from our experiments that gradient-based methods have more stable performance compared with multiplication-based ones *if number of nodes is large*. We infer the reason of such phenomenon is as follows. With increasing problem dimension, the affinity matrix $\mathbf{A}$ tends to deliver more complex (2nd-order) local behavior, thus an aggressive update strategy such as multiplication will likely jump over previous local region and diverge to an inconsistent solution path. Besides, the convergence criterion of multiplicative strategy is typically due to the gap of two consecutive objective scores. When $\mathbf{A}$ is complex, with a higher probability an ascending gradient may still exist even if the gap is small. The aforementioned factors will result in weak solution.

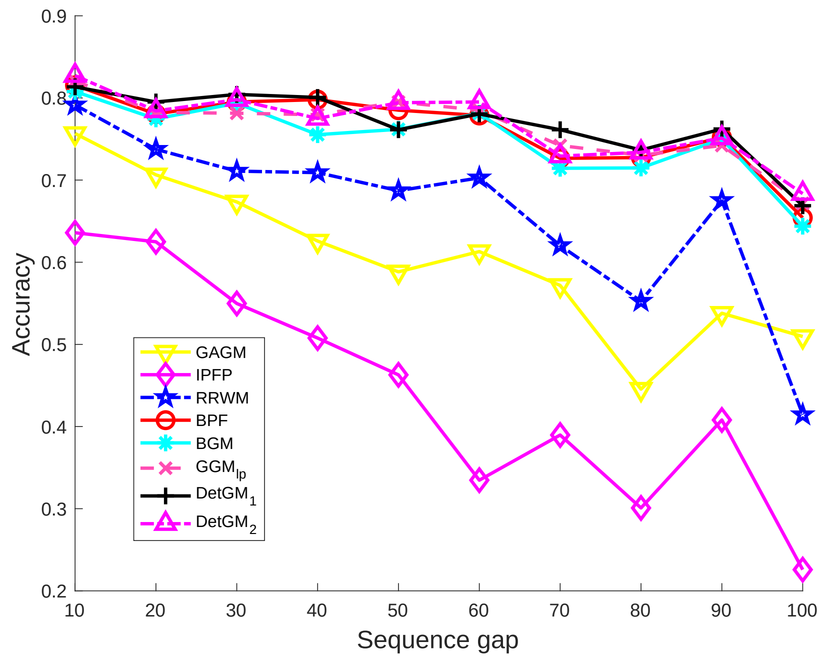
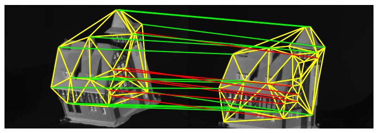


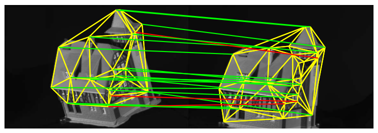
Figure 4.5: Accuracy on CMU House by Varying the Sequence Gap, for Two Graphs with 20 and 30 Nodes Respectively.

4.4.2 Results on Real-world Data

CMU House and Hotel Sequence. CMU house sequence has 110 images in total. We follow the widely adopted protocol in Cho *et al.* (2010) to conduct the test under varying sequence gap $(10, 20, ..., 100)$, resulting in 560 image pairs. For each image, 30 landmark points are manually labelled. And for each pair of images, 10 landmarks are randomly removed from the first image. The graph is established with Delaunay triangulation. The affinity is obtained as $\mathbf{K}_{ij:ab} = \exp(-|\mathbf{a}_{ij}^S - \mathbf{a}_{ab}^D|^2/\sigma_s^2)$, where $\mathbf{a}_{ij}^S$ measures the Euclidean distance between point i and j, and $\sigma_s^2 = 2500$ empirically.

Typical matching example with BPF, DetGM$_1$ and DetGM$_2$ are shown in Figure 4.6, and we report *accuracy* in Figure 4.5. It can be observed that in both settings of the proposed algorithm can reach out competitive performance against state-of-the-art algorithms BPF and GGM$_{\text{lp}}$, and significantly outperforms the resting algorithms


(a) BPF: **13/20**


(b) DetGM$_1$: **17/20**

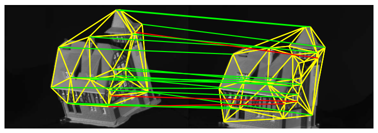
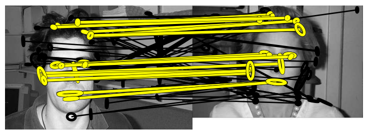
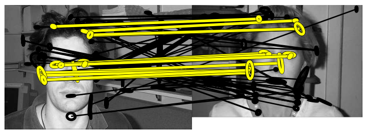

(c) DetGM$_2$: **17/20**

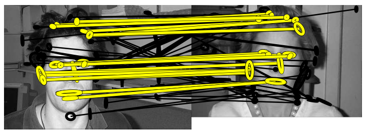
Figure 4.6: Matching Examples on CMU House Dataset with 20 and 30 Nodes for Each Graph Respectively. Correct Matchings Are in Green.

(a) Original image pair

(c) RRWM: **32/40**


(e) GGM$_{\text{lp}}$: **35/40**


(b) GAGM: **20/40**


(d) BPF: **35/40**

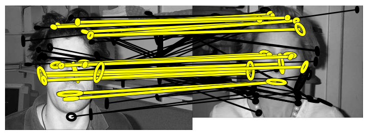

(f) DetGM: **36/40**

Figure 4.7: Examples on Caltech-101 Dataset with 40 Nodes on a Pair of Face Images. Correct Matchings Are in Yellow.

in *accuracy*. The two settings DetGM$_1$ and DetGM$_2$ show very similar performance in CMU house and synthetic tests, indicating both of the updating rules are valid and the corresponding solution paths do not diverge much. For the rest if the experiments, we only report the behavior of DetGM$_1$ due to their high similarity while abbreviating it as **DetGM**.

Caltech-101. It (Cho *et al.*, 2010) consists of 30 pairs of images from Caltech-101 (Fei-Fei *et al.*, 2007) and MSRC [1] . The features points are generated by MSER detector (Matas *et al.*, 2004) and each point is assigned its SIFT feature (Lowe, 1999). The

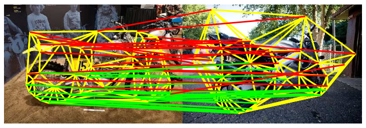
[1]http://research.microsoft.com/vision/cambridge/recognition/

(a) IPFP: **6/30**

(c) RRWM: **18/30**

(e) GGM_{lp}: **20/30**

(b) GAGM: **18/30**


(d) BPF: **20/30**

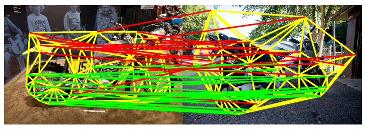

(f) DetGM: **24/30**

Figure 4.8: Examples on Pascal Dataset with 30 Inliers and 12 Outliers on a Pair of Motorbike Images. Correct Matchings are in Green.

candidate matchings are selected by comparing the feature distance with a threshold 0.6, which allows multiple correspondences for each feature. The dissimilarity between a candidate pair (i, j) and (a, b) is obtained with $\mathbf{K}_{ij:ab} = \max(50 - d_{ij:ab}, 0)$ where $d_{ij:ab}$ refers to the mutual projection error (Cho *et al.*, 2009).

Results are reported in Table 4.1 and matching examples are shown in Figure 4.7. One can see that our method outperforms the peers including most up-to-date BPF and GGM_{lp}.

Pascal Dataset. Pascal dataset (Leordeanu *et al.*, 2012) consists of 20 pairs of motorbike images and 30 pairs of car images collected from Pascal07 dataset (**?**). For each image pair, feature points and corresponding ground-truth correspondence are manually labelled. We follow the popular protocol (Zhou and Torre, 2012) to randomly select 0 to 20 outliers from the background to evaluate the algorithms under degradation. We follow Zhou and Torre (2012) to generate the graph and the corresponding affinity. For each node i, a feature $\mathbf{f}_i$ is assigned by taking its orientation of the normal vector at that node to the contour where the point was sampled. Then the node affinity between node i and j is computed as $\exp(-|\mathbf{f}_i - \mathbf{f}_j|)$. Delaunay triangulation is performed to obtain the graphs, and pairwise distance d_{ij} and absolute angle θ_{ij} are calculated on any valid edge (i, j). Thus the affinity between edge (i, j) in G^S and edge (a, b) in G^D is $\mathbf{K}_{ij:ab} = \exp(-(|d_{ij}^S - d_{ab}^D| + |\theta_{ij}^S - \theta_{ij}^D|))/2$.

Figure 4.8 and 4.4 presents an example of matching results on 6 counterparts and Figure 4.3 shows the quantitative results of the experiments. We also present time cost comparison against two typical solvers: RRWM and BPF. It can be seen that our algorithm achieves competitive performance against BPF and GGM_{lp} and outperforms them in some specific outlier settings. It should also be noted that, though BPF achieves high *accuracy*, the running speed of BPF is extremely slow. From Figure 4.4, we can find that in a typical setting with 20 outliers, BPF costs over $2,000$ seconds to finish one trial in average. However, our algorithm only need around 60 seconds in average. Though RRWM has higher efficiency, our algorithm still shows moderate computational cost in practical use.

4.5 Conclusion

To enable gradient-efficient continuation optimization, this chapter has presented a novel regularization technique for graph matching, as derived from the determinant analysis on the node matching matrix between two graphs. Theoretical property of our relaxation technique is studied and we also give some analysis on the geometric behavior compared to existing regularizers, which has rarely been considered. These findings are anticipated to bring about insights to other regularized objective with affine constraints. Extensive experiments are performed which show the state-of-the-art accuracy as well as efficiency of our method.

Chapter 5

COMBINATORIAL LEARNING OF GRAPH MATCHING

5.1 Problem Statement

For Eq. (1.1), a series of solvers haven been developed to solve graph matching problem (Leordeanu and Hebert, 2005; Cho *et al.*, 2010; Bernard *et al.*, 2018; Yan *et al.*, 2015a; Yu *et al.*, 2018b). All these methods are based on deterministic optimization, which are conditioned with pre-defined affinity matrix and no learning paradigm is involved. This fact greatly limits the performance and broad application w.r.t. different problem settings considering its NP-hard nature.

Recently, the seminal work namely deep graph matching (DGM) (Zanfir and Sminchisescu, 2018) is proposed to exploit the high capacity of deep networks for graph matching, which achieves state-of-the-art performance. This is in contrast to some early works which incorporate learning strategy separately in local stages (Caetano *et al.*, 2009; Cho *et al.*, 2013). On the other hand, Graph Convolutional Networks (GCN) (Kipf and Welling, 2017) brings about new capability on tasks over graph-like data, as it naturally integrates the intrinsic graph structure in a general updating rule:

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{5.1}$$

where $\hat{\mathbf{A}}$ is the normalized connectivity matrix. $\mathbf{H}^{(l)}$ and $\mathbf{W}^{(l)}$ are the features and weights at layer l, respectively. Node embedding is updated by aggregation from 1-neighboring nodes, which is akin to the convolution operator in CNN. By taking advantages of both DGM and GCN, Wang *et al.* (2019a) and Zhang and Lee (2019) incorporate permutation loss instead of displacement loss in (Zanfir and Sminchisescu, 2018), with notable improvement across both synthetic and real data.

Note that Eq. (1.1) involves both node and edge information, which exactly correspond to the diagonal and off-diagonal elements in $\mathbf{M}$, respectively. Edges can carry informative multi-dimensional attributes (namely weights) which are fundamental to *graph* matching. However existing embedding based graph matching methods (Wang *et al.*, 2019a; Xu *et al.*, 2019) are focused on the explicit modeling of node level features, whereby the edges are only used as topological node connection for message passing in GCN. Besides, edge attributes are neither well modeled in the embedding-free model (Zanfir and Sminchisescu, 2018) since the edge information is derived from the concatenation of node features. To our best knowledge, there is no deep graph matching method explicitly incorporating edge attributes. In contrast, edge attributes e.g. length and orientation are widely used in traditional graph matching models (Cho *et al.*, 2010; Yan *et al.*, 2015b; Yu *et al.*, 2018b) for constructing the affinity matrix $\mathbf{M}$. Such a gap shall be filled in the deep graph matching pipeline.

Another important consideration refers to the design of loss function. There are mainly two forms in existing deep graph matching works: i) displacement loss (Zanfir and Sminchisescu, 2018) similar to the use in optical flow estimation (Ren *et al.*, 2017); ii) the so-called permutation loss (Wang *et al.*, 2019a) involving iterative Sinkhorn procedure followed by a cross-entropy loss. Results in (Wang *et al.*, 2019a) show the latter is an effective improvement against the former regression based loss. However, we argue that the continuous Sinkhorn procedure (in training stage) is yet an unnatural approximation to Hungarian sampling (in testing stage) for discretization. If the network is equipped with a continuous loss function (e.g. cross-entropy), we argue that the training process will make a great "meaningless effort" to enforce some network output digits of the final matching matrix into binary and neglect the resting digits which might have notable impact on accuracy.

This part strikes an endeavor on the above two gaps and makes the following main contributions:

i) We propose a new approach for edge embedding via channel-wise operation, namely **channel-independent embedding** (**CIE**). The hope is to effectively explore the edge attribute and simulate the multi-head strategy in attention models (Veličković *et al.*, 2018) by decoupling the calculations parallel and orthogonal to channel direction. In fact, edge attribute information has not been considered in existing embedding based graph matching methods (Wang *et al.*, 2019a; Xu *et al.*, 2019).

ii) We devise a new mechanism to adjust the loss function based on the Hungarian method which is widely used for linear assignment problem, as termed by **Hungarian attention**. It resorts to dynamically generating sparse matching mask according to Hungarian sampling during training, rather than approximating Hungarian sampling with a differentiable function. As such, the Hungarian attention introduces higher smoothness against traditional loss functions to ease the training.

iii) The empirical results on three public benchmarks shows that the two proposed techniques are orthogonal and beneficial to existing techniques. Specifically, on the one hand, our CIE module can effectively boost the accuracy by exploring the edge attributes which otherwise are not considered in state-of-the-art deep graph matching methods; on the other hand, our Hungarian attention mechanism also shows generality and it is complementary to existing graph matching loss.

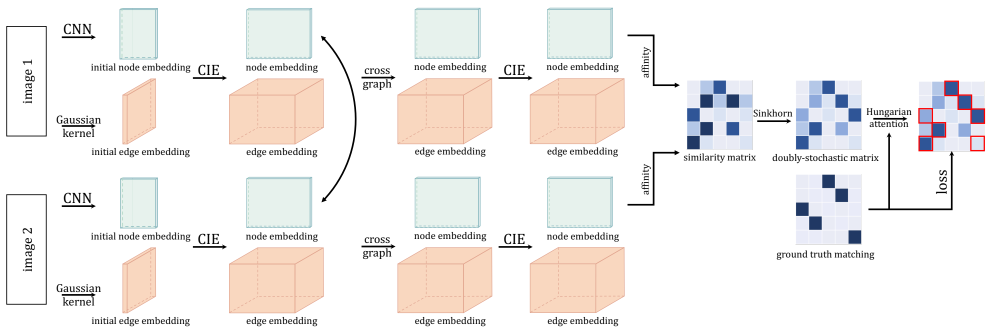


Figure 5.1: Architecture Overview of the Proposed Deep Graph Matching Networks That Consist of the Proposed Channel-independent Embedding and Hungarian Attention Layer over the Loss Function.

5.2 The Proposed Approach

5.2.1 Approach Overview

An overall structure of our approach is illustrated in Fig. 5.1. In line with (Wang *et al.*, 2019a), we employ VGG16 (Simonyan and Zisserman, 2014) to extract features from input images and bi-linearly interpolate the features at key points (provided by datasets). We concatenate lower-level (Relu4_2) and higher-level (Relu5_1) features to incorporate local and contextual information. For an image with k key points, the feature is denoted as $\mathbf{H} \in \mathbb{R}^{k \times d}$, where d is the feature dimension. Unless otherwise specified, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{k \times k}$ is consequentially constructed via Delaunay triangulation (Delaunay *et al.*, 1934), which is a widely adopted strategy to produce sparsely connected graph. To introduce more rich edge information, we also generate $k \times k$ m-dimensional edge features $\mathbf{E} \in \mathcal{R}^{m \times k \times k}$. E can be initialized with some basic edge information (e.g. length and angle and other attributes) or a commutative function $\mathbf{E}_{ij} = p(\mathbf{H}_i, \mathbf{H}_j) = p(\mathbf{H}_j, \mathbf{H}_i) \in \mathcal{R}^m$, where $\mathbf{H}_i$ refers to the feature of node i. Note for directed graph, the commutative property is not required.

The features $\mathbf{H}$ and $\mathbf{E}$, together with the adjacency $\mathbf{A}$, are then fed into GNN module. Pairs of features are processed in a Siamese fashion (Bromley *et al.*, 1994). Standard GCN's message passing rule simply updates node embedding as shown in Eq. (5.1). In contrast, each GNN layer in our model computes a new pair of node and edge embeddings simultaneously:

$$\mathbf{H}^{(l+1)} = f_i(\mathbf{H}^{(l)}, \mathbf{E}^{(l)}, \mathbf{A}; W_0^l), \quad \mathbf{E}^{(l+1)} = g(\mathbf{H}^{(l)}, \mathbf{E}^{(l)}, \mathbf{A}; W_1^l) \tag{5.2}$$

where W_0^l and W_1^l are the learnable parameters at layer l. The edge information is essential to provide structural feature enhancing graph matching. We initialize $\mathbf{H}^{(0)} = \mathbf{H}$ and $\mathbf{E}^{(0)} = \mathbf{E}$ in our setting. We will discuss the details of functions f and g in Sec. 5.2.2. Following state-of-the-art work (Wang *et al.*, 2019a), we also compute the cross-graph affinity followed by a column/row-wise softmax activation and a Sinkhorn layer (Adams and Zemel, 2011):

$$\mathbf{M}_{ij} = \exp\left(\tau \mathbf{H}_{(1)i}^{\top} \mathbf{\Lambda} \mathbf{H}_{(2)j}\right), \quad \mathbf{S} = \text{Sinkhorn}(\mathbf{M}) \tag{5.3}$$

Note here $\mathbf{M} \in \mathbb{R}^{k \times k}$ is the node-level similarity matrix encoding similarity between two graphs, differing from the edege-level affinity matrix $\mathbf{K}$ in Eq. 1.1. τ is the weighting parameter of similarity, $\mathbf{\Lambda}$ contains learnable parameters and $\mathbf{H}_{(1)i}$ is the node i's embedding from graph $\mathcal{G}_1$. The output $\mathbf{S} \in [0,1]^{k \times k}, \mathbf{S}\mathbf{1} = \mathbf{1}, \mathbf{S}^{\top}\mathbf{1} = \mathbf{1}$ is a so-called doubly-stochastic matrix. Here Sinkhorn$(\cdot)$ denotes the following update iteratively to project $\mathbf{M}$ into doubly stochastic polygon:

$$\mathbf{M}^{(t+1)} = \mathbf{M}^{(t)} - \frac{1}{n}\mathbf{M}^{(t)}\mathbf{1}\mathbf{1}^{\top} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\top}\mathbf{M}^{(t)} + \frac{1}{n^2}\mathbf{1}\mathbf{1}^{\top}\mathbf{M}^{(t)}\mathbf{1}\mathbf{1}^{\top} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\top} \tag{5.4}$$

The Sinkhorn layer is shown to be an approximation of Hungarian algorithm which produces discrete matching output (Kuhn, 1955). As there are only matrix multiplication and normalization operators involved in Sinkhorn layer, it is differentiable. In practice, Eq. (5.4) converges rapidly within 10 iterations for decades of nodes.

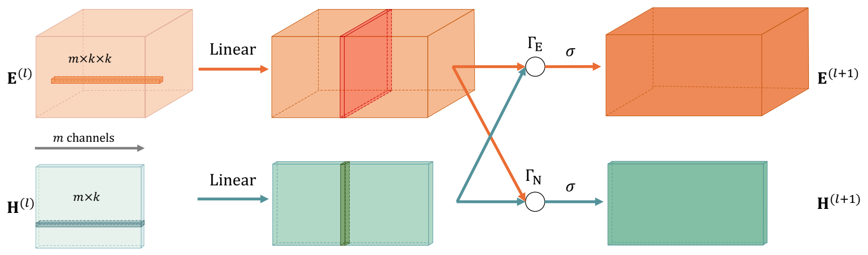


Figure 5.2: Illustration of the Proposed CIE Layer for Embedding Based Deep Graph Matching. The Operation "Linear" Refers to the Linear Mapping in Eq. (5.8).

Less iterations involved, more precise back-propagated gradients can be achieved. We employ a cross-graph node embedding strategy following (Wang *et al.*, 2019a):

$$\mathbf{H}_{(1)}^{(l)} = f_c\left(\mathrm{cat}(\mathbf{H}_{(1)}^{(l)}, \mathbf{S}\mathbf{H}_{(2)}^{(l)})\right), \quad \mathbf{H}_{(2)}^{(l)} = f_c\left(\mathrm{cat}(\mathbf{H}_{(2)}^{(l)}, \mathbf{S}^\top\mathbf{H}_{(2)}^{(l)})\right) \tag{5.5}$$

where f_c is a network and $\mathrm{cat}(\cdot,\cdot)$ is the concatenation operator. $\mathbf{H}_{(i)}$ is the node feature of graph i. This procedure seeks to merge similar features from another graph into the node feature in current graph. It is similar to the feature transfer strategy in (Aberman *et al.*, 2018) for sparse correspondence, which employs a feature merging method analogous to style transfer (Li *et al.*, 2017).

As Sinkhorn layer does not necessarily output binary digits, we employ Hungarian algorithm (Kuhn, 1955) to discretize matching output $\mathbf{S}$ in testing. The testing differs from the training due to the Hungarian discretization. We introduce a novel attention-like mechanism termed as **Hungarian attention**, along with existing loss functions (will be detailed in Sec. 5.2.3). The final training loss is as follows, where $\mathbf{S}^{\mathrm{G}}$ and $\mathcal{H}$ correspond to binary true matching and Hungarian attention loss.

$$\min \mathcal{H}(\mathbf{S}, \mathbf{S}^{\mathrm{G}}) \tag{5.6}$$

5.2.2 Channel-independent Embedding

We detail the updating rule in Eq. (5.2). We propose a method to merge edge features into node features and perform matching on nodes. Edge information acts an important role in modeling relational data, whereby such relation can be complex thus should be encoded with high-dimensional feature. To this end, Gilmer *et al.* (2017) introduce a general embedding layer, which takes node and edge features and outputs a message to node v, then fuses the message and the current embedding:

$$\mathbf{m}_v^{(l)} = \sigma\left(\sum_{w\in\mathcal{N}_v} f_t\left(\mathbf{E}_{vw}\right)\mathbf{H}_w^{(l)} + \mathbf{W}^{(l)}\mathbf{H}^{(l)}\right), \quad \mathbf{H}_v^{(t+1)} = u_t\left(\mathbf{H}_v^{(t)}, \mathbf{m}_v^{(l)}\right) \tag{5.7}$$

where $\mathbf{E}_{vw}$ is the feature corresponding to edge (v, w). In the realization of Eq. (5.7) (Gilmer *et al.*, 2017), $\mathbf{m}_v^{(l)}$ and $\mathbf{H}_v^{(l)}$ are fed to GRU (Cho *et al.*, 2014) as a sequential input. There are several variants which take into account specific tasks (Li *et al.*, 2016; Schütt *et al.*, 2017; Chen *et al.*, 2019). Among these, Li *et al.* (2016) generates a transformation matrix for each edge and Schütt *et al.* (2017) resorts to merge embedding via fully connected neural networks. While edge-wise merging is straightforward, the representation ability is also limited. On the other hand, fully connected merging strategy will result in high computational cost and instability for back-propagation. To address these issues, we propose to merge embedding in a channel-wise fashion, which is termed as Channel-Independent Embedding (**CIE**). Concretely, the updating rule is written as:

$$\mathbf{H}_v^{(l+1)} = \sigma\left(\sum_{w\in\mathcal{N}_v} \underbrace{\Gamma_{\mathrm{N}}\left(\mathbf{W}_1^{(l)}\mathbf{E}_{vw}^{(l)} \circ \mathbf{W}_2^{(l)}\mathbf{H}_w^{(l)}\right)}_{\text{channel-wise operator/function}}\right) + \sigma\left(\mathbf{W}_0^{(l)}\mathbf{H}_v^{(l)}\right) \tag{5.8}$$

$$\mathbf{E}_{vw}^{(l+1)} = \sigma\left(\mathbf{W}_1^{(l)}\mathbf{E}_{vw}^{(l)}\right) \tag{5.9}$$

where $\Gamma_{\mathrm{N}}(\cdot \circ \cdot)$ is a channel-wise operator/function (above the underbrace), and it performs calculation per-channel and the output channel dimension is the same as input. The second $\sigma(\cdot)$ term is the message a node passes to itself, which is necessary in keeping the node information contextually consistent through each CIE layer. In this fashion, CIE is thus a procedure to aggregate node and edge embedding in each channel independently, which requires the dimensions of node $(\mathbf{W}_2^{(l)}\mathbf{H}_w^{(l)})$ and edge $(\mathbf{W}_1^{(l)}\mathbf{E}_{vw}^{(l)})$ representations to be equal. Similarly, we also propose an corresponding updating rule of edge embedding by substituting Eq. (5.9):

$$\mathbf{E}_{vw}^{(l+1)} = \sigma\left(\Gamma_{\mathrm{E}}\left(\mathbf{W}_1^{(l)}\mathbf{E}_{vw}^{(l)} \circ h\left(\mathbf{H}_v^{(l)}, \mathbf{H}_w^{(l)}\right)\right)\right) + \sigma\left(\mathbf{W}_1^{(l)}\mathbf{E}_{vw}^{(l)}\right) \tag{5.10}$$

where $h(\cdot,\cdot)$ is commutative $h(\mathbf{X},\mathbf{Y}) = h(\mathbf{Y},\mathbf{X})$. Eq. (5.10) is supplementary to Eq. (5.8).

Fig. 5.2 shows a schematic diagram of CIE layer, which is motivated from two perspectives. First, CIE is motivated by counterparts in CNN (Qiu *et al.*, 2017; Tran *et al.*, 2018) which decouple a 3D convolution into two 2D ones (e.g. a $3 \times 3 \times 3$ convolution can be decomposed to a $1 \times 3 \times 3$ and a $3 \times 1 \times 1$ convolutions). In this sense, the number of parameters can be significantly reduced. As shown in Fig. 5.2, node and edge embedding is first manipulated along the channel direction via a linear layer, then operated via Γ_{N} and Γ_{E} orthogonal to the channel direction. Instead of merging node and edge as a whole, CIE layer decouples it into two operations. Second, CIE is also motivated by the triumph of multi-head structure (e.g. graph attention (Veličković *et al.*, 2018)), the key of which is to conduct unit calculation multiple times and concatenate the results. Multi-head proved effective to further improve the performance since it is capable of capturing information at different scales or aspects. Traditional neural node-edge message passing algorithms (Gilmer *et al.*, 2017; Li *et al.*, 2016; Schütt *et al.*, 2017) typically produce a unified transformation matrix for all the channels. On the other hand, in Eq. (5.8) (5.9) and (5.10), one can consider that the basic operator in each channel is repeated d times in a multi-head fashion.

The cross-channel information exchange, as signified in Eq. (5.8) (5.9) and (5.10), only happens before the channel-wise operator (i.e. weights $\mathbf{W}_i^{(l)}$ as the cross-channel matrices). The main difference between CIE and traditional multi-head approaches e.g. (Veličković *et al.*, 2018) is that CIE assumes the channel-independence of two embedded features (node and edge), while traditional ones only take one input under head-independence assumption.

5.2.3 Hungarian Attention Mechanism

For most graph matching algorithms, the output is in a continuous domain. Though there are some alternatives that deliver discrete solutions by adding more constraints or introducing numerical continuation (Zhou and Torre, 2012; Yu *et al.*, 2018b), the main line of methods is to incorporate a sampling procedure (e.g. winner-take-all and Hungarian). Among them, the Hungarian algorithm (Kuhn, 1955) is a widely adopted, for its efficiency and theoretical optimality.

However, the Hungarian algorithm incurs a gap between training (loss function) and testing stages (Hungarian sampling). We compare the permutation loss (Wang *et al.*, 2019a) for concrete analysis:

$$\mathcal{L}_{\mathrm{CE}} = -\sum_{i\in\mathcal{G}_1, j\in\mathcal{G}_2} \left(\mathbf{S}_{ij}^{\mathrm{G}} \log \mathbf{S}_{ij} + \left(1 - \mathbf{S}_{ij}^{\mathrm{G}}\right) \log\left(1 - \mathbf{S}_{ij}\right)\right) \tag{5.11}$$

Note Eq. (5.11) is an element-wise version of binary cross-entropy. During training, this loss tends to drag the digits in $\mathbf{S}$ into binary format and is likely trapped to local optima. This is because this loss will back-propagate the gradients of training samples that are easy to learn in the early training stage. In later iterations, this loss is then hard to give up the digits that have become binary. In fact, the similar phenomenon is also investigated in the focal loss (Lin *et al.*, 2017) in comparison to the traditional cross-entropy loss. During the testing stage, however, the Hungarian

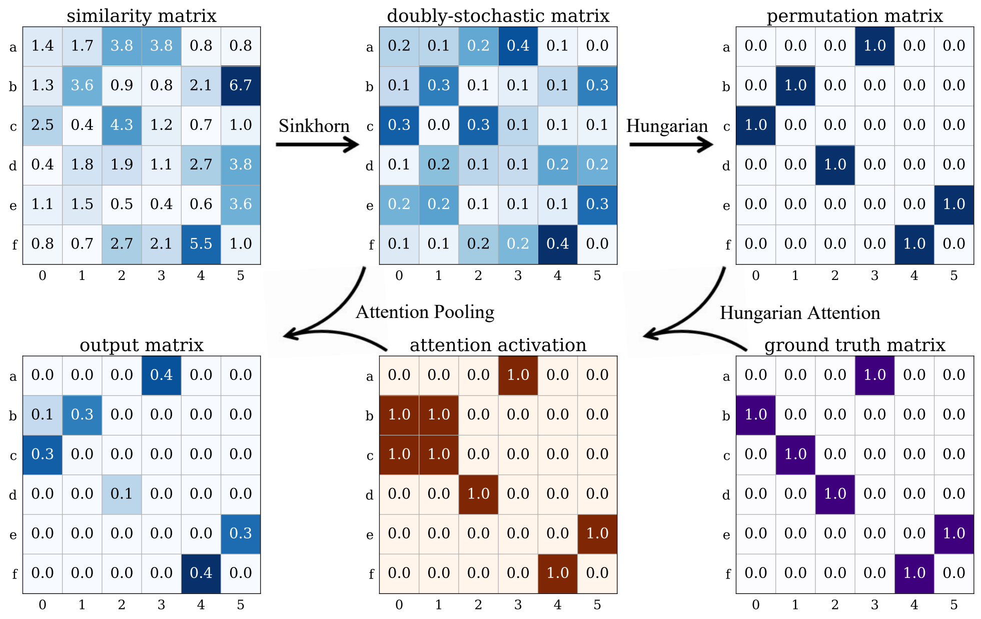


Figure 5.3: A Working Example Illustrating Our Proposed Hungarian Attention Pipeline Starting from Similarity Matrix. Sinkhorn Algorithm Solves Similarity Matrix into a Doubly-stochastic Matrix in a Differentiable Way. A Discrete Permutation Matrix Is Further Obtained via Hungarian Algorithm. Our Proposed Hungarian Attention, Taking the Ground Truth Matching Matrix into Account, Focuses on the "important" Digits Either Labeled True or Being Mis-classified. The Output Matrix Is Obtained by Attention Pooling from Doubly-stochastic Matrix, Where We Compute a Loss on It.

algorithm has no preference on the case if digits in $\mathbf{S}$ are close to $0-1$ or not. It binarizes $\mathbf{S}$ anyway. Therefore, the effort of Eq. (5.11) to drag $\mathbf{S}$ into binary might be meaningless.

This issue is likely to be solved by integrating Hungarian algorithm during the training stage. Unfortunately, Hungarian algorithm is undifferentiable and its behavior is difficult to mimic with a differentiable counterpart. In this chapter, instead of finding a continuous approximation of Hungarian algorithm, we treat it as a black box and dynamically generate network structure (sparse link) according to its output. Concretely, the sparse link is calculated as:

$$\mathbf{Z} = \text{Atten}\left(\text{Hungarian}(\mathbf{S}), \mathbf{S}^{\text{G}}\right) = \mathcal{P} \cup \mathcal{Q} \tag{5.12}$$

where the attention mechanism Atten is fulfilled by an element-wise "logic OR" function. Fig. 5.3 shows an example of Hungarian attention procedure, and Eq. (5.12) highlights the most contributing digit locations: positive digits $\mathcal{P} = \mathbf{S}$ where Hungarian agrees with the ground-truth; negative digits $\mathcal{Q} = \text{Hungarian}(\mathbf{S}) \setminus \mathbf{S}^{\text{G}}$ where Hungarian differs from ground-truth. While GT (positive digits) naturally points out the digits that must be considered, negative ones indicate the digits that most hinder the matching (most impeding ones among all mis-matchings). Thus we need only minimize the loss at $\mathbf{Z}$, without considering the rest of digits. As we note that this mechanism only focuses on a small portion of the matching matrix which is analogous to producing hard attention, we term it **Hungarian attention**. Now that with the attention mask $\mathbf{Z}$, the Hungarian attention loss becomes:

$$\mathcal{H}_{\text{CE}} = -\sum_{i \in \mathcal{G}_1, j \in \mathcal{G}_2} \mathbf{Z}_{ij} \left(\mathbf{S}^{\text{G}}_{ij} \log \mathbf{S}_{ij} + \left(1 - \mathbf{S}^{\text{G}}_{ij}\right) \log\left(1 - \mathbf{S}_{ij}\right)\right) \tag{5.13}$$

Note that Hungarian attention mechanism can also be applied to other loss functions once the matching score is calculated in an element-wise fashion. Our experiment also studies Hungarian attention loss when casted on focal loss (Lin *et al.*, 2017) and a specifically designed margin loss.

Finally we give a brief qualitative analysis on why Hungarian attention can improve matching loss. As discrete graph matching problem is actually built upon Delta function over permutation vertices (1 at ground-truth matching and 0 otherwise) (Yu *et al.*, 2018b), learning of graph matching with permutation loss is actually to approximate such functions with continuous counterparts. Unfortunately, more precise

Table 5.1: Accuracy on Pascal VOC (Best in Bold). White and Gray Background Refer to Results on Testing and Training, Respectively.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	Ave
GMN-D	31.9	47.2	51.9	40.8	68.7	72.2	53.6	52.8	34.6	48.6	72.3	47.7	54.8	51.0	38.6	75.1	49.5	45.0	83.0	86.3	55.3
GMN-P	31.1	46.2	58.2	45.9	70.6	76.4	61.2	61.7	35.5	53.7	58.9	57.5	56.9	49.3	34.1	77.5	57.1	53.6	83.2	88.6	57.9
GAT-P	46.4	60.5	60.9	51.8	79.0	70.9	62.7	70.1	39.7	63.9	66.2	63.8	65.8	62.8	39.5	82.0	66.9	50.1	78.5	90.3	63.6
GAT-H	47.2	61.6	63.2	53.3	79.7	70.1	65.3	70.5	38.4	64.7	62.9	65.1	66.2	62.5	41.1	78.8	67.1	61.6	81.4	91.0	64.6
EPN-P	47.6	65.2	62.2	52.7	77.8	69.5	63.4	69.6	37.8	62.8	63.6	63.9	64.6	61.9	39.9	80.5	66.7	45.5	77.6	90.6	63.2
PIA-D	39.7	57.7	58.6	47.2	74.0	74.5	62.1	66.6	33.6	61.7	65.4	58.0	67.1	58.9	41.9	77.7	64.7	50.5	81.8	89.9	61.6
PIA-P	41.5	55.8	60.9	51.9	75.0	75.8	59.6	65.2	33.3	65.9	62.8	62.7	67.7	62.1	42.9	80.2	64.3	59.5	82.7	90.1	63.0
PCA-P	40.9	55.0	65.8	47.9	76.9	**77.9**	63.5	67.4	33.7	65.5	63.6	61.3	68.9	62.8	44.9	77.5	67.4	57.5	**86.7**	90.9	63.8
PCA-H	49.8	60.7	63.9	52.6	79.8	72.5	63.8	71.2	38.4	62.5	71.7	65.4	66.6	62.5	40.5	84.7	66.1	47.9	80.5	91.1	64.6
PCA+-P	46.6	61.0	62.3	53.9	78.2	72.5	64.4	70.5	39.0	63.5	**74.8**	65.2	65.0	61.6	40.8	83.2	67.1	50.5	79.6	91.6	64.6
CIE$_2$-P	50.9	65.5	68.0	57.0	81.0	75.9	70.3	73.4	**41.1**	66.7	53.2	68.3	68.4	63.5	45.3	84.8	69.7	57.2	79.8	91.6	66.9
CIE$_2$-H	51.2	68.4	69.5	57.3	82.5	73.5	69.5	74.0	40.3	67.8	60.0	69.7	70.3	65.1	44.7	86.9	70.7	57.3	84.2	92.2	67.4
CIE$_1$-P	**52.1**	**69.4**	69.9	**58.9**	80.6	76.3	**71.0**	**74.2**	**41.1**	68.0	60.4	69.7	70.7	65.1	**46.1**	85.1	**70.4**	61.6	80.7	91.7	68.1
CIE$_1$-H	51.2	69.2	**70.1**	55.0	**82.8**	72.8	69.0	**74.2**	39.6	**68.8**	71.8	**70.0**	**71.8**	**66.8**	44.8	**85.2**	69.9	**65.4**	85.2	**92.4**	**68.9**
PCA-P	75.8	99.2	83.3	74.7	98.7	96.3	74.3	87.8	80.9	85.7	100.0	83.7	83.8	98.7	66.5	99.1	80.7	99.7	98.2	97.0	88.2
CIE$_1$-P	56.5	84.0	73.5	58.0	91.5	81.1	67.8	76.8	46.4	72.2	98.0	73.9	73.6	77.9	46.1	94.8	72.7	93.6	93.7	91.6	76.2
CIE$_1$-H	59.4	88.1	75.9	58.0	94.3	81.9	69.4	78.9	49.5	78.2	99.7	78.1	78.0	82.1	47.4	95.8	75.7	97.6	96.0	91.1	78.7

approximation to Delta function will result in higher non-smoothness, as discussed in Yu *et al.* (2018b). For highly non-smooth objective, the network is more likely trapped at local optima. Hungarian attention, however, focuses on a small portion of the output locations, thus does not care about if most of the output digits are in $\{0, 1\}$. In this sense, Hungarian attention allows moderate smoothness of the objective, thus optimizer with momentum is likely to avoid local optima.

5.3 Experiments

Experiments are conducted on three benchmarks widely used for learning-based graph matching: CUB2011 dataset (Welinder *et al.*, 2010) following the protocol in (Choy *et al.*, 2016), Pascal VOC keypoint matching (Everingham *et al.*, 2010b;

Bourdev and Malik, 2009) which is challenging and Willow Object Class dataset (Cho *et al.*, 2013). Mean matching accuracy is adopted for evaluation:

$$\text{Acc} = \frac{1}{k} \sum_{i \in \mathcal{G}_1, j \in \mathcal{G}_2} \text{AND}\left(\text{Hungarian}(\mathbf{S})_{ij}, \mathbf{S}^{\text{G}}_{ij}\right) \tag{5.14}$$

The algorithm abbreviation is in the form "X-Y", where "X" and "Y" refer to the network structure (e.g. CIE) and loss function (e.g. Hungarian attention loss), respectively. Specifically, **D**, **P** and **H** correspond to displacement used in (Zanfir and Sminchisescu, 2018), permutation as adopted in (Wang *et al.*, 2019a) and Hungarian attention over permutation loss devised by this chapter, respectively.

Peer methods. We compare our method with the following selected counterparts: 1) **HARG** (Cho *et al.*, 2013). This shallow learning method is based on hand-crafted feature and Structured SVM; 2) **GMN** (Zanfir and Sminchisescu, 2018). This is a seminal work incorporating graph matching and deep learning, and the solver is upon spectral matching (Leordeanu and Hebert, 2005). While the loss of this method is displacement loss, we also report the results of GMN by replacing its loss with permutation loss (**GMN-P**); 3) **PIA/PCA** (Wang *et al.*, 2019a). PCA and PIA correspond to the algorithms with and without cross-graph node embedding, respectively. Readers are referred to Wang *et al.* (2019a) for more details; We further replace the GNN layer in our framework with: 4) **GAT** (Veličković *et al.*, 2018). Graph attention network is an attention mechanism on graphs, which reweights the embedding according to attention score; 5) **EPN** (Gong and Cheng, 2019). This method exploits multi-dimensional edge embedding and can further be applied on directed graphs. The edge dimension is set to 32 in our experiments. Finally, we term our network structure **CIE** for short. To investigate the capacity of edge embedding update, we also devise a version without edge embedding, in which connectivity is initialized as reciprocal of the edge length then normalized, rather than **A**. This model is called **PCA+** since the node embedding strategy follows PCA.

Implementation details. As the node number of each graph might vary, we add dummy nodes for each graph pair such that the node number reaches the maximal graph size in a mini-batch in line with the protocol in (Wang *et al.*, 2019a). In either training or testing stages, these dummy nodes will not be updated or counted. The activation function in Eq. (5.8) (5.9) and (5.10) is set as Relu in all experiments. Specifically, the node and edge embedding is implemented by:

$$\mathbf{H}_{\cdot q}^{(l+1)} = \sigma\left(\left(\mathbf{A} \odot \left(\mathbf{W}_1^{(l)}\mathbf{E}^{(l)}\right)_{\cdot q}\right)\left(\mathbf{W}_2^{(l)}\mathbf{H}^{(l)}\right)_{\cdot q}\right) + \sigma\left(\left(\mathbf{W}_0^{(l)}\mathbf{H}^{(l)}\right)_{\cdot q}\right) \tag{5.15a}$$

$$\mathbf{E}_{\cdot q}^{(l+1)} = \sigma\left(\left|\left(\mathbf{W}_0^{(l)}\mathbf{H}^{(l)}\right)_{\cdot q} \ominus \left(\mathbf{W}_0^{(l)}\mathbf{H}^{(l)}\right)_{\cdot q}^{\top}\right| \odot \mathbf{E}_{\cdot q}^{(l)}\right) + \sigma\left(\left(\mathbf{W}_1^{(l)}\mathbf{E}^{(l)}\right)_{\cdot q}\right) \tag{5.15b}$$

where $\odot$ and $\ominus$ refer to element-wise product and pairwise difference, respectively. $\mathbf{H}_{\cdot q}$ is the qth channel of $\mathbf{H}$. In **CIE**$_1$ setting, only node-level merging Eq. (5.15a) is considered and the edge feature is updated as Eq. (5.9). In **CIE**$_2$ setting, we also replace the edge update Eq. (5.10) with Eq. (5.15b). Note edge embedding is used in both **CIE**$_1$ and **CIE**$_2$ and note **PCA-H** can be regarded as the pure node embedding version of our approach. The edge feature is initiated as reciprocal of the edge length. For training, batch size is set to 8. We employ SGD optimizer (Bottou, 2010) with momentum 0.9. Two CIE layers are stacked after VGG16.

5.3.1 Synthetic Test

Synthetic graphs are generated for training and testing following the protocol in Cho *et al.* (2010). Specifically, K_{pt} keypoints are generated for a pair of graphs with a 1024-dimensional random feature for each node, which is sampled from uniform distribution $\mathcal{U}(-1, 1)$. Disturbance is also applied to graph pairs including: Gaussian node feature noise from $\mathcal{N}(0, \sigma_{ft}^2)$; random affine transformation $\begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$ with $s \sim \mathcal{U}(0.8, 1.2), \theta \sim \mathcal{U}(-60, 60), t_x, t_y \sim \mathcal{U}(-10, 10)$ followed by Gaussian coordinate position noise $\mathcal{N}(0, \sigma_{co}^2)$. By default we assign $K_{pt} = 25, \sigma_{ft} = 1.5, \sigma_{co} = 5$. Two graphs share the same structure. We generate 10 random distributions for each test. Results are shown in Fig. 5.4. The performance of PCA and CIE is reported. We see our method significantly outperformed PCA. It can further be noticed that Hungarian attention can help to achieve an even higher accuracy. Readers are referred to Wang *et al.* (2019a) for some other results on synthetic test.

However, we also notice that the way to generate synthetic graphs is much different from the distribution of real-world data. For real-world data, on one hand, there is strong correlation on the neighboring node features. This is the reason why the message passing from nearby node features works. However, the features of synthetic data are randomly generated and there is no correlation between neighboring node features. Therefore, message passing mechanism is not very effective to reveal the relation or pattern among local nodes for synthetic data. On the other hand, features of real-world data typically lie on a manifold embedded in high dimensional space, hence is low dimensional. However, randomly generated features will span the whole space and show no patterns.

Taking into account the aforementioned factors, we believe there is a demand for a novel strategy to generate more reasonable synthetic data. This can be one of the future works.

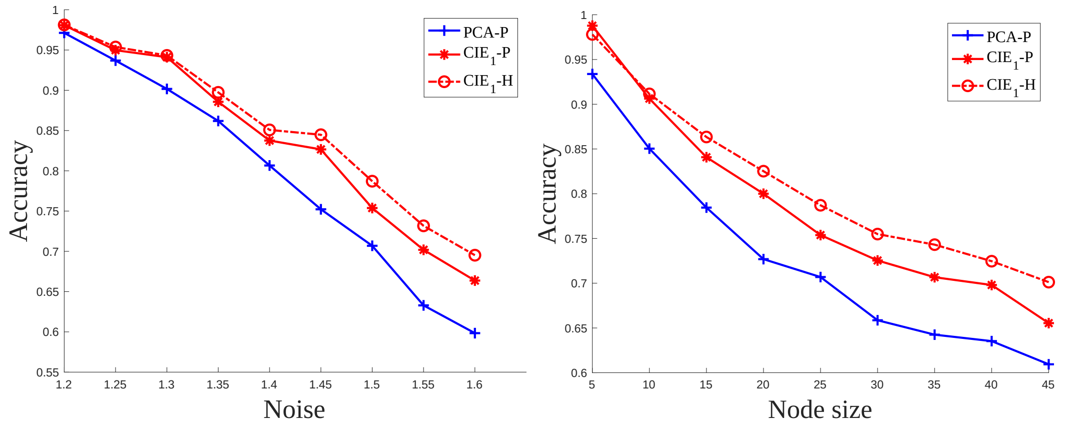
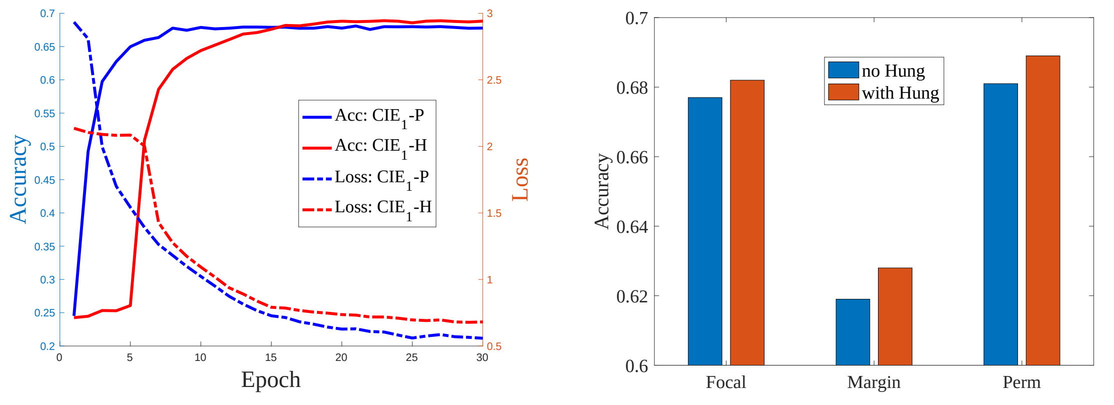


Figure 5.4: Results on Synthetic Test Where Two Different Loss Functions Are Compared in Ablative Study.

5.3.2 Results on CUB2011

CUB2011 consists of 11,788 images from 200 kinds of birds with 15 annotated parts. We randomly sample image pairs from the dataset following the implementation released by Choy *et al.* (2016). We do not use the pre-alignment of poses during testing, because their alignment result is not publicly available. Therefore, there exists significant variation in pose, articulation and appearance across images, in both training and testing phase. Images are cropped around bounding box and resized to 256×256 before fed into the network. Instead of evaluating the performance in a retrieval fashion (Zanfir and Sminchisescu, 2018), we directly evaluate the matching accuracy since the semantic key-points are pre-given. We test two settings: 1) *intra-class*. During training, we randomly sample images, with each pair sampled from the same category (out of 200 bird categories). In testing, 2,000 image pairs (100 pairs for each category) are sampled; 2) *cross-class*. We analogously sample image pairs without considering the category information and 5,000 randomly sampled image pairs are employed for testing. While the first setting is for a class-aware situation, the second setting is considered for testing the class-agnostic case. Results are shown in



(a) Accuracy/Loss vs. Training Epoch. (b) Ablation Study by Hungarian Attention.

Figure 5.5: Performance Study on Pascal Voc. Note in (a) the Loss Is Calculated on All Matching Digits for Both CIE$_1$-p and CIE$_1$-h. Note Around 10th Epoch, the Accuracy of CIE$_1$-p Almost Reaches the Highest, but the Loss Keeps Descending until 30th Epoch. This Indicates That in Most of the Latter Epochs, P-loss Performs “meaningless” Back-propagation to Drag the Output to Binary. H-loss, by Accommodating Smoothness, Can Emphasize Most Contributing Digits and Achieves Higher Accuracy.

Table 5.3.

We see our method surpasses all the competing methods in terms of matching accuracy. Besides, almost all the selected algorithms can reach over 90% accuracy, indicating that this dataset contains mostly “easy” learning samples. In this case, the Hungarian attention can slightly improve the performance since easy gradients agree with descending trend of the loss on the whole dataset.

5.3.3 Results on Pascal VOC

The Pascal VOC dataset with Key-point annotation (Bourdev and Malik, 2009) contains 7,020 training images and 1,682 testing images with 20 classes in total. To the best of our knowledge, this is the largest and most challenging dataset for graph matching in computer vision. Each image is cropped around its object bounding box and is resized to 256×256. The node size of this dataset varies from 6 to 23 and there are various scale, pose and illumination perturbations. Experimental results are summarized in Table 6.1. We see in either setting, CIE significantly outperforms all peer algorithms. Specifically, CIE_1-H achieves the best performance and has 0.8% improvement w.r.t. average accuracy over CIE_1-P. For each class, CIE_1-H and CIE_1-P carve up most of the top performance. We also note that CIE_1-H has a close performance on "table" compared with GMN-D. Since P-loss is naturally not as robust as D-loss on symmetric objects, P-loss showed great degradation over D-loss on "table" (as discussed in (Wang *et al.*, 2019a)). However, with the help of Hungarian link, H-loss can maintain relatively high accuracy despite natural flaw of P-loss. This observation indicates that H-loss can focus on "difficult" examples. We also note that CIE_1 produces better results against CIE_2, which implies that updating edge embedding is less effective compared to a singleton node updating strategy. We can also see from Table 6.1 that PCA-P has much higher performance on *training samples* than CIE_1-H, which is to the contrary of the result on testing samples. This might indicate that PCA-P overfits the training samples.

Accuracy/loss vs. training epoch. We further show the typical training behavior of P-loss and H-loss on Pascal VOC dataset in Fig. 5.5. 30 epochs are involved in a whole training process. Accuracy is evaluated on *testing samples* after each epoch while loss is the average loss value within each epoch. In the early training stage, the loss of CIE_1-P immediately drops. On the other hand, CIE_1-H hesitates for several epochs to find the most effective descending direction. On the late stage, we observe that even though P-loss (Eq. (5.11)) calculates much more digits than H-loss (Eq. (5.13)), the loss values are opposite. This counter-intuitive fact strongly indicates that P-loss makes meaningless effort, which is not helpful to improve the performance, at late stage. The proposed H-loss, on the other hand, is capable of avoiding easy but meaningless gradients.

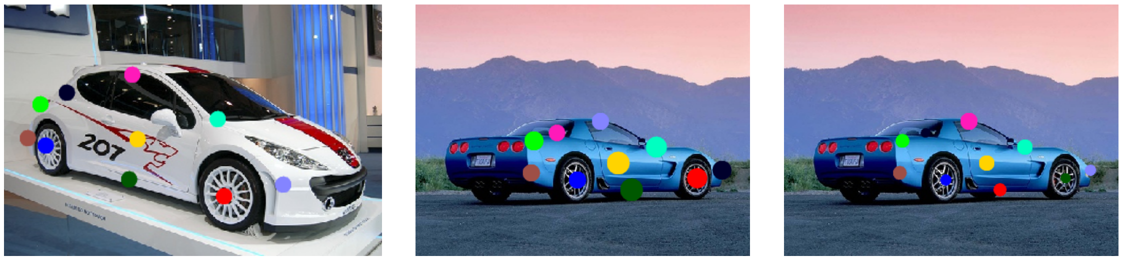
Effect of Hungarian attention mechanism. We also conduct experiments to show the improvement of Hungarian attention over several loss functions (with and without Hungarian attention): Hungarian attention is applied on Focal loss (Focal) (Lin *et al.*, 2017) as:

$$\mathcal{L}_{\text{focal}} = \begin{cases} -\alpha \mathbf{Z}_{ij}(1-\mathbf{S}_{ij})^{\gamma}\log(\mathbf{S}_{ij}), & \mathbf{S}^{\text{G}}_{ij}=1 \\ -(1-\alpha)\mathbf{Z}_{ij}\mathbf{S}^{\gamma}_{ij}\log(1-\mathbf{S}_{ij}), & \mathbf{S}^{\text{G}}_{ij}=0 \end{cases} \tag{5.16}$$

where controlling parameters $\alpha = 0.75$ and $\gamma = 2$ in our setting. We also design a margin loss (Margin) with Hungarian attention under a max-margin rule. Note we insert the Hungarian attention mask $\mathbf{Z}_{ij}$ into Eq. (5.16) and Eq. (5.17) based on the vanilla forms.

$$\mathcal{L}_{\text{margin}} = \begin{cases} \mathbf{Z}_{ij} \times \max(1-\mathbf{S}_{ij}-\beta, 0), & \mathbf{S}^{\text{G}}_{ij}=1 \\ \mathbf{Z}_{ij} \times \max(\mathbf{S}_{ij}-\beta, 0), & \mathbf{S}^{\text{G}}_{ij}=0 \end{cases} \tag{5.17}$$

where we set the margin value $\beta = 0.2$. Loss of Eq. (5.17) is valid because after Softmax and Sinkhorn operations, $\mathbf{S}_{ij} \in [0, 1]$. We also show permutation loss (Perm) (Wang *et al.*, 2019a). Result can be found in Fig. 5.5 (b) whereby the average accuracy on Pascal VOC is reported. All the settings are under CIE$_1$. For either loss, the proposed Hungarian attention can further enhance the accuracy, which is further visualized by a pair of matching results under P-loss and H-loss in Fig. 5.6.



(a) Reference Image (b) P-loss: **7/10** (c) H-loss: **8/10**

Figure 5.6: Visualization of a Matching Result: 10 Key Points in Each Image with 7 and 8 Correct Matchings Dispalyed, Respectively. Different Colors Across Images Indicate Node Correspondence. The Larger Size of Dot, the Larger Is the Predicted Value $\mathbf{S}_{ij}$. (A) the Reference Image. (B) Result on the Target Image from CIE$_1$-p. (C) Result on the Target Image from CIE$_1$-h. We See Though H-loss i.e. Hungarian Attention Loss Outputs Smaller Predicted Values, It Delivers a More Accurate Matching.

5.3.4 Results on Willow Object Class

We test the transfer ability on Willow Object Class (Cho *et al.*, 2013). It contains 256 images [1] of 5 categories in total, with three categories (face, duck and winebottle) collected from Caltech-256 and resting two (car and motorbike) from Pascal VOC 2007. This dataset is considered to have bias compared with Pascal VOC since images in the same category are with relatively fixed pose and background is much cleaner. We crop the object inside its bounding box and resize it to 256×256 as CNN input. While HARG is trained from scratch following the protocol in (Cho *et al.*, 2013), all the resting counterparts are either directly pre-trained from the previous section or fine-tuned upon the pre-trained models. We term the method "X-**V**" or "X-**W**"

[1]The data size is too small to train a deep model. Hence we only evaluate the transfer ability on this dataset.

Table 5.2: Accuracy (%) on Willow Object.

method	face	mbike	car	duck	wbottle
HARG	91.2	44.4	58.4	55.2	66.6
GMN-V	98.1	65.0	72.9	74.3	70.5
GMN-W	99.3	71.4	74.3	82.8	76.7
PCA-V	**100.0**	69.8	78.6	82.4	95.1
PCA-W	**100.0**	76.7	**84.0**	**93.5**	96.9
CIE-V	99.9	71.5	75.4	73.2	**97.6**
CIE-W	**100.0**	**90.0**	82.2	81.2	**97.6**

Table 5.3: Accuracy (%) on CUB.

method	intra-class	cross-class
GMN-D	89.6	89.9
GMN-P	90.4	90.8
GAT-P	93.2	93.4
PCA-P	92.9	93.5
PCA-H	93.7	93.5
CIE-P	94.1	93.8
CIE-H	**94.4**	**94.2**

to indicate pre-trained model on Pascal **V**OC or fine-tuned on **W**illow, respectively. CIE refers to CIE_1-H for short. Results in Table 5.2 suggest that our method is competitive to state-of-the-art.

5.4 Conclusion

We have presented a novel and effective approach for learning based graph matching. On one hand, the novelty of our method partially lies in the development of the Hungarian attention, which intrinsically adapts the matching problem. It is further observed from the experiments that Hungarian attention can improve several matching-oriented loss functions, which might bring about potential for a series of combinatorial problems. On the other hand, we also devise the channel independent embedding (CIE) technique for deep graph matching, which decouples the basic merging operations and is shown robust in learning effective graph representation. Extensive experimental results on multiple matching benchmarks show the leading performance of our solver, and highlight the orthogonal contribution of the two proposed components on top of existing techniques.

Chapter 6

LEARNING LATENT TOPOLOGY FOR GRAPH MATCHING

6.1 Motivation

With the strong learning ability of deep networks, recent research on graph matching (GM) has migrated from traditional deterministic optimization (Schellewald and Schnörr, 2005; Cho *et al.*, 2010; Zhou *et al.*, 2015a) towards learning-based methods (Zanfir and Sminchisescu, 2018; Wang *et al.*, 2019a; Yu *et al.*, 2020a). GM is a classic combinatorial and NP-hard problem (Loiola *et al.*, 2007). As the mathematical cornerstone for a series of real-world applications (e.g., image matching (Wang *et al.*, 2018b), social mining (Chiasserini *et al.*, 2018) and protein matching (Krissinel and Henrick, 2004)), GM has received persistent attention from the machine learning and optimization communities for many years.

Recently, deep learning based GM solvers (Zanfir and Sminchisescu, 2018; Wang *et al.*, 2019a; Yu *et al.*, 2020a; Fey *et al.*, 2020; Rolínek *et al.*, 2020) have enabled end-to-end training of GM on high-quality human labelled datasets (e.g., Pascal VOC (Everingham *et al.*, 2010a; Bourdev and Malik, 2009) and SPair-71k (Min *et al.*, 2019)), which greatly improved the model capacity. Any of the aforementioned deep GM algorithms behaves as an integral framework, of which the main parts cover topology construction [1] , feature extraction and differentiable GM solver. In this line of works, affinity $\mathbf{M}$ (see Eq. (1.1)) is not obtained beforehand, but calculated using node/edge features from some feature backbones given heuristically constructed connectivity, then fed to subsequent GM solvers. Therefore, recent investigation on deep

[1]Topology in some GM problems is pre-defined and needs to be fixed, such as graph isomorphism. In this proposal, we consider a more generic case where topology construction is necessary.

GM frameworks typically focuses on two essential parts: 1) node/edge feature backbone (e.g., graph convolutional networks Wang *et al.* (2019a), channel-independent embedding (Yu *et al.*, 2020a) and SplineCNN (Fey *et al.*, 2018)); 2) GM solvers (e.g., spectral (Zanfir and Sminchisescu, 2018), linear (Wang *et al.*, 2019a) and black-box (Pogancic *et al.*, 2020)). In particular, since the feature backbones are variants of Graph Neural Networks, they requires initial heuristically-constructed connectivity/topology (e.g., Delaunay (Wang *et al.*, 2019a) or k-nearest (Zhang and Lee, 2019)), and the topology remains **fixed** throughout the training procedure in almost all the existing deep GM methods. In this sense, *the construction of graph topology is only a pre-processing step, independent of the GM task.* This fixed mechanism was adopted in many GM applications ranging from computer vision (Wang *et al.*, 2019a; Yu *et al.*, 2020a; Fey *et al.*, 2020; Rolínek *et al.*, 2020) to social networks (Zhang and Tong, 2016; Heimann *et al.*, 2018; Xiong and Yan, 2020), and potentially limits the reliability under ambiguity and misleading.

From a learning perspective, we argue that freezing the graph topology for matching can hinder the capacity of deep GM frameworks. For a pre-defined graph topology, the linked nodes sometimes result in less meaningful or even misleading interaction. Though some earlier attempts (Cho and Lee, 2012; Cho *et al.*, 2013) sought to adjust the graph topology under traditional learning settings, such procedures cannot be readily integrated into end-to-end deep frameworks due to the undifferentiable nature. Our method is built upon the following hypothesis:

- *There exists some latent (distribution of) discrete topology better than what is heuristically created for GM.*

Based on this, we set out to learn the topology (or its distribution) that is more suitable for GM. We will investigate an end-to-end framework to jointly learn the latent graph topology and perform GM. Taking into account the distribution of the topology, we will try to leverage the power of graph generative model to automatically produce graph topology from given features and their geometric relations.

6.2 Learning Latent Topology for GM

In this section, we describe details of the proposed framework with two specific algorithms derived from *deterministic* and *generative* perspectives, respectively. Both algorithms are motivated by the *hypothesis* that there exists some latent topology more suitable for matching rather than a fixed one. Note that the proposed deterministic algorithm performs a standard forward-backward pass to jointly learn the topology and matching, while our generative algorithm consists of an alternative optimization procedure between estimating latent topology and learning matching under an Expectation-Maximization (EM) interpretation. In general, the generative algorithm assumes that a latent topology is sampled from a *latent distribution*, where the expected matching accuracy under this distribution is maximized. Therefore, we expect to learn a topology generator under such distribution. We reformulate GM in a Bayesian fashion for consistent discussion in Sec. 6.2.1, detail deterministic/generative latent module in Sec. 6.2.2 and discuss the loss functions from a probabilistic perspective in Sec. 6.2.3. We finally elaborate on the holistic framework and the optimization procedure for both algorithms (deterministic and generative) in Sec. 6.2.4.

6.2.1 Problem Definition and Background

Learning-based GM problem can be viewed as an extension to Eq. (1.1). Let $\mathcal{G}^{(s)}$ and $\mathcal{G}^{(t)}$ represent respectively the source and target graphs for matching. We represent a graph as $\mathcal{G} := \{\mathbf{X}, \mathbf{E}, \mathbf{A}\}$, where $\mathbf{X} \in \mathbb{R}^{n\times d_1}$ is the representation of n nodes with dimension d_1. $\mathbf{E} \in \mathbb{R}^{m\times d_2}$ are d_2-dimensional features of m edges and

$\mathbf{A} \in \{0,1\}^{n\times n}$ is initial connectivity (i.e., topology) matrix by heuristics (e.g., Delaunay triangulation). For notational brevity, we assume d_1 and d_2 remain intact after updating the features across each convolutional layers of GNN (i.e., feature dimensions of both nodes and edges will not change after each layer's update). Denote the matching $\mathbf{Z} \in \{0,1\}^{n\times n}$ between two graphs, where $\mathbf{Z}_{ij} = 1$ indicates a correspondence exists between node i in $\mathcal{G}^{(s)}$ and node j in $\mathcal{G}^{(t)}$, and $\mathbf{Z}_{ij} = 0$ otherwise. Given training samples $\{\mathbf{Z}_k, \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\}$ with $k = 1, 2, ..., N$, the objective of learning-based GM aims to maximize the likelihood:

$$\max_{\theta} \prod_{k} P_{\theta}\left(\mathbf{Z}_k | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) \tag{6.1}$$

where θ denotes model parameters. $P_{\theta}(\cdot)$ measures the probability of matching $\mathbf{Z}_k$ given the k-th pair, and is instantiated via a network parameterized by θ.

Being a generic module for producing latent topology, *our method can be flexibly and easily integrated into existing deep GM frameworks.* We build up our method based on state-of-the-art (Rolínek *et al.*, 2020), which utilizes SplineCNN (Fey *et al.*, 2018) for node/edge feature learning and black-box GM solver (Pogancic *et al.*, 2020).

SplineCNN is a method to perform graph-based representation learning via convolution operators defined based on B-splines (Fey *et al.*, 2018). The initial input to SplineCNN is $\mathcal{G} = \{\mathbf{X}, \mathbf{E}, \mathbf{A}\}$, where $\mathbf{X} \in \mathcal{G}^{n\times d_1}$ and $\mathbf{A} \in \{0,1\}^{n\times n}$ indicate node features and topology, respectively (same as in Sec. 6.2.1). $\mathbf{E} \in [0,1]^{n\times n\times d_2}$ is so-called pseudo-coordinates and can be viewed as $n^2 \times d_2$-dimensional edge features for a fully connected graph (in case $m = n^2$, see Sec. 6.2.1). Let normalized edge feature $\mathbf{e}(i,j) = \mathbf{E}_{i,j,:} \in [0,1]^{d_2}$ if a directed edge (i,j) exists ($\mathbf{A}_{i,j} = 1$), and $\mathbf{0}$ otherwise ($\mathbf{A}_{i,j} = 0$). Note topology $\mathbf{A}$ fully carries the information of $\mathcal{N}(i)$ which defines the neighborhood of node i. During the learning, $\mathbf{X}$ and $\mathbf{E}$ will be updated while topology $\mathbf{A}$ will not. Therefore SplineCNN is a geometric graph embedding method without adjusting the latent graph topology.

B-spline is employed as basic kernel in SplineCNN, where a basis function has only support on a specific real-valued interval (Piegl and Tiller, 2012).

Let $((N^q_{1,i})_{1\leq i\leq k_1}, ..., (N^q_{d,i})_{1\leq i\leq k_{d_2}})$ be d_2 B-spline bases with degree q. The kernel size is defined in $\mathbf{k} = (k_1, ..., k_{d_2})$. In SplineCNN, the continuous kernel function $g_l : [a_1, b_1] \times ... \times [a_{d_2}, b_{d_2}] \to \mathcal{G}$ is defined as:

$$g_l(\mathbf{e}) = \sum_{\mathbf{p}\in\mathcal{P}} w_{\mathbf{p},l} \cdot \mathbf{B}_{\mathbf{p}}(\mathbf{e}) \tag{6.2}$$

where $\mathcal{P} = (N^q_{1,i})_i \times ... \times (N^q_{d,i})_i$ is the B-spline bases (Piegl and Tiller, 2012) and $w_{\mathbf{p},l}$ is the trainable parameter corresponding to the lth node feature in $\mathbf{X}$, with $\mathbf{B}_{\mathbf{p}}$ being the product of the basis functions in $\mathbf{P}$:

$$\mathbf{B}_{\mathbf{p}} = \prod_{i=1}^{d} N^q_{i,p_i}(e_i) \tag{6.3}$$

where $\mathbf{e}$ is the pseudo-coordinate in $\mathbf{E}$. Then, given the kernel function $\mathbf{g} = (g_1, ..., g_{d_1})$ and the node feature $\mathbf{X} \in \mathcal{G}^{n\times d_1}$, one layer of the convolution at node i in SplineCNN reads:

$$(\mathbf{x} * \mathbf{g})(i) = \frac{1}{|\mathcal{N}(i)|} \sum_{l=1}^{d_1} \sum_{j\in\mathcal{N}(i)} x_l(j) \cdot g_l(\mathbf{e}(i,j)) \tag{6.4}$$

where $x_l(j)$ indicates the convolved node feature value of node j at lth dimension. This formulation can be tensorized into the following euqation with explicit topology matrix $\mathbf{A}$:

$$(\mathbf{x} * \mathbf{g}|\mathbf{A}) = (\hat{\mathbf{A}} \circ \mathbf{G})\hat{\mathbf{X}} \tag{6.5}$$

In this sense, we can back-propagate the gradient of $\mathbf{A}$. Reader are referred to Fey *et al.* (2018) for more comprehensive understanding of this method.

6.2.2 Modeling with Latent Topology

Existing learning-based graph matching algorithms consider $\mathbf{A}$ to be fixed throughout the computation without questioning if the input topology is optimal or not. This can be problematic since input graph construction is heuristic, and it never takes into account how suitable it is for the subsequent GM task. In our framework, instead of utilizing a fixed pre-defined topology, we consider to produce latent topology under two settings: 1) a deterministic and 2) a generative way. The former is often more efficient while the latter method can be more accurate at the cost of exploring more latent topology. Note that both methods produce *discrete topology* to verify our hypothesis about the existence of more suitable discrete latent topology for GM. The corresponding two deep structures are described below.

Deterministic learning: Given input features $\mathbf{X}$ and initial topology $\mathbf{A}$, the deterministic way of generating latent topology $\underline{\mathbf{A}} \in \{0,1\}^{n\times n}$ is [2] :

$$\begin{aligned} \underline{\mathbf{A}}_{ij} &= \text{Rounding}(\text{sigmoid}(\mathbf{y}_i^\top \mathbf{W} \mathbf{y}_j)) \\ \text{with} \quad \mathbf{Y} &= \text{GCN}(\mathbf{X}, \mathbf{A}) \end{aligned} \tag{6.6}$$

where GCN$(\cdot)$ is the graph convolutional networks (GCN) (Kipf and Welling, 2017) and $\mathbf{y}_i$ corresponds to the feature of node i in feature map $\mathbf{Y}$. $\mathbf{W}$ is the learnable parameter matrix. Note that function Rounding$(\cdot)$ is undifferentiable, and will be discussed in Sec. 6.2.4.

Generative learning: We reparameterize the representation:

$$P(\mathbf{y}_i|\mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{y}_i|\boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}^2)) \tag{6.7}$$

with $\boldsymbol{\mu} = \text{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$ and $\boldsymbol{\sigma} = \text{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$ are two GCNs producing mean and covariance. It is equivalent to sampling a random vector from i.i.d. uniform distribution

[2]We here consider the node feature $\mathbf{X}$ and topology $\mathbf{A}$. Edge feature $\mathbf{E}$ can be readily integrated as another input.

$\mathbf{s} \sim \mathcal{U}(\mathbf{0}, \mathbf{1})$, then applying $\mathbf{y} = \boldsymbol{\mu} + \mathbf{s} \cdot \boldsymbol{\sigma}$, where $(\cdot)$ is element-wise product.

Similar to Eq. (6.6), by introducing learnable parameter $\mathbf{W}$, the generative latent topology is sampled following i.i.d. distribution over each edge (i, j):

$$\begin{aligned} P(\underline{\mathbf{A}}|\mathbf{Y}) &= \prod_i \prod_j P(\underline{\mathbf{A}}_{ij}|\mathbf{y}_i, \mathbf{y}_j) \\ \text{with} \quad P(\underline{\mathbf{A}}_{ij} = 1|\mathbf{y}_i, \mathbf{y}_j) &= \text{sigmoid}(\mathbf{y}_i^\top \mathbf{W} \mathbf{y}_j) \end{aligned} \tag{6.8}$$

Since sigmoid$(\cdot)$ maps any input into $(0, 1)$, Eq. (6.8) can be interpreted as the probability of sampling edge (i, j). As the sampling procedure is undifferentiable, we apply Gumbel-softmax trick (Jang *et al.*, 2017) as another reparameterization procedure. As such, a latent graph topology $\underline{\mathbf{A}}$ can be sampled fully from distribution $P(\underline{\mathbf{A}})$ and the procedure becomes differentiable.

6.2.3 Loss Functions

In this section, we explain three loss functions and the underlying motivation: *matching loss*, *locality loss* and *consistency loss*. The corresponding probabilistic interpretation of each loss function can be found in Sec. 6.2.4. These functions are selectively activated in DLGM-D and DLGM-G (see Sec. 6.2.4). In DLGM-G, different loss functions are activated in inference and learning steps.

i) Matching loss. This common term measures how the predicted matching $\hat{\mathbf{Z}}$ diverges from ground-truth $\mathbf{Z}$. Following Rolínek *et al.* (2020), we adopt Hamming distance on node-wise matching:

$$\mathcal{L}_M = \text{Hamming}(\hat{\mathbf{Z}}, \mathbf{Z}) \tag{6.9}$$

ii) Locality loss. This loss is devised to account for the general prior that the produced/learnt graph topology should advocate local connections rather than distant ones, since two nodes may have less meaningful interaction once they are too distant from each other. In this sense, locality loss serves as a ***prior*** or regularizer in GM. As shown in multiple GM methods (Yu *et al.*, 2018b; Wang *et al.*, 2019a; Fey *et al.*, 2020), Delaunay triangulation is an effective way to deliver good locality. Therefore in our method, the locality loss is the Hamming distance between the initial topology $\mathbf{A}$ (obtained from Delaunay) and predicted topology $\underline{\mathbf{A}}$ for both the source graph and the target graph:

$$\mathcal{L}_L = \text{Hamming}(\mathbf{A}^{(s)}, \underline{\mathbf{A}}^{(s)}) + \text{Hamming}(\mathbf{A}^{(t)}, \underline{\mathbf{A}}^{(t)}) \tag{6.10}$$

We emphasize that the locality loss serves as a *prior* for latent graph. It focuses on advocating locality, but not reconstructing the initial Delaunay triangulation (as in Graph VAE (Kipf and Welling, 2016)).

iii) Consistency loss. One can imagine that a GM solver is likely to deliver better performance if two graphs in a training pair are similar. In particular, we anticipate the latent topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$ to be isomorphic under a specific matching, since isomorphic topological structures tend to be easier to match. Driven by this consideration, we devise the consistency loss which measures the level of isomorphism between latent topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$:

$$\mathcal{L}_C(\cdot|\mathbf{Z}) = |\mathbf{Z}^\top \underline{\mathbf{A}}^{(s)} \mathbf{Z} - \underline{\mathbf{A}}^{(t)}| + |\mathbf{Z} \underline{\mathbf{A}}^{(t)} \mathbf{Z}^\top - \underline{\mathbf{A}}^{(s)}| \tag{6.11}$$

Note that $\mathbf{Z}$ does not necessarily refer to the ground-truth, but can be any predicted matching. In this sense, latent topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$ can be generated jointly given the matching $\mathbf{Z}$ as guidance. This term can also serve as a consistency prior or regularization. We give a schematic example showing the merit of introducing the consistency loss in Fig. 5.1(b).

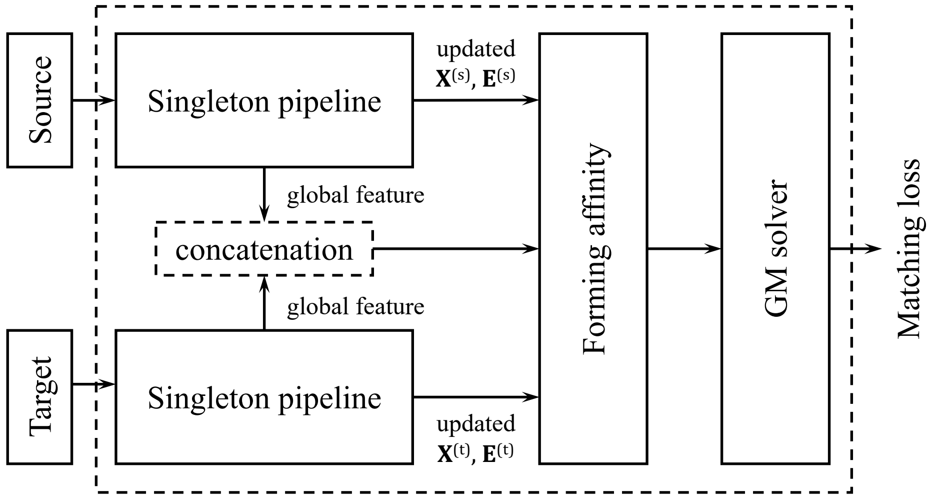


Figure 6.1: Holistic Pipeline of Dlgm Consisting of Two Singleton Pipelines. Details about a Singleton Pipeline Can Be Found In Fig. 6.2

6.2.4 Framework

A schematic diagram of our framework is given in Fig. 6.1 consisting of two singleton pipelines, with each handling a single input. Detailed diagram can be found in Fig. 5.1(a) which consists of a singleton pipeline for processing a single image. It consists of three essential modules: a feature backbone (N_B), a latent topology module (N_G) and a feature refinement module (N_R). Specifically, module N_G corresponds to Sec. 6.2.2 with deterministic or generative implementations. Note that the geometric relations of keypoints provide some prior for generating topology $\underline{\mathbf{A}}$. We employ VGG16 (Simonyan and Zisserman, 2014) as N_B and feed the produced node feature $\mathbf{X}$ and edge feature $\mathbf{E}$ to N_G. N_B also produces a global feature for each image. After generating the latent topology $\underline{\mathbf{A}}$, we pass over $\mathbf{X}$ and $\mathbf{E}$ together with $\underline{\mathbf{A}}$ to N_R (SplineCNN (Fey *et al.*, 2018)). The holistic pipeline handling pairwise graph inputs can be found in Fig. 6.1, which consists of two copies of singleton pipeline processing source and target data (in a Siamese fashion), respectively. Then

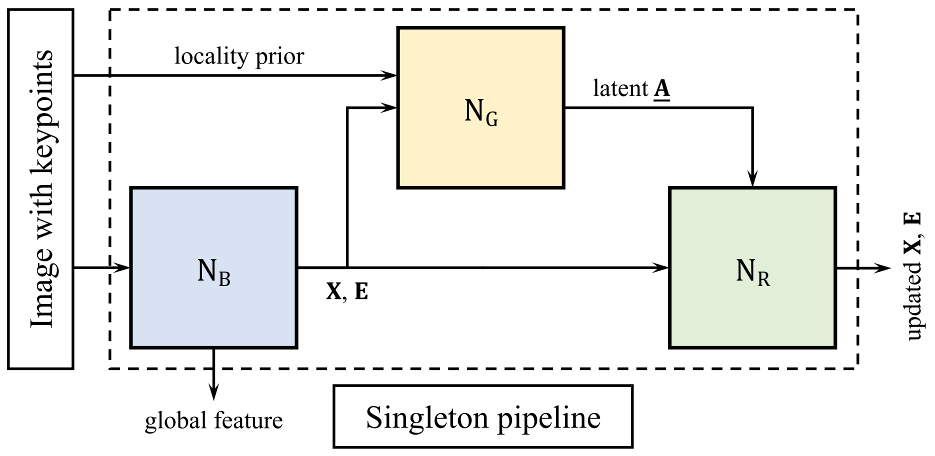


Figure 6.2: One of the Two Branches of Our DLGM Framework (See the Complete Version in Fig. 6.3). N_B: VGG16 as Backbone Producing a Global Feature of Input Image, Node and Edge Features; N_G: Deterministic or Generative Module Producing Latent Topology; N_R: SplineCNN for Feature Refinement Producing Updated Node and Edge Features.

the outputs of two singleton pipelines are formulated into affinity matrix, followed by a differentiable Blackbox GM solver (Pogancic *et al.*, 2020) with message-passing mechanism (Swoboda *et al.*, 2017). Note that, if N_G is removed, the holistic pipeline with only $\mathrm{N}_B + \mathrm{N}_R$ is identical to the method in (Rolínek *et al.*, 2020). Readers are referred to this strong baseline (Rolínek *et al.*, 2020) for more mutual algorithmic details.

Optimization with Deterministic Latent Graph

We now show how to optimize with the deterministic latent graph module, where the topology $\underline{\mathbf{A}}$ is produced by Eq. (6.6). The objective of matching conditioned on the produced latent topology $\underline{\mathbf{A}}$ becomes:

$$\max \prod_k P\left(\mathbf{Z}_k | \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}, \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) \tag{6.12}$$

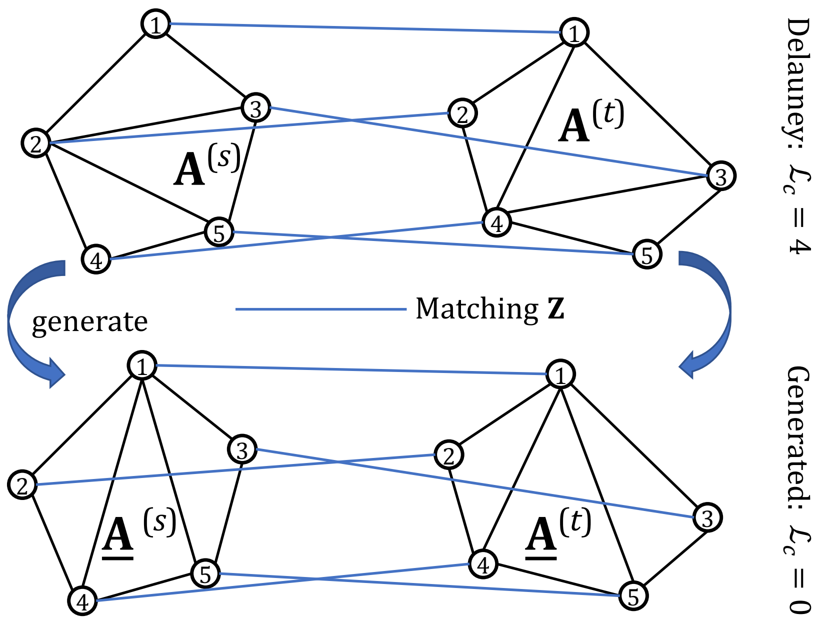

$\mathbf{A}^{(s)}$
$\mathbf{A}^{(t)}$
Delauney: $\mathcal{L}_c = 4$
generate
Matching $\mathbf{Z}$
$\underline{\mathbf{A}}^{(s)}$
$\underline{\mathbf{A}}^{(t)}$
Generated: $\mathcal{L}_c = 0$

Figure 6.3: A Schematic Figure Showing the Merit of Introducing Consistency Loss for Training. Initial Topology Is Constructed Using Delaunay Triangulation. Given Matching as Guidance, Latent Topology Is Generated from Inputs. Note That the Learned Two Structures of Topology Are Isomorphic ($\mathcal{L}_c = 0$), Which Is Easier to Match in Test, Compared to Non-isomorphic Input Structures ($\mathcal{L}_c = 4$).

Eq. (6.12) can be optimized with standard back-propagation with three loss terms activated, except for the Rounding function (see Eq. (6.6)), which makes the procedure undifferentiable. To address this, we use straight-through operator (Bengio *et al.*, 2013) which performs a standard rounding during the forward pass but approximates it with the gradient of identity during the backward pass on $[0, 1]$:

$$\partial \text{Rounding}(x)/\partial x = 1 \tag{6.13}$$

Though there exist some unbiased gradient estimators (e.g., REINFORCE (Williams, 1992)), the biased straight-through estimator proved to be more efficient and has been successfully applied in several applications (Chung *et al.*, 2017; Campos *et al.*, 2018). All the network modules ($\text{N}_G + \text{N}_B + \text{N}_R$) are simultaneously learned during the training. All three losses are activated in the learning procedure (see Sec. 6.2.3), which are applied on the predicted matching $\hat{\mathbf{Z}}$, the latent topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$. We term the algorithm under this setting **DLGM-D**.

Optimization with Generative Latent Graph

In this setting, the source and target latent topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$ are *sampled* according to Eq. (6.7) and (6.8). The objective becomes:

$$\max \prod_k \int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta \left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)} \right) \tag{6.14}$$

Unfortunately, directly optimizing Eq. (6.14) is difficult due to the integration over $\underline{\mathbf{A}}$, which is intractable. Instead, we maximize the evidence lower bound (ELBO) (Bishop, 2006) as follows:

$$\begin{aligned} &\log P_\theta(\mathbf{Z}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)}) \geq \\ &\mathbb{E}_{Q_\phi(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)})} \Big[\log P_\theta(\mathbf{Z}, \underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)}) \\ &\qquad\qquad - \log Q_\phi(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)}) \Big] \end{aligned} \tag{6.15}$$

where $Q_\phi(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)})$ can be any joint distribution of $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$ given the input graphs $\mathcal{G}^{(s)}$ and $\mathcal{G}^{(t)}$. Equality of Eq. (6.15) holds when $Q_\phi(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)}) = P_\theta(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathbf{Z}, \mathcal{G}^{(s)}, \mathcal{G}^{(t)})$. For tractability, we introduce the independence by assuming that we can use an identical latent topology module Q_ϕ (corresponding to N_G in Fig. 5.1(a)) to separately handle each input graph:

$$Q_\phi(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)}) = Q_\phi(\underline{\mathbf{A}}^{(s)}|\mathcal{G}^{(s)}) Q_\phi(\underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(t)}) \tag{6.16}$$

which can greatly simplify the model complexity. Then we can utilize a neural network to model Q_ϕ (similar to modeling P_θ). The optimization of Eq. (6.15) is studied in (Neal and Hinton, 1998), known as the Expectation-Maximization (EM) algorithm.

Optimization of Eq. (6.15) alternates between E-step and M-step. During E-step (inference), P_θ is fixed and the algorithm seeks to find an optimal Q_ϕ to approximate the true posterior distribution (see Sec. 6.2.5 for explanation):

$$P_\theta(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathbf{Z}, \mathcal{G}^{(s)}, \mathcal{G}^{(t)}) \tag{6.17}$$

During M-step (learning), Q_ϕ is instead fixed and algorithm alters to maximize the likelihood:

$$\mathbb{E}_{Q_\phi(\underline{\mathbf{A}}^{(s)}|\mathcal{G}^{(s)}),Q_\phi(\underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(t)})}\left[\log P_\theta(\mathbf{Z}, \underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)})\right] \propto -\mathcal{L}_M \tag{6.18}$$

We detail on the inference and learning steps as follows.

Inference. This step focuses on deriving posterior distribution $P_\theta(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathbf{Z}, \mathcal{G}^{(s)}, \mathcal{G}^{(t)})$ using its approximation Q_ϕ. To this end, we fix the parameters θ in modules N_B and N_R, and only update the parameters ϕ in module N_G corresponding to Q_ϕ. As stated in Sec. 6.2.2, we employ the Gumbel-softmax trick for sampling discrete $\underline{\mathbf{A}}$ (Jang *et al.*, 2017). To this end, we can formulate a 2D vector $\mathbf{a}_{ij} = [P(\underline{\mathbf{A}}_{ij} = 1), 1-P(\underline{\mathbf{A}}_{ij} = 1)]^\top$. Then the sampling becomes:

$$\mathrm{softmax}\left(\log(\mathbf{a}_{ij}) + \mathbf{h}_{ij}; \tau\right) \tag{6.19}$$

where $\mathbf{h}_{ij}$ is a random 2D vector from Gumbel distribution, and τ is a small temperature parameter. We further impose prior on latent topology $\underline{\mathbf{A}}$ given $\mathbf{A}$ through *locality loss*:

$$\log \prod_{i,j} P(\underline{\mathbf{A}}_{ij}|\mathbf{A}_{ij}) \propto -\mathcal{L}_L(\underline{\mathbf{A}}, \mathbf{A}) \tag{6.20}$$

which is to preserve the locality in initial topology $\mathbf{A}$. It should also be noted that $\mathbf{Z}$ is the *predicted* matching from current P_θ, as Q_ϕ is an approximation. Besides, we also anticipate two generated topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$ from a graph pair should be similar (isomorphic) given $\mathbf{Z}$:

$$\log P\left(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathbf{Z}\right) \propto -\mathcal{L}_C\left(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathbf{Z}\right) \tag{6.21}$$

Algorithm 5 DLGM-D

```
1:  Input: $\mathcal{G}^s$, $\mathcal{G}^t$ and ground-truth $\mathbf{Z}$;
2:  Output: matching $\hat{\mathbf{Z}}$;
3:  Pretrain $P_\theta$ using Eq. (6.12), given Delaunay as input topology;
4:  repeat
5:      # Inference (E-step):
6:      Obtain predicted matching $\hat{\mathbf{Z}}$ using fixed $P_\theta$;
7:      Update $Q_\phi$ (i.e. $\mathrm{N}_G$) with loss $\mathcal{L}_L + \mathcal{L}_C(\cdot|\hat{Z})$ according to Eq. (6.17);
8:      # Learning (M-step):
9:      Obtain predicted graph topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$ using $Q_\phi$;
10:     Update $P_\theta$ (i.e. $\mathrm{N}_B$ and $\mathrm{N}_R$) with loss $\mathcal{L}_M$ given $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$ according to
    Eq. (6.18);
11: until converge
12: Predict topology and the matching $\hat{\mathbf{Z}}$ with whole network activated (i.e. $\mathrm{N}_G +$
    $\mathrm{N}_B + \mathrm{N}_R$);
```

In summary, we activate *locality loss* and *consistency loss* as $\alpha\mathcal{L}_L + \beta\mathcal{L}_C$ during the inference step, where the latter loss is conditioned with the predicted matching rather than the ground-truth. Note that the inference step involves twice reparameterization tricks corresponding to Eq. (6.7) and (6.19), respectively. While the first generates the continuous topology distribution under edge independence assumption, the second performs discrete sampling according to the generated topology distribution.

Learning. This step optimizes P_θ by fixing Q_ϕ. We sample discrete graph topologies $\underline{\mathbf{A}}$s completely from the probability of edge $P(\underline{\mathbf{A}}_{ij} = 1)$. Once latent topology $\underline{\mathbf{A}}$s are sampled, we feed them to module N_R together with the node-level features from N_B. Only N_B and N_R are updated in this step, and only *matching loss* $\mathcal{L}_M$ is activated.

Remark. Note for each pair of graphs in training, we use an identical random vector $\mathbf{s}$ for generating both graphs' topology (see Eq. (6.7)). We pretrain the network P_θ before alternativly training P_θ and Q_ϕ. During pretraining, we activate $\mathrm{N}_B + \mathrm{N}_R$ modules and $\mathcal{L}_M$ loss during pretraining, and feed the network the topology obtained from Delaunay as the latent topology. After pretraining, the optimization will switch between inference and learning steps until convergence. We term the setting of generative latent graph matching as **DLGM-G** and summarize it in Alg. 5.

6.2.5 Mathematical Derivation of DLGM-D

We give more details of the optimization on DLGM-D in this section. This part also interprets some basic formulation conversion (e.g. from Eq. (6.1) to its Bayesian form). First, we assume there is no latent topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(s)}$ at the current stage. In this case, the objective of GM is simply:

$$\max \prod_k P_\theta \left(\mathbf{Z}_k | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)} \right) \tag{6.22}$$

where P_θ measures the probability of a matching $\mathbf{Z}_k$ given graph pair $\mathcal{G}_k^{(s)}$ and $\mathcal{G}_k^{(t)}$. If we impose the latent topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$, as well as some *distribution* over them, then Eq. (6.22) can be equivalently expressed as:

$$\begin{aligned} &\max \prod_k P_\theta \left(\mathbf{Z}_k | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)} \right) \\ =& \max \prod_k \int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta \left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)} \right) \end{aligned} \tag{6.23}$$

where $P_\theta \left(\mathbf{Z}_k | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)} \right)$ is the marginal distribution of $P_\theta \left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)} \right)$ with respect to $\mathbf{Z}_k$, since $\underline{\mathbf{A}}_k^{(s)}$ and $\underline{\mathbf{A}}_k^{(t)}$ are integrated over some distribution. Herein we can impose another distribution of the topology $Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})$ characterized by parameter ϕ, then we have:

$$\begin{aligned}
&\log \int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) \\
=&\log \int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) \frac{Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})}{Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})} \\
=&\log \left(\mathbb{E}_{Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})} \left[\frac{P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}{Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})}\right]\right) \\
\geq&\mathbb{E}_{Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})} \Big[\log P_\theta(\mathbf{Z}, \underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)}) - \\
&\quad \log Q_\phi(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)})\Big]
\end{aligned} \tag{6.24}$$

where the final step is derived from Jensen's inequality. Since optimizating Eq. 6.23 is difficult, we can alter to maximize the right hand side of inequality of Eq. (6.24) instead, which is the Evidence Lower Bound (ELBO) (Bishop, 2006). Since two input graphs are handled separately by two identical subroutines (see Fig. 6.2), we can then impose the independence of topology $\underline{\mathbf{A}}_k^{(s)}$ and $\underline{\mathbf{A}}_k^{(t)}$: $Q_\phi(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(s)}, \mathcal{G}^{(t)}) = Q_\phi(\underline{\mathbf{A}}^{(s)}|\mathcal{G}^{(s)})Q_\phi(\underline{\mathbf{A}}^{(t)}|\mathcal{G}^{(t)})$. In this sense, we can utilize the same parameter ϕ to characterize two identical neural networks (generators) for modeling Q_ϕ.

Assuming θ is fixed, ELBO is determined by Q_ϕ. According to Jensen's inequality, equality of Eq. (6.24) holds when:

$$\frac{P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}{Q_\phi\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)} = c \tag{6.25}$$

where $c \neq 0$ is a constant. We then have:

$$\begin{aligned}
&\int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) \\
=&c \int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} Q_\phi\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}|\mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)
\end{aligned} \tag{6.26}$$

As Q_ϕ is a distribution, we have:

$$\int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} Q_\phi\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) = 1 \tag{6.27}$$

Therefore, we have:

$$\int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) = c \tag{6.28}$$

We now have:

$$\begin{aligned}
&Q_\phi\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) \\
=&\frac{P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}{c} \\
=&\frac{P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}{\int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)} \\
=&\frac{P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}{P_\theta\left(\mathbf{Z}_k | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)} \\
=&P_\theta\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathbf{Z}_k, \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)
\end{aligned} \tag{6.29}$$

Eq. (6.29) shows that, once θ is fixed, maximizing ELBO amounts to finding a distribution Q_ϕ approximating the posterior probability $P_\theta\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathbf{Z}_k, \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)$. This can be done by training the generator Q_ϕ to produce latent topology $\underline{\mathbf{A}}$ given graph pair and the matching $\mathbf{Z}$. This corresponds to the **Inference** part in Sec. 6.2.4.

6.3 Experiment

We conduct experiments on datasets including Pascal VOC with Berkeley annotation (Everingham *et al.*, 2010a; Bourdev and Malik, 2009), Willow ObjectClass (Cho *et al.*, 2013) and SPair-71K (Min *et al.*, 2019). We report the per-category and average performance. The objective of all experiments is to maximize the average matching accuracy. Both our **DLGM-D** and **DLGM-G** are tested. Except for the

Table 6.1: Accuracy (%) on Pascal VOC (Best in Bold). Only Inlier Keypoints Are Considered.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	Ave
GMN	31.1	46.2	58.2	45.9	70.6	76.4	61.2	61.7	35.5	53.7	58.9	57.5	56.9	49.3	34.1	77.5	57.1	53.6	83.2	88.6	57.9
GAT-H	47.2	61.6	63.2	53.3	79.7	70.1	65.3	70.5	38.4	64.7	62.9	65.1	66.2	62.5	41.1	78.8	67.1	61.6	81.4	91.0	64.6
PCA	40.9	55.0	65.8	47.9	76.9	77.9	63.5	67.4	33.7	65.5	63.6	61.3	68.9	62.8	44.9	77.5	67.4	57.5	86.7	90.9	63.8
CIE_1-H	51.2	69.2	70.1	55.0	82.8	72.8	69.0	74.2	39.6	68.8	71.8	70.0	71.8	66.8	44.8	85.2	69.9	65.4	85.2	92.4	68.9
DGMC	50.4	67.6	70.7	70.5	87.2	85.2	82.5	74.3	46.2	69.4	69.9	73.9	73.8	65.4	51.6	98.0	73.2	69.6	94.3	89.6	73.2
BBGM	61.5	75.0	78.1	80.0	87.4	93.0	89.1	80.2	58.1	77.6	76.5	79.3	78.6	78.8	66.7	97.4	76.4	77.5	**97.7**	94.4	80.1
DLGM-D (ours)	60.8	76.0	77.5	79.6	**88.0**	**95.0**	**90.4**	81.6	67.3	82.4	**94.1**	79.6	81.2	80.5	68.9	**98.6**	77.1	87.5	97.0	95.3	82.9
DLGM-G (ours)	**64.7**	**78.1**	**78.4**	**81.0**	87.2	94.6	89.7	**82.5**	**68.5**	**83.0**	93.9	**82.3**	**82.8**	**82.7**	**69.6**	**98.6**	**78.9**	**88.9**	97.4	**96.7**	**83.8**

ablation study, we consistently conduct experiments under $\alpha = 5.0$ and $\beta = 0.3$. We will test different combinations of αs and βs in the ablation study (Sec. 6.3.4).

Peer methods. We conduct comparison experiments against the following algorithms: 1) **GMN** (Zanfir and Sminchisescu, 2018), which is a seminal work incorporating graph matching into deep learning framework equipped with a spectral solver (Egozi *et al.*, 2013); 2) **PCA** (Wang *et al.*, 2019a). This method treats graph matching as feature matching problem and employs GCN (Kipf and Welling, 2017) to learn better features; 3) **$\mathbf{CIE_1}$/GAT-H** (Yu *et al.*, 2020a). This chapter develops an embedding and attention mechanism, where GAT-H is the version by replacing the basic embedding block with Graph Attention Networks (Veličković *et al.*, 2018); 4) **DGMC** (Fey *et al.*, 2020). This method devises a post-processing step by emphasizing the neighborhood similarity; 5) **BBGM** (Rolínek *et al.*, 2020). It integrates a differentiable linear combinatorial solver (Pogancic *et al.*, 2020) into a deep learning framework and achieves state-of-the-art performance.

Table 6.2: F1-score (%) on Pascal VOC. Experiment Are Performed on a Pair of Images Where Both Inlier and Outlier Keypoints Are Considered. BBGM-max Is a Setting in Rolínek *et al.* (2020).

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	Ave
BBGM-max	35.5	68.6	46.7	36.1	85.4	58.1	25.6	51.7	27.3	51.0	46.0	46.7	48.9	58.9	29.6	93.6	42.6	35.3	70.7	79.5	51.9
BBGM	42.7	70.9	57.5	46.6	85.8	64.1	51.0	63.8	42.4	63.7	47.9	61.5	63.4	69.0	46.1	94.2	57.4	39.0	**78.0**	82.7	61.4
DLGM-D (ours)	42.5	71.8	57.8	46.8	**86.9**	70.3	**53.4**	66.7	53.8	67.6	64.7	64.6	65.2	70.1	**47.9**	95.5	59.6	47.7	77.7	82.6	63.9
DLGM-G (ours)	**43.8**	**72.9**	**58.5**	**47.4**	86.4	**71.2**	53.1	**66.9**	**54.6**	**67.8**	**64.9**	**65.7**	**66.9**	**70.8**	47.4	**96.5**	**61.4**	**48.4**	77.5	**83.9**	**64.8**

Table 6.3: Accuracy (%) on SPair-71K Compared with State-of-the-art Methods (Best in Bold).

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	dog	horse	mbike	person	plant	sheep	train	tv	Ave
DGMC	54.8	44.8	80.3	70.9	65.5	90.1	78.5	66.7	66.4	73.2	66.2	66.5	65.7	59.1	98.7	68.5	84.9	98.0	72.2
BBGM	66.9	57.7	85.8	78.5	66.9	95.4	86.1	74.6	68.3	78.9	73.0	67.5	79.3	73.0	**99.1**	74.8	95.0	**98.6**	78.9
DLGM-D (ours)	69.8	64.4	**86.8**	79.9	**69.8**	**96.8**	**87.3**	77.7	77.5	83.1	76.7	**69.6**	**85.1**	75.1	98.7	76.4	95.8	97.9	81.3
DLGM-G (ours)	**70.4**	**66.8**	86.7	**81.7**	69.2	96.4	85.8	**79.5**	**78.4**	**84.0**	**79.4**	69.4	84.5	**76.6**	**99.1**	**75.9**	**96.4**	98.5	**82.0**

6.3.1 Results on Pascal VOC.

This dataset (Everingham *et al.*, 2010a; Bourdev and Malik, 2009) consists of 7,020 training images and 1,682 testing images with 20 classes in total, together with the object bounding boxing for each. Following the data preparation in (Wang *et al.*, 2019a), each object within the bounding box is cropped and resized to 256 × 256. The number of nodes per graph ranges from 6 to 23. We further follow (Rolínek *et al.*, 2020) under two evaluating metrics: 1) Accuracy: this is the standard metric evaluated on the keypoints by filtering out the outliers; 2) F1-score: this metric is evaluated without keypoint filtering, being the harmonic mean of precision and recall. Therefore, task 2) can be viewed as *common sub-graph matching with outliers.* Exper-

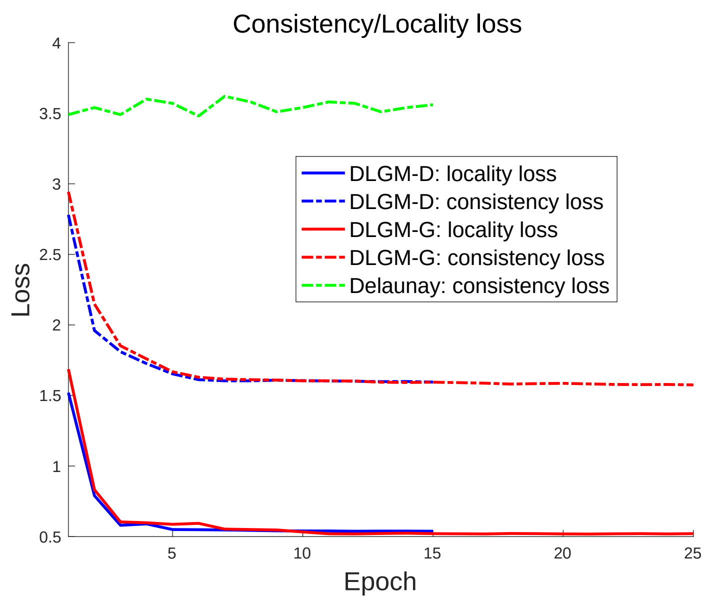


Figure 6.4: Consistency Loss (Eq. (6.10) and Locality Loss (Eq. 6.11)) Keep Decrease over Training Which Suggests the Effectiveness for Adaptive Topology Learning for Matching.

imental results on the two setting are shown in Tab. 6.1 and Tab. 6.2. The proposed method under either settings of DLGM-D and DLGM-G outperforms counterparts by accuracy and f1-score. DLGM-G generally outperforms DLGM-D.

Quality of generated topology. We further show the consistency/locality curve vs epoch in Fig. 6.4, since both consistency and locality losses can somewhat reflect the quality of topology generation. It shows that both locality and consistency losses descend during the training. Note that the consistency loss with Delaunay triangulation (green dashed line) is far more larger than our generated ones (blue/red dashed line). This clearly supports the claim that our method generates similar (more isomorphic) typologies, as well as preserving locality.

Table 6.4: Accuracy (%) on Willow Object.

Method	setting	face	mbike	car	duck	wbottle
GMN	Pt	98.1	65.0	72.9	74.3	70.5
	Wt	99.3	71.4	74.3	82.8	76.7
PCA	Pt	100.0	69.8	78.6	82.4	95.1
	Wt	100.0	76.7	84.0	93.5	96.9
CIE	Pt	99.9	71.5	75.4	73.2	97.6
	Wt	100.0	90.0	82.2	81.2	97.6
DGMC	Pt	98.6	69.8	84.6	76.8	90.7
	Wt	100.0	98.8	96.5	93.2	99.9
BBGM	Pt	100.0	95.8	89.1	89.8	97.9
	Wt	100.0	98.9	95.7	93.1	99.1
DLGM-D (ours)	Pt	100.0	95.5	91.3	91.4	97.9
	Wt	100.0	99.4	95.9	92.8	99.3
DLGM-G (ours)	Pt	99.9	96.4	92.0	91.8	98.0
	Wt	100.0	99.3	96.5	93.7	99.3

6.3.2 Results on Willow Object.

The benchmark (Cho *et al.*, 2013) consists of 256 images in 5 categories, where two categories (car and motorbike) are subsets from Pascal VOC. Following the protocol in Wang *et al.* (2019a), we crop the image within the object bounding box and resize it to 256×256. Since the dataset is relatively small, we conduct the experiment to verify the transfer ability of different methods under two settings: 1) trained on Pascal VOC and directly applied to Willow (Pt); 2) trained on Pascal VOC then finetuned on Willow (Wt). Results under the two settings are shown in Tab. 6.4. Since this dataset is relatively small, further improvement is difficult. It is shown both DLGM-D and DLGM-G have good transfer ability.

6.3.3 Results on SPair-71K.

This dataset (Min *et al.*, 2019) is much larger than Pascal VOC and WillowObject. It consists of 70,958 image pairs collected from Pascal VOC 2012 and Pascal 3D+ (53,340 for training, 5,384 for validation and 12,234 for testing). It improves Pascal VOC by removing ambiguous categories *sofa* and *dining table*. This dataset is considered to contain more difficult matching instances and higher annotation quality. Results are summarized in Tab. 6.3. Our method consistently improves the matching performance, agreeing with those in Pascal VOC and Willow.

6.3.4 Ablation Study

We conduct ablation to show the effectiveness of some factors involved in our framework (e.g., sampling size of the generator and varying loss strength α and β).

In the first part, we evaluate the performance of DLGM-D and DLGM-G by selectively deactivating different loss functions $\mathcal{L}_M$, $\mathcal{L}_C$ and $\mathcal{L}_L$. Since our method involves a sampling procedure, we also conduct the test on DLGM-G using different sample size of the generator. This ablation test is conducted on Pascal VOC dataset and average accuracy is reported in Tab. 6.5 and 6.6.

We first test the performance of both settings of DLGM by selectively activate the designated loss functions. Experimental results are summarized in Tab. 6.5. As matching loss $\mathcal{L}_M$ is essential for GM task, we constantly activate this loss for all settings. Note once $\mathcal{L}_C$ and $\mathcal{L}_L$ are both deactivated, our method will degenerate into BBGM (Rolínek *et al.*, 2020). In this case, there will be no need to train the generator

Table 6.5: Selectively Deactivating Loss Functions on Pascal VOC. $\mathcal{L}_M$, $\mathcal{L}_C$ and $\mathcal{L}_L$ Are Selectively Activated in DLGM-D and DLGM-G. "full" Indicates All Loss Functions Are Activated.

method	Average Accuracy (%)
DLGM-D ($\mathcal{L}_M + \mathcal{L}_C$)	79.8
DLGM-D ($\mathcal{L}_M + \mathcal{L}_L$)	79.5
DLGM-G ($\mathcal{L}_M + \mathcal{L}_C$)	80.9
DLGM-G ($\mathcal{L}_M + \mathcal{L}_L$)	80.4
DLGM-D (full)	82.9
DLGM-G (full)	83.8

Q_ϕ. We see that the proposed novel losses $\mathcal{L}_C$ and $\mathcal{L}_L$ can consistently enhance the matching performance. Besides, DLGM-G indeed delivers better performance than DLGM-D under fair comparison.

We then test the impact of sample size from the generator Q_ϕ under DLGM-G. Experimental results are summarized in Tab. 6.6. We see that along with the increasing sample size, the average accuracy ascends. The performance becomes stable when the sample size reaches over 16.

Remark. For time efficiency, if we consider the training time of the baseline Rolínek *et al.* (2020) to be 1x, the training time of our method under discriminative setting is around 1.2x-1.3x. The time cost of our method under generative setting is around 8x-9x with sample size 16. We didn't observe any obvious efficiency gap for testing.

In the second part, we present more detailed results by varying the loss strength αs and βs for DLGM-G. Letting the loss at inference step be $\alpha\mathcal{L}_L + \beta\mathcal{L}_C$, Tab. 6.7

Table 6.6: Average Matching Accuracy under Different Sampling Sizes from the Generator Q_ϕ with "full" DLGM-G Setting.

#Sample	Ave
1	82.5
2	83.2
4	83.2
8	83.5
16	83.8
32	83.7

shows the performance of DLGM-G with varying α and β on Pascal VOC with only inliers (Note we reported $\alpha = 5.0$ and $\beta = 0.3$ in all the previous experiments on each dataset):

6.4 Conclusion

Recent deep GM methods have delivered significant performance gain over traditional ones through learning node/edge features and GM solvers. However, beyond relying on heuristics, there is little work on learning more effective topology for improved matching. In this chapter, we hypothesize that learning a better (distribution of) discrete graph topology can significantly improve the matching, thus being essential. As such, we propose to incorporate a latent topology module under an end-to-end deep framework that learns to produce better graph topology. We present the interpretation and optimization of the topology learning module from deterministic and generative perspectives respectively. Experimental results show that, by learning the latent topology, the matching performance can be consistently and

Table 6.7: Ablation Study of DLGM-G on Pascal VOC Dataset. α and β Correspond to the Strength of Locality Loss $\mathcal{L}_L$ and Consistency Loss $\mathcal{L}_C$, Respectively. Average Accuracy (%) Is Reported.

α \ β	0.1	0.2	0.3	0.4	0.5
4.0	82.0	81.8	82.4	82.1	81.9
4.5	82.2	82.6	82.9	82.5	82.5
5.0	82.3	83.3	83.8	83.1	82.5
5.5	82.0	82.9	83.3	83.0	82.7

significantly enhanced on several public datasets, with only minimal modification to existing method.

Chapter 7

CONCLUSION

Being an instance of combinatorial problem, graph matching has attracted researchers for many years. This is due to its appealing challenge of NP-hard nature, as well its potential in a large range of real-world applications. Early study about graph matching is mainly focused on deterministic modeling and optimization, where matching is considered as an explicitly formulated optimization problem without any learnable parameters. Although several modeling methods for matching appeared (e.g., linearization, quadratic and semi-definite programming), such a deterministic fashion greatly hinders the model capacity and prohibits a solver to be applied adaptively on different tasks with various graph degradation. Therefore, with traditional deterministic modelings, one can hardly obtain a reliable graph matching solver under complex environment.

Although some early attempts resort to integrate higher reliability by modeling graph matching in a parameteric fashion, such efforts are too straightforward and shallow, resulting in limited improvement over traditional methods without any learning ability. Such a case didn't change much, until a seminal deep-learning-based framework was developed by Zanfir and Sminchisescu (2018). By exploiting the powerful learning ability of deep networks, this method outperformed the best known deterministic method by a significant margin. Since then, a series of notable works were proposed benefiting from deep learning. Nevertheless, we also note that most presented works are focusing on designing the network structure, but paying less attention to some more essential issues: how a deep learning method approximates discrete problem, and what role the topology of graphs plays in such a framework.

As such, this dissertation focuses on three identified vital aspects of graph matching: modeling, optimization and learning. By investigating each of the aspects, it is anticipated to grasp the core of reliable graph matching in terms of both theory and application under a unified perspective. Taking into account the series of issues, four specific tasks are intensively investigated corresponding to the aforementioned aspects. These series of works aim at tackling the essential issues separately, as well as providing a unified and feasible way of deriving more reliable graph matching solvers under uncertainty.

The investigation in Chapter 3 and Chapter 4 is generally for understanding and exploring the mechanism of reliable approximation and optimization for graph matching under the deterministic setting. In particular, the proposed modeling paradigm in Chapter 3 provides a generalized counterpart of QAP, which is capable of incorporating more complex graph deformation and offering alternative optimization trajectory. The proposed regularization technique in Chapter 4 is yet another mathematical continuation approach to derive a discrete solution in an asymptotic fashion, which can in turn be applied to the modeling in Chapter 3. Therefore, under a traditional deterministic perspective, Chapter 3 and 4 can serve as complements for each other. Together, these two chapters offer a feasible mathematical backbone to be readily incorporated with deep learning techniques. Chapter 5 and 6, on the other hand, extend deterministic optimization of graph matching into a learning-based framework, meanwhile seek to tackle several essential issues derived from the combinatorial nature of the graph matching problem. Chapter 5 studies the Hungrarian algorithm, and proposes a surrogate to selectively perform back-propagation, which can fill the gap between the discrete Hungarian sampling and its continuous approximation (i.e. Sinkhorn layer). Such a mechanism offers a feasible and efficient way to avoid overfitting, leading to higher reliability in several real-world benchmarks. Chapter 6 considers a more general and basic problem of how the topology of graph can impact the matching. We demonstrate in Chapter 6 that the necessity and the existence of latent topology for graph matching task, and provide an effective way of deriving such latent topology using a series of graph matching solvers. To the best of our knowledge, the performance of the proposed method in Chapter 6 outperformed the previous state-of-the-art method by a large margin on several very challenging real-world benchmarks (e.g., Pascal VOC and SPair-71K). Since such benchmarks generally consists of high noise and massive outliers, our work has made an effort towards more reliable graph matching. In summary, the research in Chapter 3, 4, 5 and 6 in this dissertation is coherent. While Chapter 3 and 4 investigate basic modeling and optimization of deterministic graph matching, Chapter 5 and 6 draw inspiration from previous chapters and seek to enhance the matching reliability under learning paradigm by taking into account intrinsic characteristics of graph matching (e.g., discreteness and topology).

In summary, this dissertation is presented to provide a novel and coherent study covering a series of essential problems of graph matching towards addressing the reliability issue, following the advances from deterministic optimization to combinatorial learning. Indeed, we are still facing other challenging issues in terms of reliability in graph matching and relevant combinatorial problems. However, with the development of theory, mathematical tools and availability of high-quality data, we anticipate that the research in this dissertation can inspire future investigation on related topics by any means.

7.1 Future Work

In this section, we discuss some potential future research directions related to graph matching from both theoretical and applicable perspectives. Such directions do not only consists of aspects in terms of reliability for graph matching, but also another key issues – scalability. While in this dissertation we mainly focus on reliability issue about noises, outliers and topology, here we pose another challenging problem to be investigated where only *partial observation* is available:

- A part of the graph information can be missing due to transmission failure. Besides, collecting ground-truth matching of graphs can be highly human-labor intensive and thus we sometimes can merely obtain partial data with moderate budget. In either case, GM problem becomes obscure under partial evidence. While taking into account that such cases can be encountered in real-world applications, it is demanding that corresponding theory and algorithms can be explored.

Scalability is another essential issue in parallel with reliability. The key of scalability is to find proper ways to extend current learning-based GM solvers to much larger graphs (million-level nodes) and facilitate simultaneous large-scale *multiple* graph co-matching (hundreds of graphs). We identify several directions which have rarely been investigated yet, to the best of our knowledge:

- QAP on pairwise graphs intrinsically suffers from scalability problem since it squares the variable size to be optimized. For example, existing GM solvers can be successfully applied to Pascal VOC dataset where the number of node is up to 100, but is extremely hard to be exploited to social networks in which number of nodes can easily exceeds thousands.

- Another consequence of scalability issue occurs when one wants to perform GM among many graphs simultaneously. Simultaneous matching on a large number of graphs may require intensive and dense computation, and can greatly hinder the wide use of GM techniques.

REFERENCES

Aberman, K., J. Liao, M. Shi, D. Lischinski, B. Chen and D. Cohen-Or, "Neural best-buddies: Sparse cross-domain correspondence", SIGGRAPH (2018).

Adamczewski, K., Y. Suh and K. Lee, "Discrete tabu search for graph matching", in "ICCV", (2015).

Adams, R. P. and R. S. Zemel, "Ranking via sinkhorn propagation", arXiv:1106.1925 (2011).

Alberich, R., A. Alcala, M. Llabrés, F. Rosselló and G. Valiente, "Alignet: alignment of protein-protein interaction networks", arXiv preprint arXiv:1902.07107 (2019).

Bai, Y., H. Ding, S. Bian, T. Chen, Y. Sun and W. Wang, "Simgnn: A neural network approach to fast graph similarity computation", in "WSDM", (2019).

Bengio, Y., N. Léonard and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation", arXiv preprint arXiv:1308.3432 (2013).

Bernard, F., C. Theobalt and M. Moeller, "Ds*: Tighter lifting-free convex relaxations for quadratic matching problems", in "CVPR", (2018).

Bishop, C. M., *Pattern recognition and machine learning* (springer, 2006).

Bojchevski, A., O. Shchur, D. Zügner and S. Günnemann, "Netgan: Generating graphs via random walks", in "ICML", (2018).

Bottou, L., "Large-scale machine learning with stochastic gradient descent", in "COMPSTAT", (2010).

Bourdev, L. and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations", in "ICCV", (2009).

Bromley, J., I. Guyon, Y. LeCun, E. Säckinger and R. Shah, "Signature verification using a "siamese" time delay neural network", in "NIPS", (1994).

Caetano, T., T. Caelli, D. Schuurmans and D. Barone, "Graphical models and point pattern matching", TPAMI **28**, 10, 1646–1663 (2006).

Caetano, T., J. McAuley, L. Cheng, Q. Le and A. J. Smola, "Learning graph matching", TPAMI **31**, 6, 1048–1058 (2009).

Campos, V., B. Jou, X. Giró-i Nieto, J. Torres and S.-F. Chang, "Skip rnn: Learning to skip state updates in recurrent neural networks", in "ICLR", (2018).

Chen, P., W. Liu, C.-Y. Hsieh, G. Chen and S. Zhang, "Utilizing edge features in graph neural networks via variational information maximization", arXiv preprint arXiv:1906.05488 (2019).

Chiasserini, C.-F., M. Garetto and E. Leonardi, "De-anonymizing clustered social networks by percolation graph matching", ACM Transactions on Knowledge Discovery from Data (TKDD) **12**, 2, 1–39 (2018).

Cho, K., B. Van Merriënboer, D. Bahdanau and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", arXiv preprint arXiv:1409.1259 (2014).

Cho, M., K. Alahari and J. Ponce, "Learning graphs to match", in "ICCV", (2013).

Cho, M., J. Lee and K. Lee, "Feature correspondence and deformable object matching via agglomerative correspondence clustering", in "ICCV", (2009).

Cho, M., J. Lee and K. M. Lee, "Reweighted random walks for graph matching", in "ECCV", (2010).

Cho, M. and K. M. Lee, "Progressive graph matching: Making a move of graphs via probabilistic voting", in "CVPR", (2012).

Choy, C. B., J. Gwak, S. Savarese and M. Chandraker, "Universal correspondence network", in "NIPS", (2016).

Chung, J., S. Ahn and Y. Bengio, "Hierarchical multiscale recurrent neural networks", in "ICLR", (2017).

De Cao, N. and T. Kipf, "Molgan: An implicit generative model for small molecular graphs", arXiv preprint arXiv:1805.11973 (2018).

Delaunay, B. *et al.*, "Sur la sphere vide", Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk **7**, 793-800, 1–2 (1934).

Duchenne, O., F. Bach, I.-S. Kweon and J. Ponce, "A tensor-based algorithm for high-order graph matching", TPAMI **33**, 12, 2383–2395 (2011).

Egozi, A., Y. Keller and H. Guterman, "A probabilistic approach to spectral graph matching", TPAMI (2013).

Erdos, P. and A. Renyi, "On random graphs i", in "Publicationes Mathematicae Debrecen 6", (1959).

Everingham, M., L. Gool, C. K. Williams, J. Winn and A. Zisserman, "The pascal visual object classes (voc) challenge", Int. J. Comput. Vision **88**, 2, 303–338 (2010a).

Everingham, M., L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, "The pascal visual object classes (voc) challenge", IJCV **88**, 2, 303–338 (2010b).

Fei-Fei, L., R. Fergus and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories", Computer Vision and Image Understanding **106**, 1, 59–70 (2007).

Fey, M., J. Eric Lenssen, F. Weichert and H. Müller, "Splinecnn: Fast geometric deep learning with continuous b-spline kernels", in "CVPR", (2018).

Fey, M., J. E. Lenssen, C. Morris, J. Masci and N. M. Kriege, "Deep graph matching consensus", in "ICLR", (2020).

Fink, A. M., "Hadamard's inequality for log-concave functions", Mathematical and Computer Modelling **32**, 5–6, 625–629 (2000).

Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, "Neural message passing for quantum chemistry", in "ICML", (2017).

Gold, S. and A. Rangarajan, "A graduated assignment algorithm for graph matching", TPAMI (1996).

Gómez-Bombarelli, R., J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams and A. Aspuru-Guzik, "Automatic chemical design using a data-driven continuous representation of molecules", ACS central science **4**, 2, 268–276 (2018).

Gong, L. and Q. Cheng, "Exploiting edge features for graph neural networks", in "CVPR", (2019).

Heimann, M., H. Shen, T. Safavi and D. Koutra, "Regal: Representation learning-based graph alignment", in "Proceedings of the 27th ACM International Conference on Information and Knowledge Management", pp. 117–126 (ACM, 2018).

Hsu, C.-H., S.-C. Hung, H. Chen, F.-K. Sun and Y.-W. Chang, "A dag-based algorithm for obstacle-aware topology-matching on-track bus routing", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2020).

Huang, J., M. Patwary and G. Diamos, "Coloring big graphs with alphagozero", arXiv:1902.10162 (2019).

Ibarrola, F. J., N. Ravikumar and A. F. Frangi, "Partially conditioned generative adversarial networks", arXiv preprint arXiv:2007.02845 (2020).

Jang, E., S. Gu and B. Poole, "Categorical reparameterization with gumbel-softmax", in "ICLR", (2017).

Jiang, B., J. Tang, C. Ding, Y. Gong and B. Luo, "Graph matching via multiplicative update algorithm", in "NIPS", pp. 3190–3198 (2017a).

Jiang, B., J. Tang, C. Ding and B. Luo, "Binary constraint preserving graph matching", in "CVPR", (2017b).

Johnson, D. D., "Learning graphical state transitions", in "ICLR", (2017).

Kipf, T. N. and M. Welling, "Variational graph auto-encoders", arXiv preprint arXiv:1611.07308 (2016).

Kipf, T. N. and M. Welling, “Semi-supervised classification with graph convolutional networks”, in “ICLR”, (2017).

Kool, W. and M. Welling, “Attention solves your tsp”, arXiv:1803.08475 (2018).

Krissinel, E. and K. Henrick, “Secondary-structure matching (ssm), a new tool for fast protein structure alignment in three dimensions”, Acta Crystallographica Section D: Biological Crystallography **60**, 12, 2256–2268 (2004).

Kuhn, H. W., “The hungarian method for the assignment problem”, in “Export. Naval Research Logistics Quarterly”, pp. 83–97 (1955).

Lawler, E. L., “The quadratic assignment problem”, Management Science pp. 586–599 (1963).

Leordeanu, M. and M. Hebert, “A spectral technique for correspondence problems using pairwise constraints”, in “ICCV”, (2005).

Leordeanu, M., M. Hebert and R. Sukthankar, “An integer projected fixed point method for graph matching and map inference”, in “NIPS”, (2009).

Leordeanu, M., R. Sukthankar and M. Hebert, “Unsupervised learning for graph matching”, IJCV pp. 28–45 (2012).

Li, Y., C. Fang, J. Yang, Z. Wang, X. Lu and M.-H. Yang, “Universal style transfer via feature transforms”, in “NIPS”, (2017).

Li, Y., D. Tarlow, M. Brockschmidt and R. Zemel, “Gated graph sequence neural networks”, in “ICLR”, (2016).

Lin, T.-Y., P. Goyal, R. Girshick, K. He and P. Dollár, “Focal loss for dense object detection”, in “ICCV”, (2017).

Loiola, E. M., N. M. de Abreu, P. O. Boaventura-Netto, P. Hahn and T. Querido, “A survey for the quadratic assignment problem”, EJOR pp. 657–90 (2007).

Lowe, D., “Object recognition from local scale-invariant features”, in “ICCV”, (1999).

Matas, J., O. Chum, M. Urban and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions”, Image and Vision Computing **22**, 10, 761–767 (2004).

Min, J., J. Lee, J. Ponce and M. Cho, “Spair-71k: A large-scale benchmark for semantic correspondence”, arXiv preprint arXiv:1908.10543 (2019).

Neal, R. M. and G. E. Hinton, “A view of the em algorithm that justifies incremental, sparse, and other variants”, in “Learning in graphical models”, pp. 355–368 (Springer, 1998).

Nowak, A., S. Villar, A. Bandeira and J. Bruna, “Revised note on learning quadratic assignment with graph neural networks”, in “DSW”, (2018).

Pan, S., R. Hu, G. Long, J. Jiang, L. Yao and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding", (2018).

Piegl, L. and W. Tiller, *The NURBS book* (Springer Science & Business Media, 2012).

Pogancic, M. V., A. Paulus, V. Musil, G. Martius and M. Rolínek, "Differentiation of black-box combinatorial solvers", in "ICLR", (2020).

Qiu, Z., T. Yao and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks", in "ICCV", (2017).

Rangarajan, A., A. Yuille and E. Mjolsness, "Convergence properties of the softassign quadratic assignment algorithm", Neural Computation (1999a).

Rangarajan, A., A. Yuille and E. Mjolsness, "Statistical physics algorithms that converge", Neural Computation **11**, 1455–1474 (1999b).

Ren, Z., J. Yan, B. Ni, B. Liu, X. Yang and H. Zha, "Unsupervised deep learning for optical flow estimation", in "AAAI", (2017).

Rolínek, M., P. Swoboda, D. Zietlow, A. Paulus, V. Musil and G. Martius, "Deep graph matching via blackbox differentiation of combinatorial solvers", in "ECCV", (2020).

Schellewald, C. and C. Schnörr, "Subgraph matching with semidefinite programming", Electronic Notes in Discrete Mathematics **12**, 279–289 (2003).

Schellewald, C. and C. Schnörr, "Probabilistic subgraph matching based on convex relaxation", in "EMMCVPR", (2005).

Schütt, K. T., F. Arbabzadah, S. Chmiela, K. R. Müller and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks", Nature communications **8**, 13890 (2017).

Shi, X., H. Ling, W. Hu, J. Xing and Y. Zhang, "Tensor power iteration for multi-graph matching", in "CVPR", (2016).

Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", in "ICLR", (2014).

Sinkhorn, R., "A relationship between arbitrary positive matrices and doubly stochastic matrices", The annals of mathematical statistics **35**, 2, 876–879 (1964).

Swoboda, P., A. Mokarian, C. Theobalt, F. Bernard *et al.*, "A convex relaxation for multi-graph matching", in "CVPR", (2019).

Swoboda, P., C. Rother, H. Abu Alhaija, D. Kainmuller and B. Savchynskyy, "A study of lagrangean decompositions and dual ascent solvers for graph matching", in "CVPR", (2017).

Tian, Y., J. Yan, H. Zhang, Y. Zhang, X. Yang and H. Zha, "On the convergence of graph matching: Graduated assignment revisited", in "ECCV", (2012).

Torr, P. H. S., "Solving markov random fields using semidefinite programmin", in "AISTATS", (2003).

Tran, D., H. Wang, L. Torresani, J. Ray, Y. LeCun and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition", in "CVPR", (2018).

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, "Graph Attention Networks", in "ICLR", (2018).

Wang, H., J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets", in "AAAI", (2018a).

Wang, Q., X. Zhou and K. Daniilidis, "Multi-image semantic matching by mining consistent features", in "CVPR", (2018b).

Wang, R., J. Yan and X. Yang, "Learning combinatorial embedding networks for deep graph matching", in "ICCV", (2019a).

Wang, R., J. Yan and X. Yang, "Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching", arXiv preprint arXiv:1911.11308 (2019b).

Wang, T., H. Ling, C. Lang and J. Wu, "Branching path following for graph matching", in "ECCV", (2016).

Weighill, D., M. B. Guebila, C. Lopes-Ramos, K. Glass, J. Quackenbush, J. Platig and R. Burkholz, "Gene regulatory network inference as relaxed graph matching", BioRxiv (2020).

Welinder, P., S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie and P. Perona, "Caltech-UCSD Birds 200", Tech. Rep. CNS-TR-2010-001, California Institute of Technology (2010).

Williams, R. J., "Simple statistical gradient-following algorithms for connectionist reinforcement learning", Machine Learning **8**, 3, 229–256 (1992).

Xiong, H. and J. Yan, "Btwalk: Branching tree random walk for multi-order structured network embedding", IEEE Transactions on Knowledge and Data Engineering (2020).

Xu, H., D. Luo, H. Zha and L. Carin, "Gromov-wasserstein learning for graph matching and node embedding", ICML (2019).

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

Yan, J., J. Wang, H. Zha, X. Yang and S. Chu, "Consistency-driven alternating optimization for multigraph matching: A unified approach", IEEE Transactions on Image Processing **24**, 3, 994–1009 (2015a).

Yan, J., S. Yang and E. R. Hancock, "Learning for graph matching and related combinatorial optimization problems", in "IJCAI", (2020).

Yan, J., C. Zhang, H. Zha, W. Liu, X. Yang and S. Chu, "Discrete hyper-graph matching", in "CVPR", (2015b).

Yu, T. and R. Wang, "Graph matching with low-rank regularization", in "WACV", (2016).

Yu, T., R. Wang, J. Yan and B. Li, "Learning deep graph matching with channel-independent embedding and hungarian attention", in "ICLR", (2020a).

Yu, T., R. Wang, J. Yan and B. Li, "Deep latent graph matching", in "ICML", (2021).

Yu, T., J. Yan and B. Li, "Determinant regularization for gradient-efficient graph matching", in "CVPR", (2020b).

Yu, T., J. Yan, W. Liu and B. Li, "Incremental multi-graph matching via diversity and randomness based graph clustering", in "ECCV", (2018a).

Yu, T., J. Yan, Y. Wang, W. Liu and B. Li, "Generalizing graph matching beyond quadratic assignment model", in "NeurIPS", (2018b).

Yu, T., J. Yan, J. Zhao and B. Li, "Joint cuts and matching of partitions in one graph", in "CVPR", (2018c).

Yuille, A. and J. Kosowsky, "Statistical physics algorithms that converge", Neural Computation **6**, 341–356 (1994).

Zanfir, A. and C. Sminchisescu, "Deep learning of graph matching", in "CVPR", (2018).

Zhang, S. and H. Tong, "Final: Fast attributed network alignment", in "Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", pp. 1345–1354 (ACM, 2016).

Zhang, Z. and W. S. Lee, "Deep graphical feature learning for the feature matching problem", in "ICCV", (2019).

Zhao, Z., Y. Qiao, J. Yang and L. Bai, "From dense subgraph to graph matching: A label propagation approach", in "2014 International Conference on Audio, Language and Image Processing", pp. 301–306 (IEEE, 2014).

Zhou, F. and F. Torre, "Factorized graph matching", TPAMI (2016).

Zhou, F. and F. D. Torre, "Factorized graph matching", in "CVPR", (2012).

Zhou, F. and F. D. Torre, "Deformable graph matching", in "CVPR", (2013).

Zhou, T., Y. J. Lee, S. X. Yu and A. A. Efros, "Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences", in "CVPR", pp. 1191–1200 (2015a).

Zhou, X., M. Zhu and K. Daniilidis, “Multi-image matching via fast alternating minimization”, in “ICCV”, (2015b).