Information Theoretic Graph Kernels

Lu Bai

A Thesis Submitted for the Degree of Doctor of Philosophy

Department of Computer Science
University of York

May 2014

Abstract

This thesis addresses the problems that arise in state-of-the-art structural learning methods for (hyper)graph classification or clustering, particularly focusing on developing novel information theoretic kernels for graphs.

To this end, we commence in Chapter 3 by defining a family of Jensen-Shannon diffusion kernels, i.e., the information theoretic kernels, for (un)attributed graphs. We show that our kernels overcome the shortcomings of inefficiency (for the unattributed diffusion kernel) and discarding un-isomorphic substructures (for the attributed diffusion kernel) that arise in the R-convolution kernels. In Chapter 4, we present a novel framework of computing depth-based complexity traces rooted at the centroid vertices for graphs, which can be efficiently computed for graphs with large sizes. We show that our methods can characterize a graph in a higher dimensional complexity feature space than state-of-the-art complexity measures. In Chapter 5, we develop a novel unattributed graph kernel by matching the depth-based substructures in graphs, based on the contribution in Chapter 4. Unlike most existing graph kernels in the literature which merely enumerate similar substructure pairs of limited sizes, our method incorporates explicit local substructure correspondence into the process of kernelization. The new kernel thus overcomes the shortcoming of neglecting structural correspondence that arises in most state-of-the-art graph kernels. The novel methods developed in Chapters 3, 4, and 5 are only restricted to graphs. However, real-world data usually tends to be represented by higher order relationships (i.e., hypergraphs). To overcome the shortcoming, in Chapter 6 we present a new hypergraph kernel using substructure isomorphism tests. We show that our kernel limits tottering that arises in the existing walk and subtree based (hyper)graph kernels.

In Chapter 7, we summarize the contributions of this thesis. Furthermore, we analyze the proposed methods. Finally, we give some suggestions for the future work.

Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to express my sincere appreciation and gratitude to my supervisor, Prof. Edwin R. Hancock, for his support and advice throughout my PhD work at York. I have learnt a lot from Edwin, such as how to formulate an academic problem in theoretical way and how to develop the research outcome in professional manner. I also thank my assessor, Dr. William Smith, for his impartial assessment and constructive comments on my work, and my co-authors, Prof. Horst Bunke, Prof. Francisco Escolano, Prof. Andrea Torsello, Prof. Peng Ren, Dr. Luca Rossi and Dr. Lin Han, for their help on my research. My thanks also go to Prof. Richard Wilson, Dr. Pablo Suau, Dr. Furqan Aziz and all friends at York for their academic suggestions, help, kindness and friendship.

There are some other researchers who have provided valuable help for my research. In this respect, I thank Prof. Karsten Borgwardt and Dr. Nino Shervashidze for providing the Matlab implementation for the various graph kernel methods, and Dr. Geng Li for providing the graph datasets.

Finally, I would like to dedicate this thesis to my family. Their help and support were invaluable for the successful completion of my PhD.

Declaration

I hereby declare that all the work in this thesis is solely my own, except where attributed and cited to another author. Most of the material in this thesis has been previously published by the author. Below, a complete list of publications can be found.

Journal Papers

1. Lu Bai, Luca Rossi, Andrea Torsello, and Edwin R. Hancock. A Quantum Jensen-Shannon Graph Kernel for Unattributed Graphs. To appear in *Pattern Recognition*, 2014.

2. Lu Bai, and Edwin R. Hancock. Depth-Based Complexity Traces of Graphs. In *Pattern Recognition*, vol. 47, no. 3, pp. 1172-1186, 2014.

3. Lu Bai, and Edwin R. Hancock. Graph Kernels from the Jensen-Shannon Divergence. In *Journal of Mathematical Imaging and Vision*, vol. 47, no. 1-2, pp. 60-69, 2013.

Conference Papers

1. Lu Bai, Luca Rossi, Horst Bunke and Edwin R. Hancock. Attributed Graph Kernels Using the Jensen-Tsallis q-Differences. To appear in *Proceedings of European*

Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2014.

2. Lu Bai, Peng Ren, Francisco Escolano, and Edwin R. Hancock. Depth-Based Hypergraph Complexity Traces from Directed Line Graphs. To appear in *Proceedings of 22nd International Conference on Pattern Recognition* (ICPR), 2014.

3. Lu Bai, Peng Ren, and Edwin R. Hancock. A Hypergraph Kernel from Isomorphism Test. To appear in *Proceedings of 22nd International Conference on Pattern Recognition* (ICPR), 2014.

4. Lu Bai, Horst Bunke, and Edwin R. Hancock. A Jensen-Shannon Diffusion Kernel for Attributed Graphs. To appear in *Proceedings of 22nd International Conference on Pattern Recognition* (ICPR), 2014.

5. Lu Bai, Peng Ren, Xiao Bai, and Edwin R. Hancock. A Graph Kernel from the Depth-Based Representation. To appear in *Joint IAPR International Workshops on Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition* (S+SSPR), 2014.

6. Andrea Torsello, Andrea Gasparetto, Luca Rossia, Lu Bai, and Edwin R. Hancock. Transitive State Alignment for the Quantum Jensen-Shannon Kernel. To appear in *Joint IAPR International Workshops on Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition* (S+SSPR), 2014.

7. Lu Bai, Edwin R. Hancock, and Lin Han. A Graph Embedding Method using the Jensen-Shannon Divergence. In *Proceedings of 15th International Conference on Computer Analysis of Images and Patterns* (CAIP), pp. 102-109, 2013.

8. Lu Bai, and Edwin R. Hancock. A Fast Jensen-Shannon Subgraph Kernel. In *Proceedings of 17th International Conference on Image Analysis and Processing* (ICIAP), pp. 181-190, 2013.

9. Lu Bai, Edwin R. Hancock, Andrea Torsello, and Luca Rossi. A Quantum Jensen-Shannon Graph Kernel using the Continuous-Time Quantum Walk. In *Proceedings of 9th IAPR-TC-15 International Workshop on Graph-Based Representations in Pattern Recognition* (GbRPR), pp. 121-131, 2013.

10. Lu Bai, and Edwin R. Hancock. Graph Clustering using Graph Entropy Complexity Traces. In *Proceedings of 21st International Conference on Pattern Recognition* (ICPR), pp. 2881-2884, 2012.

11. Lu Bai, Edwin R. Hancock, and Peng Ren. Jensen-Shannon Graph Kernel using Information Functionals. In *Proceedings of 21st International Conference on Pattern Recognition* (ICPR), pp. 2877-2880, 2012.

12. Lin Han, Richard C. Wilson, Edwin R. Hancock, Lu Bai, and Peng Ren. Sampling Graphs from A Probabilistic Generative Model. In *Proceedings of 21st International Conference on Pattern Recognition* (ICPR), pp. 1643-1646, 2012.

13. Lu Bai, Edwin R. Hancock and Peng Ren. A Jensen-Shannon Kernel for Hypergraphs. In *Proceedings of Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition* (S+SSPR), pp. 181-189, 2012.

14. Lu Bai, and Edwin R. Hancock. Graph Complexity from the Jensen-Shannon Divergence. In *Proceedings of Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition* (S+SSPR), pp. 79-88, 2012.

15. Lu Bai, and Edwin R. Hancock. Graph Clustering Using the Jensen-Shannon K-ernel. In *Proceedings of 14th International Conference on Computer Analysis of Images and Patterns* (CAIP), pp. 394-401, 2011.

Submitted Journal Papers

1. Lu Bai, and Edwin R. Hancock. Fast Depth-based Subgraph Kernels for Unattributed Graphs. Submitted to *Pattern Recognition*.

2. Lu Bai, Peng Ren, Francisco Escolano, Cheng Ye, and Edwin R. Hancock. Depth-based Hypergraph Complexity Traces from Directed Line Graphs. Submitted to *IEEE Transactions on Neural Networks and Learning Systems*.

Chapter 1

Introduction

In this chapter we provide an introduction and motivation for the research work presented in this thesis, explaining why we are interested in kernel methods for structured data (i.e., graphs and hypergraphs). We commence by introducing the problems encountered in existing state-of-the-art methods on structured data. Then we briefly describe the possible alternative approaches that overcome these problems, following by our research goals and contributions. Finally, an outline of the thesis is provided at the end of this chapter.

1.1 The Problems

Graph based relational representations, which are widely used in the field of structural pattern recognition, have proven to be both powerful and flexible. Compared to vector based pattern recognition, a major drawback with graph representations is the lack of a natural correspondence order for the vertices or edges. This limits the direct application of standard machine learning algorithms to problems such as clustering or classifying graphs. One way to overcome this problem is to embed graphs into a vector space, where standard machine learning techniques can be employed. Specifically, in the embedding space, similar graph structures are expected to be close while dissimilar ones far apart.

However, the vector space embedding presents two obstacles. First, since graphs can be of different sizes, the vectors may be of different lengths. The second problem is that some information residing on the edges of a graph is discarded. In order to overcome these problems, Riesen and Bunke recently proposed a method for embedding graphs into a vector space [1] that bridges the gap between the powerful graph based representation and the algorithms available for the vector based representation. The ideas underpinning graph dissimilarity embedding framework were first described in Duin and Pekalska's work [2]. Riesen and Bunke generalized and substantially extended the methods to the graph mining domain. The key idea is to use the edit distance from a sample graph to a number of class prototype graphs to give a vectorial description of the sample graph in the embedding space. Furthermore, this approach potentially allows any (dis)similarity measure of graphs to be used for graph (dis)similarity embedding as well. Unfortunately, the edit distance between a sample graph and a prototype graph requires expensive computations, and as a result the graph dissimilarity embedding using the edit distance can not be efficiently computed for graphs.

Other successful approaches that embed graphs into vectors include a) representing a graph structure using permutation invariant polynomials computed from the eigenvectors of the Laplacian matrix based on algebraic graph theory [3], and b) computing permutation-invariant graph features via the Ihara zeta function [4]. Unfortunately, the computation of these methods also tends to be expensive. Because the characteristics values highly rely on the graph size, and tend to be infinite valued with graphs of large sizes (e.g., graphs having more than 500 vertices). Furthermore, all these graph embedding methods tend to approximate structural correlations of graphs in a low dimensional pattern space, and thus lead to information loss.

To address the shortcomings that arise in the graph embedding methods, an interesting recent addition to the literature is to use graph kernels [5]. Graph kernels can characterize graph features in a high dimensional space and thus better preserve graph structures. Most existing graph kernels (i.e., the R-convolution kernels) can be generally categorized into three classes [6, 7], i.e., graph kernels based on comparing all pairs of a) walks, b) paths, and c) restricted subgraph and subtree structures. There are mainly three shortcomings arising in the R-convolution kernels. First, the R-convolution kernels do not easily scale up to large sized structures, and thus compromise to use limited sized substructures. Unfortunately, graph kernels with substructures of limited sizes tend to reflect restricted topological information of a graph. Second, the R-convolution kernels roughly compare any pair of isomorphic substructures, and thus ignore the relative locations between the substructures within a graph. Third, the R-convolution kernels only count the number of pairwise isomorphic substructures. As a result, the substructures having no corresponding isomorphic substructures are discarded. Generally speaking, these shortcomings limit the precise similarity measure (i.e., the kernel value) for graphs. Therefore, it is fair to say that developing efficient and effective graph kernels still remains a challenge.

Furthermore, complex data in real world tends to exhibit multiple relationships that are hard to characterize by using graphs. To overcome this problem, hypergraph based strategies have been investigated for representing and processing structures where the relations present are of higher order. A hypergraph is a generalization of a graph [8]: unlike the pairwise edges in a graph, hypergraph representations allow a hyperedge to encompass an arbitrary number of vertices, and can hence capture multiple relationships among features. There have been several successful methods for characterizing hypergraphs, which include a) marginalizing higher order relationships to unary order [9], b) marginalizing the higher order relationships to pairwise order and then adopting pairwise graph matching methods [10], c) performing visual clustering by adopting tensors for representing uniform hypergraphs [11], and d) exploiting a set of coefficients from hypergraph Ihara zeta function to capture frequency of the cycle structures in a hypergraph [12]. One main limitation of the existing methods for hypergraph characterization is that they are usually limited to uniform structures, and do not fully capture hypergraph characteristics. On the other hand, existing hypergraph characterization methods also tend to require prohibitive computational overheads. In order to overcome these problems, an attractive alternative is to use kernel methods. Wachman and Khardon [13] have summarized the existing graph kernels based on walks and then proposed a rooted kernel for hypergraphs. Unfortunately, like the walk based graph kernels, the rooted hypergraph kernel also suffers from the notorious tottering problem. This occurs when a random walk on a hypergraph moves to one direction and then immediately returns to the starting position through the same vertices and hyperedges multiple times. As a result, tottering may result in many redundant paths in a hypergraph. This shortcoming limits the precise kernel measure between hypergraphs.

1.2 Motivations and Our Goals

The main goal of this thesis is to develop novel methods that address the problems encountered in the mentioned state-of-the-art methods. Specifically:

I) We present novel information theoretic kernels, i.e., a family of Jensen-Shannon diffusion kernels, for (un)attributed graphs based on the Jensen-Shannon divergence. We will show how the unattributed diffusion kernel can overcome the inefficiency of the R-convolution kernels. Furthermore, we will show how the attributed diffusion kernel can overcome the shortcoming of discarding substructures having no corresponding isomorphic substructures that arises in the R-convolution kernels.

II) We present a novel framework for characterizing graphs based on computing depth-based complexity traces. We will show that the complexity traces can, not only reflect high dimensional complexity characteristics of graphs, but also be computed efficiently for graphs of large sizes.

III) We develop a novel unattributed graph kernel (i.e., the depth-based matching kernel) based on the depth-based representations of graphs and a graph matching using these representations. We will show that the new depth-based matching kernel can, not only reasonably reflect depth-based characteristics of graphs, but also overcome the shortcoming of neglecting structural correspondence that arises in the R-convolution kernels.

IV) We develop a new hypergraph kernel based on a new developed Weisfeiler-Lehman isomorphism test for directed graphs. We will show that our new hypergraph kernel can limit the tottering problem that arises in the existing walk and subtree based (hyper)graph kernels.

1.3 Contributions

To achieve the research goals described in Section 1.2, we make the following specific contributions.

1.3.1 Jensen-Shannon Diffusion Kernels for (Un)attributed Graphs

In Chapter 3 we propose a family of Jensen-Shannon diffusion kernels for (un)attributed graphs using the Jensen-Shannon divergence. In mutual information, the Jensen-Shannon divergence is a dissimilarity measure between probability distributions in terms of the entropy difference associated with the probability distributions [14]. To develop a novel kernel using the divergence measure, we require an entropy measure for each graph. To this end, for an unattributed graph, we compute the von Neumann entropy developed by Han et al. [15]. Furthermore, we also develop a new Shannon entropy associated with a steady state random walk for the undirected graph. For an attributed graph, we perform a tree-index method developed by Dahm et al. [16] for the purpose of strengthening the vertex labels. For a vertex, the tree-index method strengthens the vertex label by taking the union of the neighbouring vertex labels as lists. Unfortunately, this tree-index method tends to lead a rapid explosion of the strengthened label length. Moreover, strengthening a vertex label by only taking the union of the neighbouring label lists also tends to ignore the original label information of the vertex. To overcome these problems and improve the tree-index method, we propose to strengthen the label of a vertex as a new label list by taking the union of both the original vertex label and its neighbouring vertex labels. We also use the Hash function for the purpose of compressing the strengthened label list into a new short index label. As a result, we compute a new label Shannon entropy for the attributed graph in terms of the frequency of the strengthened labels. With a pair of (un)attributed graphs and their entropies to hand, the diffusion kernel for the graphs can be computed using the Jensen-Shannon divergence between a composite entropy of the graphs and their individual entropies.

We show the advantages of our Jensen-Shannon diffusion kernels for (un)attributed graphs. For the unattributed Jensen-Shannon diffusion kernel, the required entropy for a graph can be computed without the need to decompose the graph. As a result, the unattributed diffusion kernel overcomes the inefficiency arising in the R-convolution kernels. On the other hand, for the attributed Jensen-Shannon diffusion kernel, each strengthened vertex label corresponds to a subtree rooted at the vertex. Furthermore, each of the strengthened labels is used for computing the label Shannon entropy. As a result, unlike existing R-convolution kernels which count the number of pairwise isomorphic substructures, our method incorporates all of the identified subtrees into the computation of the kernel. The new attributed diffusion kernel thus overcomes the shortcoming of discarding substructures having no corresponding isomorphic substructures.

We study the performance for either the unattributed or the attributed Jensen-Shannon diffusion kernel on several graph datasets abstracted from bioinformatics databases. We show that our new Jensen-Shannon diffusion kernel for unattributed graphs can easily outperform the existing state of the art graph kernels in terms of computational efficiency. Moreover, we show that our new Jensen-Shannon diffusion kernel for attributed graphs can easily outperform the existing state of the art graph kernels in terms of the classification accuracy.

1.3.2 Depth-based Complexity Traces of Graphs

In Chapter 4, we propose a novel framework to compute the depth-based complexity traces for graphs by linking the ideas of graph entropies and depth-based representations. The depth-based representations of undirected graph structures have been proven powerful tool for characterizing the topological structure in terms of the intrinsic complexity [17, 18]. One approach is to gauge the information content flow through a family of K layer subgraphs of a graph (e.g., subgraphs around a vertex having a maximum topology distance or minimal path length K) of increasing size and to use the flow as a structural signature. Furthermore, this approach allows a complexity trace to be defined which gauges how the complexity of the graph varies as a function of depth. Unfortunately, to construct such a complexity trace of a graph requires a burdensome measure of intrinsic structural complexity, e.g., the time complexity of the heat flow complexity measure on a subgraph having n vertices is $O(n^5)$. Moreover, straightforwardly constructing a complexity trace that characterizes a graph structure in global manner is an elusive problem, since it is difficult to determine a fine root vertex in the graph.

To overcome the problems, we focus on developing an efficient depth-based signature, that can both capture fine structure and be evaluated relatively efficiently. To locate dominant substructures within a graph, we commence by identifying a centroid vertex which has the minimum shortest path length variance to the remaining vertices. For each graph a family of centroid expansion subgraphs is derived from the centroid vertex in order to capture dominant structural characteristics of the graph. Since the centroid vertex is identified through a global analysis of the shortest path length distribution, the expansion subgraphs provide a fine representation of a graph structure. We then compute the depth-based complexity traces of a graph based on two strategies. The first strategy is to establish an entropy complexity trace by measuring how the entropies on the centroid expansion subgraphs vary with the increasing layer size of the expansion subgraphs. While the second strategy is that we construct a complexity trace by measuring how the differences between the subgraphs and the graph vary with respect to the increasing layer size of the expansion subgraphs. Since the required entropy measures on the condensed subgraph family enable efficient complexity computation, the complexity traces resulting from the two strategies can be constructed efficiently. We empirically demonstrate that depth-based complexity traces of graphs can easily scale up to large graphs. The performance of our framework is competitive to the state-of-the-art graph based learning methods in the literature.

1.3.3 A Depth-Based Matching Kernel for Unattributed Graphs

In Chapter 5, we investigate how to incorporate the depth-based representations into graph matching and thus develop a novel graph kernel for unattributed graphs. We commence by generalizing the depth-based complexity trace around the centroid vertex that was developed in Chapter 4. We compute the complexity traces of a graph around each vertex. To avoid the inefficient subgraph enumeration in computing the intrinsic complexity [17], we compute the depth-based representation around a vertex by measuring the Shannon entropy on each of its expansion subgraphs associated with the steady state random walk. The depth-based representation gauges the Shannon entropy flow via the expansion subgraphs, and thus reflects a high dimensional complexity characteristics of the graph around the vertex. Based on the obtained depth-based representations for two graphs we develop a matching strategy similar to that developed by Scott et al. in [19], for point set matching. The purpose of this step is to match the vertices of the graphs by using the vertex information extracted from the depth-based representations. For a pair of graphs, we use the Euclidean distance between the depth-based representations to compute an affinity matrix. The correspondences between pairwise vertices are obtained from the affinity matrix. The affinity matrix characterizes local structural similarity between a pair of graphs and can be used for graphs of different sizes. Finally, we develop the novel depth-based graph matching kernel by counting the matched vertex pairs. We show that the novel kernel is positive definite. Furthermore, we show the relationship between the depth-based graph kernel and the all subgraph kernel and thus explain the reasons for the effectiveness of the new graph kernel. We empirically demonstrate the effectiveness and efficiency of our new graph kernel on graphs from computer vision databases.

1.3.4 A Hypergraph Kernel from The Subtree Isomorphism Tests

The family of Jensen-Shannon diffusion kernels proposed in Chapter 3, the depth-based complexity traces proposed in Chapter 4, and the depth-based matching kernel proposed in Chapter 5 are only restricted to graphs. However, real-world data usually tends to be represented by higher order relationships (i.e., hypergraphs). To address the limitation, in Chapter 6, we propose a new hypergraph kernel based on isomorphic substructures. This is facilitated by a Weisfeiler-Lehman (WL) isomorphism test for undirected graphs [20]. The WL isomorphism test for undirected graphs is a canonical labeling method. The key idea of the WL isomorphism test is to strengthen the set of vertex labels by the labels of the set of neighbouring vertices (i.e., strengthen a vertex label using a tree-index method). Here each new label of a vertex corresponds a subtree rooted from the vertex. Since the computational complexity of the WL isomorphism test on an undirected graph is only linear in the number of edges or quadratic in the number of vertices, the isomorphism test offers an elegant way of defining a fast graph kernel. Shervashidze and Borgwardt [7] have defined a fast subtree kernel (i.e., the WL subtree kernel) for undirected graphs by performing the WL isomorphism test to update the vertex labels, and then counting the number of matched vertex labels (i.e., counting the number of pairwise isomorphic subtrees).

Unfortunately, straightforwardly measuring the WL isomorphism test for hypergraphs tends to be elusive since a hypergraph may exhibit various relational orders. To overcome the mentioned problems and construct a hypergraph kernel using the WL isomorphism test, we transforme a hypergraph into a directed line graph using the Perron-Frobenius operator [12]. The Perron-Frobenius operator accurately reflects the multiple relationships exhibited by both uniform and nonuniform hypergraphs, moreover it also represents a directed backtrackless structure for (hyper)graph representations [21]. Hence, the directed line graph for a hypergraph representation provides not only a convenient framework for measuring isomorphisms but also an elegant way of limiting the tottering problem arising in the walk based (hyper)graph kernels [13, 22].

We propose a new directed WL isomorphism test on directed graphs for the purpose of measuring the isomorphism between hypergraphs. The directed isomorphism test is based on two steps. The first is to assign a vertex a new in-label using the in-degree of the vertex and that of its in-neighborhood. The second is to assign a vertex a new out-label using the out-degree of the vertex and that of its out-neighborhood. As a result, the proposed kernel for a pair of hypergraphs is defined by performing the new directed WL isomorphism test to update the in-labels and out-labels of their directed line graphs, and then counting the number of newly matched in-labels and out-labels (i.e., counting the number of pairwise isomorphic in-subtrees and out-subtrees). We empirically demonstrate the effectiveness and efficiency of our new hypergraph kernel on several challenging (hyper)graph datasets.

1.4 Thesis Outline

The rest of the thesis is organized as follows: In Chapter 2, we give a thorough review of the relevant literature. First, we review the concepts of kernel methods in pattern recognition. These methods include a) the kernels for vectorial data, b) the information theoretic kernels for probability distributions over structured data, and c) the kernels for graphs (i.e., graph kernels). Second, we review some state-of-the-art complexity measures for graphs. Finally, we review hypergraph representations in pattern recognition. In Chapter 3, we propose a family of Jensen-Shannon diffusion kernels for (un)attributed graphs, by using the Jensen-Shannon divergence. In Chapter 4, we propose a novel framework to compute depth-based complexity traces of graphs. In Chapter 5, we propose a novel depth-based matching kernel for unattributed graphs. The kernel is based on the depth-based representations of graphs and a graph matching using the representations. In Chapter 6, we propose a new hypergraph kernel based on a new developed directed Weisfeiler-Lehman isomorphism test for directed graphs. Finally, in Chapter 7 we summarize the contributions of this thesis and suggest avenues for future work.

Chapter 2

Literature Review

Graphs are important representations in the field of structural pattern recognition. To address the problems of existing state-of-the-art methods mentioned in Section 1.1, we aim to develop novel methods that can be effectively and efficiently performed on graphs for the objective of graph classification or clustering. To achieve this, we focus in more details on using the kernel methods and entropy based complexity measures for graphs.

In the light of this aim, we commence in Section 2.1 by reviewing kernel theory in pattern recognition. We introduce the concepts of kernels for vectorial data in general, and for graphs in particular. Furthermore, we discuss the strength and weakness for some state-of-the-art graph kernels. We explain our motivation of defining novel kernel methods for graphs. In Section 2.2, we review several state-of-the-art deterministic and probabilistic complexity measures for graphs. We discuss the strength and weakness of the previous researches on quantification of these complexity measures for (un)directed graphs. Furthermore, we review an attractive alternative complexity measure for undirected graphs, namely the thermodynamic depth-based complexity. We point out that this approach allows a depth-based complexity trace to be defined as a function of depth. We discuss the problem of constructing such a complexity trace for a graph. Finally, in this section (i.e., Section 2.2), we explain our motivation for computing an efficient and effective depth-based complexity trace of a graph by linking the ideas of the depth-based complexity and the entropy measures.

Complex data in the real world tends to exhibit multiple relationships that are hard to characterize by using graphs. Thus, in Chapter 2.3 we also review hypergraph representations in pattern recognition. We review some state-of-the-art hypergraph based learning methods. We explain our motivation for computing a new hypergraph kernel based on isomorphism tests. Finally, in Section 2.4, we conclude this chapter.

2.1 Kernel Methods

In pattern recognition and machine learning, kernel methods have been widely used in kernel machines (e.g., the Support Vector Machine (SVM) [23] and the kernel Principle Component Analysis (kPCA) [24]) for classification or clustering. Typical applications include a) image classification [1], b) protein prediction [25], c) handwriting recognition [26], and d) molecule classification [13]. The main advantages of using kernel methods are threefold [25]. First, kernel methods allow efficient algorithms to be developed that can deal with high dimensional data without the need of constructing an explicit high dimensional feature space. Second, kernel methods provide an elegant way of making standard machine learning methods for vectorial data applicable to more complex data (e.g, strings, trees, graphs and hypergraphs), and thus bridge the gap between statistical and structural pattern recognition. Third, kernel methods allow us to extend linear algorithms to non-linear ones.

This subsection provides a general introduction to kernel methods. To this end, we commence by reviewing the kernel methods. Some basic concepts and properties of kernel methods are introduced. Furthermore, we review a family of alternative kernels on probability distributions over structured data, namely the information theoretic kernels. Finally, we review some state-of-the-art graph kernels. We explain the motivation of developing novel graph kernel methods.

2.1.1 Kernel Functions

In this subsection, we review some basic concepts of kernel methods. We commence by introducing the concept of a positive definite kernel, based on the definition from Schölkopf et al. [27]. Let $\mathcal{X}$ denote a nonempty pattern set. A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric function, i.e., $k(y_p, y_q) = k(y_q, y_p)$, that maps the pair of patterns y_p and y_q to a real value. The kernel function k is called a positive definite (**pd**) kernel if and only if

$$\sum_{p,q=1}^{N} c_p c_q k(y_p, y_q) \geq 0, \tag{2.1}$$

for all N, $\{c_1, \ldots, c_N\} \subseteq \mathbb{R}$, and any choice of $\{y_1, \ldots, y_N\} \subseteq \mathcal{X}$.

Note that, a positive definite kernel function is usually called a valid kernel [28]. Moreover, a kernel for a pair of patterns can be seen as a similarity measure between the patterns. In the literature [29], some standard kernel functions have been developed for the case where $\mathcal{X}$ is a vector space. Examples include a) the linear kernel, b) the RBF kernel, c) the Polynomial kernel, and d) the Sigmoid kernel. For a pair of vectors $y_p, y_q \in \mathcal{X}$ in a vector space $\mathcal{X}$, these kernels are defined as follows.

1. **Linear kernel:**

$$k_{\langle\rangle}(y_p, y_q) = \langle y_p, y_q \rangle. \tag{2.2}$$

2. **RBF kernel:**

$$k_{RBF}(y_p, y_q) = \exp(-\gamma \|y_p - y_q\|^2), \tag{2.3}$$

where $\gamma > 0$.

3. **Polynomial kernel:**

$$k_{poly}(y_p, y_q) = (\langle y_p, y_q \rangle + c)^d, \tag{2.4}$$

where $d \in \mathbb{N}$ and $c \geq 0$.

4. **Sigmoid kernel:**
$$k_{sig}(y_p, y_q) = \tanh(\alpha\langle y_p, y_q\rangle + \beta), \tag{2.5}$$
where $\alpha > 0$ and $\beta < 0$.

Note that, the linear kernel, the RBF kernel, and the Polynomial kernel are positive definite (see details in [29]). On the other hand, the Sigmoid kernel is not always valid. However, the Sigmoid kernel has nonetheless been successfully used for real-world applications [25]. Furthermore, if α is close to zero and β is small enough, the Sigmoid kernel tends to behave similar to the RBF kernel [25].

In fact, from these given kernels, we can also compute some more sophisticated kernels that better represent the data, based on the closure properties [25]. Assume k_1 and k_2 are two valid kernels on $\mathcal{X} \times \mathcal{X}$, k_3 is a valid kernel on $\mathcal{H} \times \mathcal{H}$, $\varphi : \mathcal{X} \to \mathcal{H}$ and $f : \mathcal{X} \to \mathbb{R}$ are two mappings, and $a \in \mathbb{R}^+$. The kernel functions defined by a) $k(y_p, y_q) = k_1(y_p, y_q) + k_2(y_p, y_q)$, b) $k(y_p, y_q) = k_1(y_p, y_q)k_2(y_p, y_q)$, c) $k(y_p, y_q) = ak_1(y_p, y_q)$, d) $k(y_p, y_q) = f(y_p)f(y_q)$, and e) $k(y_p, y_q) = k_3(\varphi(y_p), \varphi(y_p))$ are also valid kernels.

Next, we show how the kernel matrix can be computed using a given kernel function k. Let $\{y_1, \ldots, y_N\} \subseteq \mathcal{X}$ be a training set having N patterns. The kernel matrix $\mathbf{K}$ is a $N \times N$ square matrix,
$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & \ldots & k_{1N} \\ k_{21} & k_{22} & \ldots & k_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ k_{N1} & k_{N2} & \ldots & k_{NN} \end{bmatrix}, \tag{2.6}$$
where each element $k_{p,q}$ ($1 \leq p, q \leq N$) is a real number given by $k_{p,q} = k(y_p, y_q)$.

This matrix plays a central role in kernel-based methods (e.g., the kernel machines SVM [23] and kPCA [24]), since all the information available to a kernel-based method is contained in $\mathbf{K}$ [25]. As a result, the kernel matrix $\mathbf{K}$ is an interface between the pattern space $\mathcal{X}$ and the kernel based method.

2.1.2 Information Theoretic Kernels

There has recently been an increasing interest in positive definite kernels for probability distributions [30]. By mapping each data point in the input space $\mathcal{X}$ to a fitted distribution in a parametric family S, where a kernel for the distributions may be defined, a kernel for the data points in terms of the distributions can be automatically induced on the original input space. This framework provides us an alternative way of defining kernels that map data (e.g., structured data) to a statistical manifold [31]. In real-world applications, these kernels outperform SVM classifiers associated with linear kernels [32]. Some of these kernels create a bridge between kernel methods and information theory, and thus have an information theoretic interpretation [33, 34].

In [30], Martins et al. have reinforced the bridge by developing a new family of nonextensive information theoretic kernels for probability distributions over structured data using nonextensive entropies. The extensive entropy is such that the entropy for different variables is additive over the independent variables. Assume X is a random variable that takes values in a finite set $\mathcal{X}$ based on a probability distribution P_X. An entropy, e.g., the Shannon entropy H_S and $H_S(X) \triangleq \mathbb{E}[\ln P_X]$, is extensive: if X and Y are independent, then $H_S(X, Y) = H_S(P_X) + H_S(P_Y)$. By contrast, the nonextensive entropy abandons the requirement of extensivity. The idea of nonextensive entropy was first introduced by Havrda and Charvit [35], and its applications have been extensively documented by Gell-Mann and Renyi [36].

Some of the nonextensive entropic kernels (i.e., the nonextensive information theoretic kernels) are related to the Jensen-Shannon divergence, that is a mutual information dissimilarity measure between probability distributions in terms of the entropy difference associated with the probability distributions. Examples include a) the Jensen-Shannon kernel [34], b) the exponentiated Jensen-Shannon kernel [37], c) the weighed Jensen-Shannon kernel [30], and d) the exponentiated weighed Jensen-Shannon kernel [30]. All these kernels have been performed for text categorization applications that demonstrate their effectiveness.

We are interested in computing an information theoretic kernel for graphs from the Jensen-Shannon divergence measure. The problems of computing the divergence based kernel for graphs are those of constructing the required probability distributions and computing their associated entropies. As a result, the kernel for a pair of graphs can be defined as the difference between the entropies of the graphs and that of a composite graph formed by the graphs, using the Jensen-Shannon divergence. Unfortunately, each of these problems has been proved elusive, and thus leads a serious obstacle to the successful construction of information theoretic graph kernels.

2.1.3 Graph Kernels

Most existing graph kernels are instances of the R-convolution kernel [38] proposed by Haussler. R-convolution is a generic way for defining graph kernels by comparing all pairs of isomorphic substructures under decomposition (e.g., graph kernels based on comparing all pairs of a) walks, b) paths, and c) restricted substructures). Different types of decompositions will result in a new graph kernel. For a pair of graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, suppose $\{\mathcal{S}_{1;1}, \ldots, \mathcal{S}_{1;n_1}, \ldots, \mathcal{S}_{1;N_1}\}$ and $\{\mathcal{S}_{2;1}, \ldots, \mathcal{S}_{2;n_2}, \ldots, \mathcal{S}_{q;N_2}\}$ are the sets of the substructures of G_1 and G_2 respectively. An R-convolution kernel k_R [38] between G_1 and G_2 can be defined as

$$k_R(G_1, G_2) = \sum_{n_1=1}^{N_1} \sum_{n_2=1}^{N_2} \delta(\mathcal{S}_{1;n_1}, \mathcal{S}_{2;n_2}), \tag{2.7}$$

where

$$\delta(\mathcal{S}_{1;n_1}, \mathcal{S}_{2;n_2}) = \begin{cases} 1 & \text{if } \mathcal{S}_{1;n_1} \simeq \mathcal{S}_{2;n_2}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

δ is the Kronecker delta, that is, it is 1 if the arguments are equal and 0 otherwise. It can be shown that k_R is a positive definite kernel [38].

With this scenario, Kashima et al. [22] have proposed a random walk kernel by comparing pairs of isomorphic random walks in a pair of graphs. The main drawback of the random walk kernel is the notorious tottering problem. This occurs when a random walk on a graph moves in one direction and then immediately returns to the starting position through the same vertices and edges possibly multiple times. To overcome this shortcoming, Borgwardt et al. [39] have proposed a shortest path kernel by counting the numbers of pairwise shortest paths having the same length in a pair of graphs. Aziz et al. [21] have defined a backtrackless kernel using the same length cycles in a pair of graphs. Both of the methods overcome the tottering problem by capturing backtrackless substructures (i.e., the shortest paths or cycles in graphs). Unfortunately, shortest paths and cycles are structurally simple, and reflect limited topology information. Moreover, the computational efficiency of the two kernels also tends to be burdensome (i.e., the runtime may be a couple of days) for graphs of large sizes (e.g. a graph having more than one thousand vertices). To address the problem of inefficiency, Shervashidze et al. [7] have developed a fast subtree kernel by comparing pairs of subtrees identified by the Weisfeiler-Lehman algorithm. Costa and Grave [40] have defined a neighborhood subgraph pairwise distance kernel by counting the number of pairwise isomorphic neighborhood subgraphs in a pair of graphs. Both kernels can be computed in polynomial time. Other graph kernels that are developed from the R-convolution framework include a) the segmentation graph kernel developed by Harchaoui and Bach [41], b) the point cloud kernel developed by Bach [42], and c) the subgraph matching kernel developed by Kriege and Mutzel [43].

Unfortunately, there are three common shortcomings arising in the existing R-convolution kernels. First, the R-convolution kernels may discard the substructures having no corresponding isomorphic substructures when a pair of graphs are compared. This occurs when we compute a kernel by simply counting the number of isomorphic substructure pairs. In other words, the R-convolution kernel may discard some information residing on the structural arrangement of graphs. Second, the R-convolution kernels do not easily scale up to structures of large sizes. Thus, these kernels compromise to use substructures of limited sizes. Although this strategy curbs the notorious inefficiency of comparing large substructures, these kernels still require significant computational overheads. Moreover, graph kernels with limited sized substructures can only reflect restricted topological characteristics of a graph. Third, the R-convolution kernels do not indicate the relative locations between substructures within a graph, and thus cannot establish a precise structural correspondence between vertices of a pair of graphs. Generally speaking, these shortcomings limit the precise similarity measure (i.e., the kernel value) for graphs.

We aim to propose novel information theoretic graph kernels addressing the shortcomings of state-of-the-art graph kernels. To this end, we consider to propose a family of Jensen-Shannon diffusion kernels for (un)attributed graphs by measuring the Jensen-Shannon divergence between graph entropies. For a pair of unattributed graphs, the computational complexity of the diffusion kernel is only quadratic in the vertex number of the larger graph. The Jensen-Shannon diffusion kernel for unattributed graphs thus overcomes the inefficiency arising in the R-convolution kernels. On the other hand, for a pair of attributed graphs, we strengthen the vertex labels for each graph by using a tree-index method, and each strengthened vertex label corresponds to a subtree. The required Shannon entropy for an attributed graph is computed in terms of all the strengthened labels (i.e., all the identified subtrees from the tree-index method), the Jensen-Shannon diffusion kernel for attributed graphs thus overcomes the shortcoming of discarding un-isomorphism substructures that arises in the R-convolution kernels. Furthermore, we also consider using depth-based representations [17] as a means of defining a new depth-based matching kernel for graphs. The depth-based representation is a powerful tool that can characterize a graph in terms of rich complexity information (see details in Section 2.2.2). On the other hand, the resulted depth-based matching reflects the correspondence information between pairwise vertices. Thus, the depth-based matching kernel can overcome the shortcomings of using limited substructures and neglecting local substructure correspondence that arise in the R-convolution kernels.

2.2 Complexity Measures of Graphs

In this section, we start by reviewing the concepts of the deterministic and probabilistic complexity measures for graphs. We focus more on the entropy-based complexity measures. Moreover, we also review an alternative complexity measure for graphs, namely the thermodynamic depth-based complexity measure for graphs. Finally, we explain the motivation for computing a fast depth-based complexity trace for a graph by linking the ideas of entropy measures and depth-based representations.

2.2.1 Deterministic and Probabilistic Complexity Measures

The quantification of the complexity of undirected graphs and networks has attracted significant attention due to its fundamental practical importance, and has proven powerful in a number of fields such as network analysis [44], chemistry, and pattern recognition [5]. Broadly speaking, there are two different approaches to measuring the complexity of graphs, namely a) deterministic complexity measures, and b) probabilistic complexity measures.

The available deterministic complexity measures include the encoding, substructure counting, and generative approaches. The encoding approach aims to measure the Kolmogorov complexity of the structure [45]. The substructure counting approach is based on the frequency of different substructures in a graph [46, 47, 48]. Finally, the generative approach aims to specify a basis set of elementary graph substructures and a set of operations which allow the substructures to be combined to form a larger graph. The complexity of the graph is then defined as the minimum number of operations required to form the graph from the basis structures [49].

The probabilistic complexity measures include the idea of measuring the entropy of a probability distribution associated with a graph. Existing approaches to computing probabilistic complexity are based on the notion of either randomness complexity or statistical complexity. Randomness complexity aims to quantify the degree of randomness or disorganization of a combinatorial structure. This approach aims to characterize an observed graph structure probabilistically and compute its associated Shannon entropy. Statistical complexity, on the other hand, aims to characterize a combinatorial structure using statistical features such as vertex degree statistics, edge density or the Laplacian spectrum. Viewed historically, most early work in this area falls into the randomness class, while recent work is statistically based. The main drawback of randomness complexity is that it does not properly capture the correlations between vertices [50]. Statistical complexity aims to overcome this problem by measuring regularities beyond randomness, and does not necessarily grow monotonically with randomness. In this thesis, we focus in more detail on using the entropy as a probabilistic complexity measure.

Unfortunately, the computation of the entropy of a graph is by no means a straightforward problem. In early work, Köner [51] developed a graph entropy which poses complexity characterization as an optimization problem. Assuming that there is a probability distribution associated with the vertices of the graph, the complexity is the minimal cross entropy between the probability distribution and the vertex packing polytype of the graph. However, this approach is not applicable to more general unweigthed graphs. To address this shortcoming, Dehmer [52, 53] has turned to information theory and proposed a novel and efficient means of computing graph entropies using information functionals which are derived from the topological structure of a graph and quantify the information content of the given graph structure. This approach avoids the combinatorial computations over different vertex partitions and achieves polynomial time complexity by constructing local information subgraphs for a given graph. Anand et al. [54] and Passerini et al. [55] have applied the von Neumann entropy (or quantum entropy) to the domain of graphs through a mapping between discrete Laplacians and quantum states [56]. If the discrete

Laplacian [57] is scaled by the inverse of the volume of the graph we obtain a density matrix whose entropy can be computed using the spectrum of the discrete Laplacian. The measure distinguishes between different structures. For instance it is maximal for random graphs, minimal for complete ones and takes on intermediate values for star graphs. In addition, when there is degree heterogeneity then the Shannon (classical) and von Neumann (quantum information theoretic) entropies are correlated. However, since the von Neumann entropy relies on the computation of the normalized Laplacian spectrum, its computational complexity is cubic in the number of vertices.

To render the computation more efficient, Han et al. [15] have shown how the computation can be rendered quadratic in the number of the vertices by its quadratic counterpart. An analysis of the quadratic entropy reveals that it can be computed from a number of permutation invariant matrix trace expressions. This leads to a simple expression for the approximate entropy in terms of elements of the adjacency matrix. Another straightforward route to compute the entropy of a graph is to use the probability state vectors of random walks on a graph [58]. For instance, in the case of the steady state of the discrete time random walk, the elements of the state vector are proportional to the normalized degrees of the vertices. From this probability distribution it is straightforward to compute the Shannon entropy.

Furthermore, to develop the approximate von Neumann entropy of Han et al. [15] one step further, Ye et al. [59] have explored how the von Neumann entropy for an undirected graph can be extended for a directed graph. To do this, they used Chung's [60] definition of the normalized Laplacian on a directed graph. According to this definition, the directed normalized Laplacian is Hermitian, so the interpretation of Passerini et al. in [55] still holds for the domain of directed graphs. The von Neumann entropy is essentially the Shannon entropy associated with the normalized Laplacian eigenvalues. The resulted von Neumann entropy expression of a directed graph depends on the in-degree and out-degree of pairs of vertices connected by edges.

2.2.2 Thermodynamic Depth-based Complexity Measures

An attractive alternative complexity measure for an undirected graph is to compute its thermodynamic depth complexity using depth-based representations [17]. The depth-based representation of undirected graph structures has been proven powerful tool for characterizing the topological structure in terms of the intrinsic complexity [17, 18]. One approach is to gauge the information content flow through a family of K layer subgraphs of an undirected graph (e.g., subgraphs around a vertex having a maximum topology distance or minimal path length K) of increasing size and to use the flow as a structural signature. By measuring the heat flow complexity of each subgraph, Escolano et al. [17] have shown how to use this approach to characterize each casual trajectory of an undirected graph leading a vertex to the graph by using the minimal enclosing Bregman balls (MEBBs) [61]. Then the depth complexity of such an undirected graph relies on the variability of the different trajectories from each vertex to the graph. Furthermore, this approach also allows a depth-based complexity trace to be defined which gauges how the complexities of the subgraphs vary as a function of depth [17]. However, to construct such a complexity trace of an undirected graph requires an expensive computation on measuring the intrinsic structural complexity. Moreover, straightforwardly constructing a complexity trace that characterizes an undirected graph structure in global manner is an elusive problem, since it is difficult to determine a fine root vertex in the graph.

To develop a depth-based complexity trace that addresses the mentioned problems, a condense family of expansion subgraphs around a dominant vertex and an efficient complexity measure for each of the (sub)graphs are required. To this end, we consider to define a centroid vertex as the dominant root vertex for a graph, and compute the entropies on a family of centroid expansion subgraphs derived from the vertex. As stated in Section 2.2.1, some of the entropies can be computed in a polynomial time. The resulted complexity trace of the graph linking the ideas of the depth-based representation and entropy measures can thus be efficiently computed. Like the graph embedding methods mentioned in Section 1.1, such a complexity trace offers us an elegant way to represent a graph into a feature vector. Moreover, the complexity trace can reflect a high dimensional complexity characteristics for a graph through the family of centroid expansion subgraphs. By contrast, the existing entropy based and the thermodynamic depth complexity measures only provide an one dimensional complexity feature for a graph.

2.3 Hypergraph Representations in Pattern Recognition

Graph based representations have been proven powerful in structure based pattern recognition. However, complex data in real world tends to exhibit multiple relationships that are hard to characterize by using graphs. To overcome this problem, hypergraph based strategies have recently been investigated for representing and processing structures where the relations present between objects are higher order. A hypergraph is a generalization of a graph [8]. Unlike the pairwise edges in a graph, hypergraph representations allow a hyperedge to encompass an arbitrary number of vertices, and can hence capture multiple relationships among features. One way to manipulate hypergraph structures is to exploit existing graph based methods for learning higher order models. Agarwal et al. [62] have performed hypergraph clustering by partitioning a weighted graph obtained by transforming the original hypergraph using a weighted sum of hyperedges to form edges. Zhou et al. [63] have presented a similar graph approximation method for hypergraphs by normalizing the Laplacian matrix of the star expansion of a hypergraph. On the side, the tensor (higher order matrix) based strategy has also been adopted for straightforwardly characterizing the higher order relations in a hypergraph rather than conducting a graph based pairwise approximation. Zass et al. [9] and Duchenne et al. [64] have separately applied high-degree affinity arrays (i.e., tensors) to formulate hypergraph matching problems using different cost functions. Both methods address the matching process in an algebraic manner but become intractable to compute if the hyperedges are not suitably sampled.

Shashua et al. [11, 65] have performed visual clustering using tensors to represent uniform hypergraphs (i.e. those for which the hyperedges have identical cardinality) extracted from images and videos. For detecting numbers of clusters in a tensor-based framework, their work has been complemented by He et al.'s [66] algorithm. Similar methods include those described in [67, 68, 69, 70, 71], in which tensors are used to represent the multiple relationships between objects. One limitation of the existing methods for hypergraph characterization is that they are usually restricted to uniform structures and cannot be applied to hypergraphs with arbitrary relational orders. To address this shortcoming, Ren et al. [12] have exploited a set of polynomial coefficients obtained from the hypergraph Ihara zeta function for characterizing nonuniform hypergraphs. Though effectively capturing the varying relational orders, the hypergraph Ihara coefficients tend to require an expensive computation even for hypergraphs of intermediate sizes.

Like graph based pattern recognition, an alternative approach for characterising hypergraphs is to use kernel methods. Unfortunately, most existing R-convolution kernels can not be performed on hypergraphs, due to the high order relationship among vertices. To overcome this problem, Wachman and Khardon [13] have generalized the existing graph kernels based on walks and then proposed a rooted kernel for hypergraphs based on random walks. Unfortunately, similar to the walk based graph kernels, one limitation of the rooted hypergraph kernel is the notorious tottering problem. This shortcoming limits the precise kernel measure between hypergraphs.

To address this problem, the substructure based strategy can be used. To this end, we consider to propose a new hypergraph kernel based on a new directed Weisfeiler-Lehman isomorphism algorithm for directed graphs. The new kernel for a pair of hypergraphs can be computed by performing the new isomorphism algorithm on the directed line graphs transformed from the hypergraphs. We show that the directed line graph of a (hyper)graph is a backtrackless structure, the new hypergraph kernel from the line graphs thus overcomes the shortcoming of tottering arising in the rooted hypergraph kernel.

2.4 Conclusion

We have reviewed the research literature on the domains of the entropy based complexity measures, depth-based complexity measures and kernel methods on structured data. We have analyzed the deficiencies of existing state-of-the-art methods and pointed out our possible solutions for overcoming these shortcomings. This chapter can be summarized as follows.

In Section 2.1, we review the basic concepts of kernel methods for vectorial data. Furthermore, we review some state-of-the-art information theoretic kernels for probability distributions. Finally, we review some state-of-the-art graph kernels, i.e., the R-convolution kernels. We analyze the strength and weakness of the existing graph kernels. We point out that defining an efficient and effective graph kernel still remains a challenge. To overcome the shortcomings of the existing graph kernels, we consider to define a new family of Jensen-Shannon diffusion kernel for (un)attributed graphs using the Jensen-Shannon divergence. Moreover, we also propose to use the depth-based representations as a means of defining a depth-based matching kernel for graphs.

In Section 2.2, we first review two kinds of graph complexity measures, namely a) deterministic complexity measures, and b) probabilistic complexity measures. Furthermore, we generally review the history of developing entropy based complexity measures for graphs. We point out that some of the entropies can be computed in polynomial time. Finally, we review an attractive alternative complexity measure for an undirected graph, namely the thermodynamic depth complexity using depth-based representations. This approach allows a depth-based complexity trace to be defined for a graph as a function of depth. Unfortunately, to construct such a complexity trace for an undirected graph requires an expensive computation on measuring the intrinsic structural complexity. Moreover, this approach also requires a fine root vertex in the graph. To address these problems, we propose to define a family of centroid expansion subgraphs rooted at a centroid vertex of a graph. As a result, a complexity trace of the graph can be computed by measuring the efficient entropies on the subgraphs.

In Section 2.3, we give the concept of a hypergraph. A hypergraph is a generalization of a graph. We review some state-of-the-art methods for hypergraphs. We analyze the shortcomings of the existing methods. To address these problems, we propose to transform a hypergraph into a directed line graph which can reflect full characteristics of the hypergraph. Then a new hypergraph kernel can be developed by performing a new developed directed Weisfeiler-Lehman isomorphism algorithm on the directed line graphs.

Chapter 3

Jensen-Shannon Diffusion Kernels for (Un)attributed Graphs

In this chapter, we present our first contribution to the design of a family of Jensen-Shannon diffusion kernels for (un)attributed graphs. For an unattributed graph, we commence by computing the von Neumann entropy that is developed by Han et al. [15]. Furthermore, we also develop a new Shannon entropy associated with a steady state random walk for the graph. For an attributed graph, we perform a tree-index method developed by Dahm et al. [16] for the purpose of strengthening the vertex labels. For a vertex, the tree-index method strengthens the vertex label by taking the union of the neighbouring vertex labels as lists. Unfortunately, this tree-index method tends to lead a rapid explosion of the strengthened label length. Moreover, strengthening a vertex label by only taking the union of the neighbouring label lists also tends to ignore the original label information of the vertex. To overcome these problems and improve the tree-index method, we propose to strengthen the label of a vertex as a new label list by taking the union of both the original vertex label and its neighbouring vertex labels. We also use the Hash function for the purpose of compressing the strengthened label list into a new short index label. As a result, we compute a new label Shannon entropy for the attributed graph in terms of the frequency of the strengthened labels.

With a pair of (un)attributed graphs and their entropies to hand, the diffusion kernel for the graphs can be computed using the Jensen-Shannon divergence between a composite entropy of the graphs and their individual entropies. We show that the Jensen-Shannon diffusion kernel for unattributed graphs overcomes the inefficiency arising in the R-convolution kernels. Moreover, we show that the Jensen-Shannon diffusion kernel for attributed graphs not only accommodates attributed graphs but also overcomes the shortcoming of discarding un-isomorphic substructures arising in the R-convolution kernels. We explore our Jensen-Shannon diffusion kernel on several graph datasets abstracted from bioinformatics databases. The experimental results demonstrate the effectiveness and efficiency of our new kernel.

Chapter outline

The remainder of this chapter is organized as follows. Section 3.1 gives the concepts of the von Neumann entropy and the Shannon entropy associated with a steady state random walk for an unattributed graph. Section 3.2 gives the concept of a tree-index based vertex label strengthening algorithm. Moreover, we analyze the shortcoming of the tree-index method and show how the tree-index method can be improved by addressing the shortcoming. Finally, a label Shannon entropy for an attributed graph is defined. Section 3.3 gives the definition of the Jensen-Shannon diffusion kernel for (un)attributed graphs. Section 3.4 provides the experimental evaluation. Finally, Section 3.5 concludes our work.

3.1 Entropy Measures for Unattributed Graphs

In this section we introduce two entropy measures for unattributed graphs required in this chapter. We commence by reviewing the concept of the von Neumann entropy proposed by Han et al. in [15]. Finally, we propose an alternative Shannon entropy using the probability distribution associated with a steady state random walk on a graph.

3.1.1 Von Neumann Entropy

We commence by reviewing the definition of the von Neumann entropy for a graph. The von Neumann entropy of a graph is the Shannon entropy associated with the eigenvalues of the normalized graph Laplacian [54]. We denote the graph under study by $G(V, E)$ where V is the set of vertices and $E \subseteq V \times V$ is the set of undirected edges. The symmetric adjacency matrix A for $G(V, E)$ is a $|V| \times |V|$ matrix that has elements

$$A(i,j) = \begin{cases} 1 & \text{if}(v_i, v_j) \in E; \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

The vertex degree matrix of $G(V, E)$ is a diagonal matrix D whose elements are given by $D(v_i, v_i) = d(v_i) = \sum_{v_j \in V} A(i,j)$. From the degree matrix and the adjacency matrix we can construct the Laplacian matrix $L = D - A$. The normalized Laplacian matrix is given by $\hat{L} = D^{-1/2} L D^{-1/2}$. The spectral decomposition of the normalized Laplacian matrix is $\hat{L} = \hat{\Phi}\hat{\Lambda}\hat{\Phi}^T$ where $\hat{\Lambda} = diag(\hat{\lambda}_1, \hat{\lambda}_2, ..., \hat{\lambda}_{|V|})$ is a diagonal matrix with the ordered eigenvalues as elements $(0 = \hat{\lambda}_1 < \hat{\lambda}_2 < ... < \hat{\lambda}_{|V|})$ and $\hat{\Phi} = (\hat{\phi}_1 | \hat{\phi}_2 | ... | \hat{\phi}_{|V|})$ is a matrix with the corresponding ordered orthonormal eigenvectors as columns. The normalized Laplacian matrix is positive semi-definite and so has all eigenvalues non-negative. The number of zero eigenvalues is the number of connected components in $G(V, E)$. The von Neumann entropy of $G(V, E)$ associated with the normalized Laplacian eigenspectrum [54] is defined as

$$H_{VN} = - \sum_{i=1}^{|V|} \frac{\hat{\lambda}_i}{|V|} \log \frac{\hat{\lambda}_i}{|V|}. \tag{3.2}$$

The computation of the von Neumann entropy requires a number of operations that is cubic in the number of vertices, since it requires computing the eigenvalues. Han et al. [15] have shown how the computation can be reduced to quadratic time by a) approximating the Shannon entropy by its quadratic counterpart, and b) evaluating traces of $\hat{L}$ using degree distribution. To commence, we follow the definition of Han et al. in [15] and approximate the Shannon entropy $\frac{\hat{\lambda}_i}{|V|}(1 - \frac{\hat{\lambda}_i}{|V|})$ as

$$\begin{aligned} H_{VN} &= -\sum_{i=1}^{|V|} \frac{\hat{\lambda}_i}{|V|} \log \frac{\hat{\lambda}_i}{|V|} \simeq \sum_{i=1}^{|V|} \frac{\hat{\lambda}_i}{|V|}(1 - \frac{\hat{\lambda}_i}{|V|}) \\ &= \frac{\sum_{i=1}^{|V|} \hat{\lambda}_i}{|V|} - \frac{\sum_{i=1}^{|V|} \hat{\lambda}_i^2}{|V|^2}. \end{aligned} \tag{3.3}$$

Using the fact that $Tr[\hat{L}^n] = \sum_{i=1}^{|V|} \hat{\lambda}_i^n$, the quadratic entropy can be rewritten as

$$H_{VN} = \frac{Tr[\hat{L}]}{|V|} - \frac{Tr[\hat{L}^2]}{|V|^2}. \tag{3.4}$$

Since the normalized Laplacian matrix $\hat{L}$ is symmetric and has unit diagonal elements, then we have

$$Tr[\hat{L}] = |V|. \tag{3.5}$$

Similarly, for the trace of the squared normalized Laplacian, we have

$$\begin{aligned} Tr[\hat{L}^2] &= \sum_{v_i \in V} \sum_{v_j \in V} \hat{L}_{ij} \hat{L}_{ij} = \sum_{v_i \in V} \sum_{v_j \in V} (\hat{L}_{ij})^2 \\ &= \sum_{\substack{v_i, v_j \in V \\ i=j}} (\hat{L}_{ij})^2 + \sum_{\substack{v_i, v_j \in V \\ v_i \neq v_j}} (\hat{L}_{ij})^2 \\ &= |V| + \sum_{(v_i, v_j) \in E} \frac{1}{d(v_i) d(v_j)}. \end{aligned} \tag{3.6}$$

Substituting Eq.(3.4) and Eq.(3.5) into Eq.(3.3), the entropy becomes

$$H_{VN}(G) \simeq 1 - \frac{1}{|V|} - \sum_{(v_i, v_j) \in E} \frac{1}{|V|^2 d(v_i) d(v_j)}. \tag{3.7}$$

For the graph $G(V, E)$ with $|V|$ vertices, the approximated von Neumann entropy $H_{VN}(G)$ requires time complexity $O(|V|^2)$. This is because the degree matrix D of $G(V, E)$ can be computed by just visiting the entries in the adjacency matrix A, then the required entropy $H_{VN}(G)$ of $G(V, E)$ can be directly computed by visiting all the $|V|^2$

pairs of vertices once in the adjacency matrix A. By contrast, the original von Neumann entropy defined in [54] requires time complexity $|V|^3$. This indicates that the approximated von Neumann entropy can be computed in polynomial time, compared to the original von Neumann entropy.

3.1.2 Shannon Entropy associated with Steady State Random Walks

Finally, we use the probability distribution associated with the steady state random walk on a graph to calculate its Shannon entropy. For a vertex $v_i \in V$, the probability of a steady state random walk on $G(V, E)$ visiting vertex v_i is

$$P_G(v_i) = d(v_i) / \sum_{v_j \in V} d(v_j). \tag{3.8}$$

From this probability distribution it is straightforward to compute the Shannon entropy as

$$H_S(G) = -\sum_{i=1}^{|V|} P_G(v_i) \log P_G(v_i). \tag{3.9}$$

For the graph $G(V, E)$ with $|V|$ vertices, the Shannon entropy $H_S(G)$ associated with the steady state random walk requires time complexity $O(|V|^2)$. Because this entropy also relies on the computation of the degree matrix D of $G(V, E)$, which can be computed by visiting all the $|V|^2$ pairs of vertices once in the adjacency matrix A.

3.2 A Label Entropy for An Attributed Graph

In this section, we describe how to compute a label entropy for an attributed graph. We commence by reviewing the definition of a tree-index vertex label strengthening method developed by Dahm et al. in [16]. Finally, we show how to compute a label Shannon entropy for attributed graphs associated with the probability distribution over the strengthened labels.

3.2.1 A Tree-Index Based Vertex Label Strengthening Method

We use the tree-index (TI) method for strengthening the vertex labels. Given an attributed graph $G(V,E)$, the label of a vertex $v_i \in V$ $(i = 1,\ldots,|V|)$ is $f(v_i)$. Using the TI method, the new strengthened label for v_i at the iteration h is defined as

$$TI_h(v_i) = \begin{cases} f(v_i) & \text{if } h = 0, \\ \cup_{v_j}\{TI_{h-1}(v_j)\} & \text{otherwise.} \end{cases} \tag{3.10}$$

where v_j $(j = 1,\ldots,|V|)$ is a vertex adjacent to v_i. At each iteration h, the TI method takes the union of neighbouring vertex label lists as a new label list for the vertex v_i from the previous iteration (the initial step is identical to listing). This iteratively creates a deeper list corresponding to a subtree, rooted at v_i and branching for h layers. An example of how the TI method defined in Eq.(3.10) strengthens the vertex label is shown in Fig.3.1. In this example, the initialized vertex labels for vertices A to E are their corresponding vertex degrees, i.e., 1, 2, 3, 2 and 2 respectively. Using the TI method, the second iteration indicates the strengthened labels for vertices A to E as $\{\{1,3\}\}$, $\{\{2\},\{2,2,2\}\}$, $\{\{1,3\},\{2,3\},\{2,3\}\}$,$\{\{2,2,2\},\{2,3\}\}$, and $\{\{2,2,2\},\{2,3\}\}$ respectively.

Unfortunately, as Fig.3.1 indicates, the above procedure clearly leads to a rapid explosion of the labels length. Moreover, strengthening a vertex label by only taking the union of the neighbouring label lists also ignores the original label information of the vertex. To overcome these problems, at each iteration h we propose to strengthen the label of a vertex as a new label list by taking the union of both the original vertex label and its neighbouring vertex labels. We use a Hash function to compress the strengthened label list into a new short label. The pseudocode of the re-defined TI algorithm is shown in Algorithm 2, where the neighbourhood of a vertex $v \in V$ is denoted as $\mathcal{N}(v) = \{u|(v,u) \in E\}$.

In step 4 of Algorithm 1, we propose to use the Hash function for the objective of compressing the strengthened label. A Hash function is a function that can map the digital data of arbitrary size into the digital data of required fixed size [72]. The reasons of using the Hash function are twofold. First, the Hash function can easily compress each

Algorithm 1: Vertex labels strengthening procedure

```
1: Initialization.
     • Input an attributed graph $G(V, E)$.
     • Set h=0. For a vertex $v \in V$, assign the original label $f(v)$ as the initial label $\mathcal{L}_h(v)$.
2: Update the label for each vertex.
     • Set h=h+1. For each vertex $v \in G$, assign a new strengthened label as
           $\mathcal{L}_h(v) = \cup_{u \in \mathcal{N}(v)} \{\mathcal{L}_{h-1}(u)\}.$
     • Arrange the sequence of the elements in $\mathcal{L}_h(v)$ as ascending order and concatenate the
       elements into a tuple as $s_h(v)$. Add $\mathcal{L}_{h-1}(v)$ as a suffix of $s_h(v)$.
     • Re-write the new strengthened label for $v$ as
           $\mathcal{L}_h(v) = s_h(v).$                                   (3.11)
3: Compress the vertex label into a new short label.
     • Using a label compressing function $f : \mathcal{L} \to \Sigma$, compress the label $\mathcal{L}_h(v)$ into a new short
       label for each vertex $v$ as
           $\mathcal{L}_h(v) = f(\mathcal{L}_h(v)).$                      (3.12)
4: Check $h$.
     • Check h. Repeat steps 2, 3 and 4 until the iteration $h$ achieves an expected value.
```

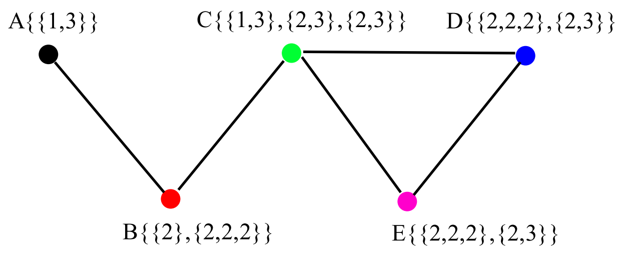


Figure 3.1: Example of TI algorithm.

strengthened label described by a long list into a new short integer. This will provides us a beneficial way of saving the physical memory, for the matlab computation. Second, the compressed label from the Hash function can be efficiently visited [72] using the index assigned by the function. As a result, we can efficiently strengthen and process a vertex label using the TI method. Note that, in step 4, we need to use the same Hash function. This guarantees that all the identical labels of different graphs are mapped into the same index. Moreover, for a graph $G(V,E)$ and its pairwise vertices v_i and v_j, if the labels $\mathcal{L}_h(v_i) = \mathcal{L}_h(v_j)$ the subtrees corresponded by the labels are isomorphic.

3.2.2 A Label Shannon Entropy for Attributed Graphs

In this subsection, we define a Shannon entropy over the label probability distribution for an attributed graph. The entropy measures the uncertainty of the labels for the graph (i.e., the ambiguity of the subtrees corresponded by the particular labels). Assume $\mathcal{L} = \{l_1, \ldots, l_x, \ldots, l_{|\mathcal{L}|}\}$ is a label set that contains all the vertex labels (including the original and strengthened labels) for different graphs. Given an attributed graph $G(V,E)$ and its compressed strengthening label $\mathcal{L}_h(v)$ defined in Eq.(6.8) for any vertex $v \in V$ at iteration h, we compute the frequency of a particular label l_x contained in $G(V,E)$, i.e., $c_G^h(l_x)$ for iteration h. The probability $p_G^h(l_i)$ of a label l_x for $G(V,E)$ at iteration h is

$$p_G^h(l_x) = \frac{c_G^h(l_x)}{\sum_{x=1}^{|\mathcal{L}|} c_G^h(l_x)}. \tag{3.13}$$

From the probability distribution P_G^h of $G(V,E)$, i.e., $p_G^h(l_1),\ldots,p_G^h(l_i),\ldots,p_G^h(l_{|\mathcal{L}|})$, the label Shannon entropy H_S^L for $G(V,E)$ at iteration h is defined as

$$H_S^{\mathcal{L}}(G) = H_S^{\mathcal{L}}(P_G^h) = -\sum_{i=x}^{|\mathcal{L}|} p_G^h(l_x)\log p_G^h(l_x). \tag{3.14}$$

3.3 Jensen-Shannon Diffusion Kernels for Graphs

In this section, we define the new kernel for (un)attributed graphs using the Jensen-Shannon divergence. We commence by presenting the definition of the Jensen-Shannon divergence. Moreover, we review the concept of a state-of-the-art graph kernel using the divergence and analyze its drawbacks. Finally, we define the Jensen-Shannon diffusion kernel for (un)attributed graphs by using the divergence measure.

3.3.1 The Jensen-Shannon Divergence

The Jensen-Shannon divergence is a mutual information dissimilarity measure between probability distributions in terms of the entropy difference associated with the probability distributions. Assume $M_+^1(\chi)$ is a set of probability distributions where χ is a set provided with some $\sigma - algebra$ of measurable subsets, the Jensen-Shannon divergence D_{JS} : $M_+^1(\chi) \times M_+^1(\chi) \to \mathbb{R}^+$ between the probability distributions P and Q is a negative definite (**nd**) function [14, 54] as:

$$\begin{aligned} D_{JS}(P,Q) &= \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M) \\ &= \frac{1}{2}\int_{\chi}\ln(\frac{dP}{dM})dP + \frac{1}{2}\int_{\chi}\ln(\frac{dQ}{dM})dQ, \end{aligned} \tag{3.15}$$

where $M = \frac{P+Q}{2}$ and $D_{KL}(P||M) = \int_{\chi}\ln(\frac{dP}{dM})dP$ is the Kullback-Leibler divergence between P and M. If χ is countable, i.e., $P = (p_1, p_m, \ldots, p_M)$ and $Q = (q_1, q_m, \ldots, q_M)$ are two discrete probability distributions, a general definition is

$$D_{JS}(P,Q) = H_S(\frac{P+Q}{2}) - \frac{H_S(P) + H_S(Q)}{2}. \tag{3.16}$$

where $H_S(P) = -\sum_{m=1}^{M} p_m \log p_m$ is the Shannon entropy of the probability distribution P, $\frac{P+Q}{2}$ is the composite probability distribution for P and Q, and $H_S(\frac{P+Q}{2}) = -\sum_{m=1}^{M} \frac{p_m+q_m}{2} \log \frac{p_m+q_m}{2}$ is the composite Shannon entropy from $\frac{P+Q}{2}$ for P and Q.

We are interested in computing a graph kernel for a pair of graphs using the Jensen-Shannon divergence. In the literature, some kernels for structured data from the Jensen-Shannon divergence have been developed. One instance is the Jensen-Shannon kernel [30, 54]. Assume the probability distributions P and Q are computed over two structured data, the Jensen-Shannon kernel between the two data is defined as

$$k_{JSK}(P,Q) = \log 2 - D_{JS}(P,Q), \tag{3.17}$$

where k_{JSK} is a positive definite kernel [30].

In [73], we have developed the Jensen-Shannon kernel k_{JSK} one step further and thus defined a new kernel for graphs using the Jensen-Shannon divergence. Assume a pair of graphs $G(V,E)$ and $G'(V',E')$. The Jensen-Shannon graph kernel for the graphs is defined as

$$\begin{aligned} k_{JSGK}(G,G') &= \log 2 - D_{JS}(G,G') \\ &= \log 2 - H_E(G_\times) + \frac{H_E(G) + H_E(G')}{2}. \end{aligned} \tag{3.18}$$

Here, $H_E(\cdot)$ can be either the von Neumann entropy $H_{VN}(\cdot)$ defined in Eq.(3.7) or the Shannon entropy $H_S(\cdot)$ defined in Eq.(3.9). $G_\times$ is a product graph for G and G' (i.e., a composite graph for G and G'), and has the vertex set $V_\times$ and edge set $E_\times$ as

$$\begin{cases} V_\times = \{(v,v') \in V \times V' \ : \ v \in V \wedge \ v' \in V'\}; \\ E_\times = \{((u,u'),(v,v')) \in V_\times^2 \ : \ (u,v) \in E \wedge (u',v') \in E'\}. \end{cases} \tag{3.19}$$

Unlike the R-convolution kernels, the entropy associated with a probability distribution of an individual graph can be computed without decomposing the graph or enumerating its substructures. As a result, the computation of the Jensen-Shannon kernel between a pair of graphs avoids burdensome (dis)similarity measurements which are computed by comparing all substructure pairs.

Unfortunately, there are four shortcomings arising in the Jensen-Shannon graph kernel. First, this kernel can only capture the global similarity between a pair of graphs, and hence lacks information concerning the interior topology of the graphs. Second, the required composite entropy is computed from a product graph formed by the pair of graphs, the kernel thus does not reflect the detailed correspondence information. Third, Eq.(3.19) indicates that the size of the product graph $G_\times$ is multiple in the sizes of G and G' (i.e., $|V_\times| = |V| \times |V'|$). As a result, the kernel value may be dominated by $G_\times$ of large size. Furthermore, computing the Jensen-Shannon kernel for graphs from the product graph is less intuitive, thus making it hard to prove whether the kernel k_{JSGK} is positive definite. Fourth, the kernel k_{JSGK} is restricted on unattributed graphs. To overcome these problems, we propose new graph kernels by using the Jensen-Shannon divergence measure.

3.3.2 The Unattributed Jensen-Shannon Diffusion Kernel

To compute the Jensen-Shannon diffusion kernel for a pair of unattributed graphs using the Jensen-Shannon divergence, we require a composite entropy for the graphs. Unfortunately, directly computing the composite entropy tends to be elusive, since the number of discrete probabilities for the graphs may be different. One way to overcome this problem is to compute the composite entropy from a composite structure formed by the pair of graphs. To address the shortcoming of dominating kernel value by the large product graph that arises in the Jensen-Shannon graph kernel k_{JSGK}, we propose to compute the composite entropy for a pair of graphs from their disjoint union graph (i.e., a composite graph), instead of the product graph. For a pair of graphs $G(V, E)$ and $G'(V', E')$, the disjoint union graph $G_U(V_U, E_U)$ of G and G' is defined as [74]

$$G_U = G \cup G' = \{V \cup V', E \cup E'\}. \tag{3.20}$$

Based on the definition in [51], the entropy of G_U is

$$H_E(G_U) = \pi H_E(G) + \pi' H_E(G'), \tag{3.21}$$

where $\pi = |V|/(|V| + |V'|)$, $\pi' = |V'|/(|V| + |V'|)$, and $\pi = 1 - \pi'$. Here, $H_E(\cdot)$ can be either the von Neumann entropy $H_{VN}(\cdot)$ defined in Eq.(3.7) or the Shannon entropy $H_S(\cdot)$ defined in Eq.(3.9). Through Eq.(3.21), we observe that the composite entropy for a pair of graphs from their disjoint union graph can be directly computed based on their individual entropies. As a result, the composite entropy can be efficiently computed.

For a pair of unattributed graphs $G(V, E)$ and $G'(V', E')$, we commence by computing their disjoint union graph G_U and their individual entropies. The Jensen-Shannon divergence between the unattributed graphs $G(V, E)$ and $G'(V', E')$ is defined as

$$D_{JS}(G, G') = H_E(G_U) - \frac{H_E(G) + H_E(G')}{2}, \tag{3.22}$$

where $H_E(G_U)$ is the composite entropy of G and G' defined in Eq.(3.21), and $H_E(\cdot)$ can be either the von Neumann entropy $H_{VN}(\cdot)$ or the Shannon entropy $H_S(\cdot)$ associated with the steady state random walk.

With the Jesnen-Shannon divergence for unattributed graphs defined in Eq.(3.22) to hand, we define a Jensen-Shannon diffusion kernel k_{JS}: $G \times G' \to R^+$ as

$$\begin{aligned} k_{JS}(G, G') &= \exp\{-\lambda D_{JS}(G, G')\} \\ &= \exp\{\lambda \frac{H_E(G) + H_E(G')}{2} - \lambda H_E(G_U)\}. \end{aligned} \tag{3.23}$$

where λ is a decay factor and satisfies $0 < \lambda \leq 1$. For simplification, we set $\lambda = 1$. Note that, the computation of the diffusion kernel k_{JS} only depends on the individual entropies $H_E(G)$ and $H_E(G')$, since the required composite entropy $H_E(G_U)$ can be directly computed from $H_E(G)$ and $H_E(G')$ using Eq.(3.21).

Lemma 3.1 *The Jensen-Shannon diffusion kernel k_{JS} is positive definite (**pd**).* □

Proof. This follows the definition in [75, 76], if a dissimilarity measure $s_G(G, G')$ between a pair of graphs $G(V, E)$ and $G'(V', E')$ satisfies symmetry, then a diffusion kernel

$k_s = \exp(-\lambda s_G(G, G'))$ associated with the dissimilarity measure $s_G(G, G')$ is **pd**. For the graphs G and G', the Jensen-Shannon divergence D_{JS} is a dissimilarity measure and satisfies symmetry. Thus, the kernel k_{JS} is a **pd** kernel by exponentiating the Jensen-Shannon divergence D_{JS}. ■

Time Complexity: For N graphs each of which has n vertices, computing the $N \times N$ kernel matrix using the Jensen-Shannon diffusion kernel k_{JS} requires time complexity $O(Nn^2 + N^2)$. This is because computing the von Neumann entropy or the random walk Shannon entropy only requires time complexity $O(n^2)$. Computing the entropies for all the N graphs thus requires time complexity $O(Nn^2)$. Computing the $N \times N$ kernel matrix requires time complexity $O(N^2)$, because the composite entropy for each pair of graphs can be directly computed using Eq.(3.21). As a result, the complete time complexity is $O(Nn^2 + N^2)$. Moreover, for a pair of graphs (i.e., $N = 2$), the time complexity is $O(n^2)$.

The computational complexity of our Jensen-Shannon diffusion kernel k_{JS} for unattributed graphs is only quadratic in the number of graph vertices. As a result, unlike most existing R-convolution kernels the unattributed diffusion kernel k_{JS} can be efficiently computed. Furthermore, since the composite entropy for a pair of graph is computed from their disjoint union graph, the size of which is the sum of the graph sizes. The unattributed diffusion kernel k_{JS} also overcomes one of the shortcomings arising in the Jensen-Shannon graph kernel k_{JSGK} [73], i.e., the Jensen-Shannon graph kernel value is dominated by the product graph of large size. Finally, Reisen and Bunke [25] have observed that in a diffusion kernel the exponentiation enhances the (dis)similarity measure between the graphs. As a result, the unattributed diffusion kernel k_{JS} is not only a positive definite kernel but also enhances the similarity measure for graphs by exponentiating the Jensen-Shannon divergence measure.

Unfortunately, like the Jensen-Shannon graph kernel k_{JSGK} the new unattributed diffusion kernel k_{JS} has three other drawbacks. First, from Eq.(3.21), Eq.(3.22) and Eq.(3.23), we observe that the probability distribution from a disjoint union graph G_U

of G and G' cannot reflect any correspondence information between the discrete probabilities for the graphs G and G'. Second, the required entropy, either the von Neumann entropy or the random walk Shannon entropy, of a graph for the kernel k_{JS} relates to the vertex degree. The vertex degree is structurally simple and reflects limited topology information, the kernel k_{JS} thus reflects limited interior topology information for graphs. Third, the kernel k_{JS} is only restricted to unattributed graphs and cannot capture any label information residing on the graph vertices. To overcome the shortcomings, below we will develop another new Jensen-Shannon diffusion kernel for attributed graphs.

3.3.3 The Attributed Jensen-Shannon Diffusion Kernel

We compute a Jensen-Shannon diffusion kernel for attributed graphs by measuring the Jensen-Shannon divergence between their label Shannon entropies. For a pair of graphs $G(V, E)$ and $G'(V', E')$, we commence by strengthening the vertex labels using Algorithm 2 for each iteration. For iteration h, the probability distributions for the strengthened vertex labels are $P_G^h = \{p_G^h(l_1), \ldots, p_G^h(l_x), \ldots, p_G^h(l_{|\mathcal{L}|})\}$ and

$$P_{G'}^h = \{p_{G'}^h(l_1), \ldots, p_{G'}^h(l_x), \ldots, p_{G'}^h(l_{|\mathcal{L}|})\}$$

respectively. The Jensen-Shannon divergence between the attributed graphs $G(V, E)$ and $G'(V', E')$ at iteration h is defined as

$$\begin{aligned} D_{JS}^h(G, G') &= D_{JS}^h(P_G^h, P_{G'}^h) \\ &= H_S^{\mathcal{L}}(\frac{P_G^h + P_{G'}^h}{2}) - \frac{H_S^{\mathcal{L}}(P_G^h) + H_S^{\mathcal{L}}(P_{G'}^h)}{2} \\ &= -\sum_{x=1}^{|\mathcal{L}|} \frac{p_G^h(l_x) + p_{G'}^h(l_x)}{2} \log \frac{p_G^h(l_x) + p_{G'}^h(l_x)}{2} \\ &+ \sum_{x=1}^{|\mathcal{L}|} p_G^h(l_x) \log p_G^h(l_x) + \sum_{x=1}^{|\mathcal{L}|} p_{G'}^h(l_x) \log p_{G'}^h(l_x). \end{aligned} \quad (3.24)$$

where H_L^S is the label Shannon entropy defined in Eq.(3.14), and $\frac{P_G^h+P_{G'}^h}{2}$ is the composite probability distribution of P_G^h and $P_{G'}^h$.

With the Jesnen-Shannon divergence for attributed graphs defined in Eq.(3.24) to hand, we define a Jensen-Shannon diffusion kernel k_{JS}^H: $G \times G' \to R^+$ as

$$\begin{aligned} k_{JS}^H(G,G') &= \sum_{h=0}^{H} \exp\{-\lambda D_{JS}^h(G,G')\} \\ &= \sum_{h=0}^{H} \exp\{\lambda \sum_{x=1}^{|\mathcal{L}|} \frac{p_G^h(l_x)+p_{G'}^h(l_x)}{2} \log \frac{p_G^h(l_x)+p_{G'}^h(l_x)}{2} \\ &\quad - \frac{1}{2}\lambda \sum_{x=1}^{|\mathcal{L}|} p_G^h(l_x) \log p_G^h(l_x) - \frac{1}{2}\lambda \sum_{x=1}^{|\mathcal{L}|} p_{G'}^h(l_x) \log p_{G'}^h(l_x)\}. \end{aligned} \tag{3.25}$$

where H is the largest number of iteration h, P_G^h is the label probability distribution of G from the TI method as the iteration h and is defined in Eq.(3.13), and λ is a decay factor and satisfies $0 < \lambda \leq 1$. For simplification, we set $\lambda = 1$.

Lemma 3.2 *The Jensen-Shannon diffusion kernel k_{JS}^H is positive definite (**pd**).* □

Proof. This follows the definition in [75, 76], if a dissimilarity measure $s_G(G,G')$ between a pair of graphs $G(V,E)$ and $G'(V',E')$ satisfies symmetry, then a diffusion kernel $k_s = \exp(-\lambda s_G(G,G'))$ associated with the dissimilarity measure $s_G(G,G')$ is **pd**. For the graphs G and G', the Jensen-Shannon divergence measure $D_{JS}^h(G,G') = D_{JS}^h(P_G^h, P_{G'}^h)$ associated with their label probability distributions P_G^h and $P_{G'}^h$ is a dissimilarity measure and satisfies symmetry. We thus define a **pd** kernel, i.e., the Jensen-Shannon base diffusion kernel for G and G' at each iteration h (for the TI method), and is defined as

$$\begin{aligned} k_{JSB}^h(G,G') &= \exp\{-\lambda D_{JS}^h(G,G')\} \\ &= \exp\{\lambda \sum_{x=1}^{|\mathcal{L}|} \frac{p_G^h(l_x)+p_{G'}^h(l_x)}{2} \log \frac{p_G^h(l_x)+p_{G'}^h(l_x)}{2} \\ &\quad - \frac{1}{2}\lambda \sum_{x=1}^{|\mathcal{L}|} p_G^h(l_x) \log p_G^h(l_x) - \frac{1}{2}\lambda \sum_{x=1}^{|\mathcal{L}|} p_{G'}^h(l_x) \log p_{G'}^h(l_x)\}. \end{aligned} \tag{3.26}$$

Thus, the Jensen-Shannon diffusion kernel k_{JS}^H can be re-written as

$$\begin{aligned} k_{JS}^H(G,G') &= k_{JSB}^1(G,G') + \ldots + k_{JSB}^H(G,G') \\ &= \sum_{h=0}^{H} k_{JSB}^h(G,G'), \end{aligned} \tag{3.27}$$

i.e., the Jensen-Shannon diffusion kernel k_{JS}^H is the sum of several Jensen-Shannon base diffusion kernels. The function which is the sum of **pd** kernels is also a **pd** kernel. As a result, the Jensen-Shannon diffusion kernel k_{JS}^H is **pd**. ■

Time Complexity: For N graphs each of which has n vertices and label set $\mathcal{L}$, computing the $N \times N$ kernel matrix using the Jensen-Shannon diffusion kernel k_{JS}^H requires time complexity $O(HN^2n^2+HN^3n)$. This is because computing the compressed strengthened labels for a graph at each iteration h $(0 \leq h \leq H)$ needs to visit all the n^2 entries of the adjacency matrix, and thus requires time complexity $O(Hn^2)$ for all the H iterations. Computing the probability distribution for a graph requires time complexity $O(HNn^2)$ (for the worst case, i.e. each vertex label for the N graphs at all the H iterations are all different and there thus are NHn different labels in L), because it needs to visit all the HNn entries in L for the n vertices. Computing the $N \times N$ kernel matrix requires time complexity $O(HN^3n)$, because the label Shannon entropy for each pair of graphs requires time complexity $O(HNn)$. As a result, the complete time complexity is $O(HN^2n^2 + HN^3n)$. Moreover, for a pair of graphs (i.e., $N = 2$), the time complexity is thus $O(Hn^2)$.

Compared to the Jensen-Shannon diffusion kernel k_{JS} for unattributed graphs, the Jensen-Shannon diffusion kernel k_{JS}^H for attributed graphs has three advantages. First, from Eq.(3.24) we observe that there is correspondence between the discrete probabilities identified by a strengthened label l_x. As a result, our attributed diffusion kernel k_{JS}^H overcomes the shortcoming of lacking correspondence information between probabilities that arises in the unattributed diffusion kernel k_{JS}. Second, for the attributed diffusion kernel k_{JS}^H, the identical strengthened labels from the TI method correspond to the same class of isomorphic subtrees, the correspondence between the probability distribution also reflects the correspondence information between pairs of isomorphic subtrees. By contrast, for the unattributed diffusion kernel k_{JS} the required entropy computed from the vertex degree can only reflect limited topology information, because the vertex degree is structurally simple. As a result, the attributed diffusion kernel k_{JS}^{H} overcomes the shortcoming of reflecting limited interior structural information that arises in the unattributed diffusion kernel k_{JS}. Third, the attributed diffusion kernel k_{JS}^{H} through the TI method overcomes the restriction to unattributed graphs that arises in the unattributed diffusion kernel k_{JS}.

Furthermore, from Eq.(3.14) we also observe that the label Shannon entropy required for the attributed diffusion kernel k_{JS}^{H} represents the ambiguity of all the compressed strengthening labels at an iteration h. Each label corresponds to a subtree rooted at the vertex containing the label, and all the subtrees are considered in the computation of the kernel measure. As a result, the attributed diffusion kernel k_{JS}^{H} also overcomes the shortcoming of discarding un-isomorphic substructures arising in the R-convolution kernels. These observations indicate that our Jensen-Shannon diffusion kernel k_{JS}^{H} for attributed graphs has better ability of characterizing graphs than most existing graph kernels.

Finally, like the unattributed diffusion kernel k_{JS}, the attributed diffusion kernel k_{JS}^{H} also enhances the similarity measure for graphs by exponentiating the Jensen-Shannon divergence measure.

3.4 Experimental Results

In this section, we empirically evaluate the performance of the new Jensen-Shannon diffusion kernels for (un)attributed graphs. Our experimental evaluation consists of two parts. First, we test our new kernel on classification problems using standard graph datasets. These datasets are abstracted from bioinformatics. Furthermore, we also compare our new kernels to several state-of-the-art graph kernels. Second, we evaluate the computational efficiency of the new kernels.

3.4.1 Graph Datasets

We demonstrate the performance of our new kernel on six standard graph datasets from bioinformatics databases. These datasets include: MUTAG, NCI1, NCI109, ENZYMES, PPIs and PTC(MR) [17, 77, 78, 79, 80]. More details are shown in Table.4.1.

MUTAG: The MUTAG dataset consists of graphs representing 188 chemical compounds, and aims to predict whether each compound possesses mutagenicity.

NCI1 and NCI109: The NCI1 and NCI109 datasets consist of graphs representing two balanced subsets of datasets of chemical compounds screened for activity against non-small cell lung cancer and ovarian cancer cell lines respectively. There are 4110 and 4127 graphs in NCI1 and NCI109 respectively.

ENZYMES: The ENZYMES dataset consists of graphs representing protein tertiary structures consisting of 600 enzymes from the BRENDA enzyme database. The task is to correctly assign each enzyme to one of the 6 EC top-level classes.

PPIs: The PPIs dataset consists of protein-protein interaction networks (PPIs). The graphs describe the interaction relationships between histidine kinase in different species of bacteria. Histidine kinase is a key protein in the development of signal transduction. If two proteins have direct (physical) or indirect (functional) association, they are connected by an edge. There are 219 PPIs in this dataset and they are collected from 6 different kinds of bacteria.

PTC: The PTC (The Predictive Toxicology Challenge) dataset records the carcinogenicity of several hundred chemical compounds for Male Rats (MR), Female Rats (FR), Male Mice (MM) and Female Mice (FM). These graphs are very small (i.e., $20-30$ vertices, and $25-40$ edges) and sparse. We select the graphs of MR for evaluation. There are 344 graphs in MR.

Table 3.1: Information of the graph-based datasets

Datasets	MUTAG	NCI1	NCI109	ENZYMES	PPIs	PTC(MR)
Max # vertices	28	111	111	126	238	109
Min # vertices	10	3	4	2	3	2
Mean # vertices	17.93	29.87	29.68	32.63	109.63	25.60
# graphs	188	4110	4127	600	219	344
# classes	2	2	2	6	6	2

3.4.2 Experiments on Graph Classification

We evaluate the performance of our unattributed Jensen-Shannon diffusion kernel k_{JS} (JSDKU) with either the random walk Shannon entropy or the approximated von Neumann entropy. We also evaluate the performance of our attributed Jensen-Shannon diffusion kernel k_{JS}^{H} (JSDKA). Moreover, we compare our diffusion kernels with several alternative state-of-the-art graph kernels. The graph kernels used for comparison include: 1) the quantum Jensen-Shannon kernel (QJSK) [110], 2) the Weisfeiler-Lehman subtree kernel (WLSK) [7], 3) the shortest path graph kernel (SPGK) [39], 4) the graphlet count graph kernel with graphlet of size 3 (GCGK) [82], 5) the backtraceless random walk k-ernel using the Ihara zeta function based cycles (BRWK) [21], and 6) the random walk graph kernel (RWGK) [22]. For our JSDKA kernel, we set the largest number of iteration H as 10. The reason for this is that the kernel value tends to be stable when $h > 9$. For the WLSK kernel, we set the highest dimension (i.e., the highest height of subtrees) of the Weisfeiler-Lehman isomorphism as 10.

For each kernel, we compute the kernel matrix on each graph dataset. We perform 10-fold cross-validation using the C-Support Vector Machine (C-SVM) to compute the classification accuracies, from LIBSVM [83]. We use nine folds for training and one fold for testing. Each of the C-SVMs were performed along with their parameters optimized on each dataset. We repeat the experiment 10 times. We report the average classification

Table 3.2: Classification accuracy (in % ± standard error) comparisons

Datasets	MUTAG	NCI1	NCI109	ENZYMES	PPIs	PTC(MR)
JSDKA	**85.33** ± .65	**85.87** ± .14	**85.63** ± .13	**56.93** ± .41	**89.87** ± .43	**59.88** ± .40
JSDKU(Shannon entropy)	83.11 ± .80	62.50 ± .33	63.00 ± .35	20.81 ± .29	34.57 ± .54	57.29 ± .41
JSDKU(von Neumann Entropy)	83.50 ± .79	62.20 ± .35	62.00 ± .37	22.92 ± .27	32.37 ± .50	56.30 ± .40
QJSK	82.72 ± .44	69.09 ± .20	70.17 ± .23	36.58 ± .46	65.61 ± .77	56.70 ± .49
WLSK	82.88 ± .57	84.77 ± .13	84.49 ± .13	52.75 ± .44	88.09 ± .41	58.26 ± .47
SPGK	83.38 ± .81	74.21 ± .30	73.89 ± .28	29.00 ± .48	59.04 ± .44	55.52 ± .46
GCGK3	82.04 ± .39	63.72 ± .12	62.33 ± .13	24.87 ± .22	46.61 ± .47	55.41 ± .59
BRWK	77.50 ± .75	60.34 ± .17	59.89 ± .15	20.56 ± .35	–	53.97 ± .31
RWGK	80.77 ± .72	–	–	22.37 ± .35	41.29 ± .89	55.91 ± .37

Table 3.3: CPU runtime comparisons on graph datasets

Datasets	MUTAG	NCI1	NCI109	ENZYMES	PPIs	PTC(MR)
JSDKA	8”	7h	7h	10′11”	2′50”	1′8”
JSDKU(Shannon entropy)	1”	1”	1”	1”	1”	1”
JSDKU(von Neumann Entropy)	1”	1”	1”	1”	1”	1”
QJSK	13”	2h55′	2h55′	4′23”	3′24”	1′6”
WLSK	3”	2′30”	2′30”	20”	20”	9”
SPGK	1”	16”	16”	4”	22”	1”
GCGK3	1”	5”	5”	2”	4”	1”
BRWK	11”	6′49”	6′49”	3′5”	> 1 day	29”
RWGK	14”	> 1 day	> 1 day	9′52”	4′26”	2′35”

accuracies and standard errors for each kernel in Table.3.2. Note that, we vary H from 0 to 10 for our JSDKA kernel. As a result, for each dataset we compute 10 kernel matrices for our JSDKA kernel. The classification accuracy for each time is thus the average accuracy over the 10 kernel matrices. Moreover, we also report the runtime of computing the kernel matrices of each kernel in Table.3.3, with the runtime measured under Matlab R2011a running on a 2.5GHz Intel 2-Core processor (i.e., i5-3210m). Note that, both our JSDKA kernel and the WLSK kernel are able to accommodate attributed graphs. In our experiments, the graphs in the PPIs dataset are unattributed. We thus use the vertex degree as a vertex label for the PPIs dataset.

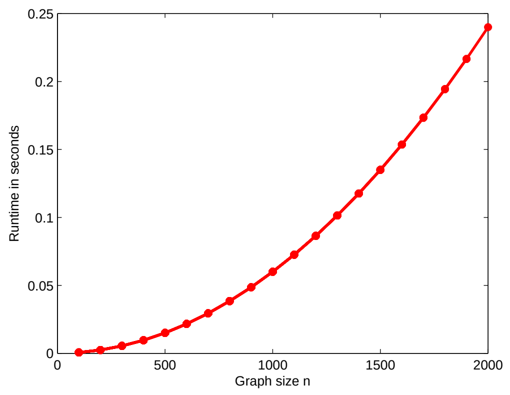
Results and Discussions: In terms of the classification accuracies, it is clear that our

Jensen-Shannon diffusion kernel JSDKA outperforms all these alternative kernels. Only the WLSK subtree kernel is competitive to our kernel. The reason for this is that the WL-SK subtree kernel also relies on a tree-index based algorithm (i.e., the Weisfeiler-Lehman algorithm), and like our kernel can identify the subtrees rooted at each vertex. However, as an example of an R-convolution kernel, the WLSK subtree kernel will discard some subtrees having no isomorphic subtree. By contrast, our JSDKA kernel computes the label Shannon entropy using all the identified subtrees. As a result, our JSDKA k-ernel overcomes the shortcoming of the WLSK subtree kernel and outperforms it on all datasets. Compared to our Jensen-Shannon diffusion kernel JSDKU with either the random walk Shannon entropy or the approximated von Neumann entropy, the performance of our kernel JSDKA is significantly better, although both the kernels are based on the Jensen-Shannon divergence measure between graphs. As we have stated in Section 3.3.3, the reason for this is that the JSDKU kernel can not reflect the interior topology information for graphs, lacks correspondence information, and is also restricted to non-attributed graphs. The kernel JSDKA overcomes the shortcomings that arise in the JSDKU kernel. Moreover, our JSDKA kernel also outperforms the SPGK, BRWK and GCGK graph kernels. The reason for this is that the SPGK, BRWK and GCGK graph kernels are also examples of the R-convolution kernels and have the shortcoming of discarding non-isomorphic substructures.

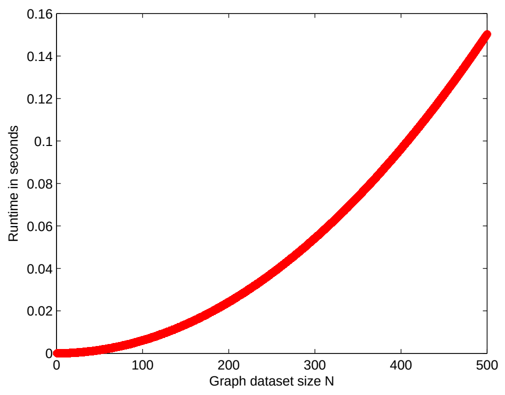
In terms of the runtime, it clearly that our JSDKU kernel, with either the random walk Shannon entropy or the approximated von Neumann entropy, is the most efficient kernel on all datasets, and easily outperforms other kernels. There is no kernel that is competitive to our JSDKU kernel. Moreover, our JSDKA kernel is not the fastest kernel, but it can still finish the computation in a polynomial time on any dataset. By contrast, some kernels cannot finish the computation on some datasets in one day.

3.4.3 Computational Evaluation

Finally, we evaluate the computational efficiency (i.e., the CPU runtime) of our unattributed and attributed Jensen-Shannon diffusion kernels, and reveal the relationship between the computational overheads and the structural complexity or number of the associated graphs.



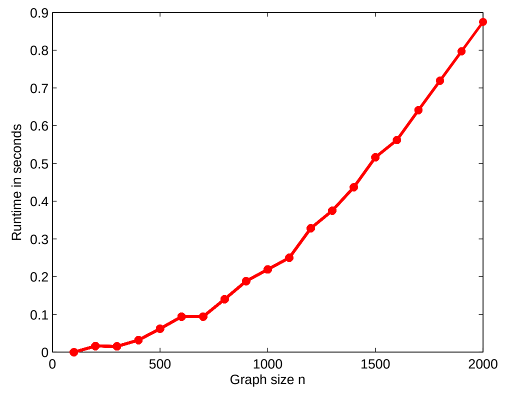
(a) For graph size n



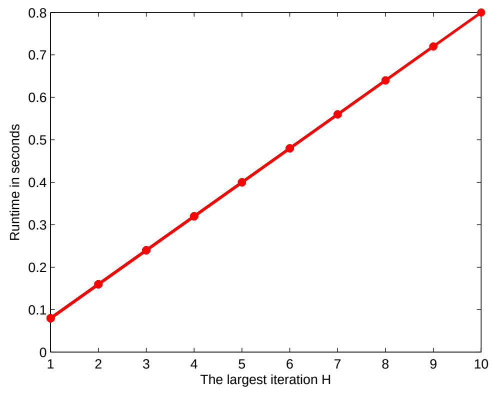
(b) For data size N

Figure 3.2: Runtime evaluations for the unattributed diffusion kernel.

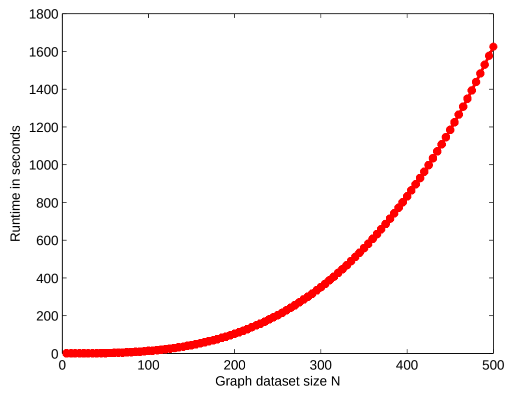
Experimental setup: For either the unattributed or the attributed Jensen-Shannon diffusion kernel, we evaluate the computational efficiency on randomly generated graphs with respect to two parameters: the graph size n and the graph dataset size N. We vary $n = \{100, 200, \ldots, 2000\}$ and $N = \{1, 2, \ldots, 500\}$, separately. a) For the experiments with graph size n, we generate 20 pairs of graphs with increasing number of vertices. We report the runtime for computing the kernel values between pairwise graphs ($H = 10$ for the attributed diffusion kernel). b) For the graph dataset size N, we generate 500 graph datasets with an increasing number of test graphs. In each dataset, one graph has 200 vertices. We report the runtime for computing the kernel matrices for each graph dataset ($H = 10$ for the attributed diffusion kernel). Note that, for the unattributed diffusion kernel the evaluation results with the Shannon and von Neumann entropy are nearly



(a) For graph size n



(b) For largest iteration H



(c) For data size N

Figure 3.3: Runtime evaluations for the attributed diffusion kernel.

the same. Thus, we only show the results for the unattributed diffusion kernel with the Shannon entropy. Furthermore, for our attributed Jensen-Shannon diffusion kernel, we also evaluate the computational efficiency with respect to the largest iteration H of the TI method. Here, we vary $H = \{1, 2, \ldots, 10\}$. For the evaluations with the larger parameter H, we generate a pair of graphs each of which has 200 vertices. We report the runtime for computing the kernel values of the pair of graphs as a function of H. The CPU runtime for the unattributed and the attributed Jensen-Shannon diffusion kernels is reported in Fig.3.2 and Fig.3.3 respectively, as operated in Matlab R2011b on a 2.5GHz

Intel 2-Core processor (i.e., i5-3210m).

Experimental results: Figs.3.2 (a) and (b) show the results for the unattributed Jensen-Shannon diffusion kernel when varying the parameters n and N, respectively. Figs.6.4 (a), (b) and (c) show the results for the attributed Jensen-Shannon diffusion kernel when varying the parameters n, H and N, respectively.

For the unattributed diffusion kernel, we observe that the runtime scales quadratically with n and N. For the attributed diffusion kernel, we observe that the runtime scales quadratically with n, linearly with H, and cubicly with N. These results verify that both of our unattributed and attributed Jensen-Shannon diffusion kernels can be computed in polynomial time.

3.5 Conclusion

In this chapter, we have defined a family of Jensen-Shannon diffusion kernels for (un)attributed graphs using the Jensen-Shannon divergence. For the unattributed graphs, we compute the von Neumann entropy or the random walk Shannon entropy for each graph. With the entropies for a pair of unattributed graphs to hand, we have shown how to compute the Jensen-Shannon diffusion kernel for the unattributed graphs by measuring the entropy difference between their individual entropies and a composite entropy from their disjoint union graph. Our new diffusion kernel for unattributed graphs overcomes the inefficiency arising in the R-convolution kernels. For the attributed graphs, our new kernel is based on an improved tree-index (TI) label strengthening algorithm on attributed graphs. We compute a label Shannon entropy using the probability distribution associated with the strengthened labels. With the entropies for a pair of attributed graphs to hand, we have shown how to compute the Jensen-Shannon diffusion kernel for the attributed graphs by measuring the entropy difference between their individual entropies and a composite entropy from their composite probability distribution. Our new diffusion kernel for attributed graphs overcomes the shortcoming of discarding non-isomorphic substructures that arises in the R-convolution kernels. Moreover, this kernel also overcomes the shortcomings of restriction to unattributed graphs, lacking correspondence information and reflecting limited interior topology information that arise in our diffusion kernel for unattributed graphs. The experimental results demonstrate the effectiveness and efficiency of our kernels.

Chapter 4

Depth-based Complexity Traces of Graphs

In this chapter, we present our second contribution. We develop a novel framework for measuring the depth-based complexity traces for graphs, by linking the ideas of graph entropies and depth-based representations. We commence by reviewing the entropy based graph complexity measures that will be used in this work. We then show how the complexity trace for a graph can be computed by measuring the entropies on a family of centroid expansion subgraphs derived from the graph. The complexity traces for graphs not only capture the structural characteristics but can also be evaluated efficiently. Experimental results on several bioinformatics and computer vision datasets empirically demonstrate that our framework is competitive with complexity based graph methods and alternative state-of-the-art graph based learning methods reported in the literature.

Chapter outline

The remainder of this chapter is organized as follows. Section 4.1 briefly reviews the concepts of several graph entropies. Section 4.2 gives the definitions of the centroid vertex and the centroid expansion subgraphs, and describes how we construct a depth-based complexity trace for a graph. Section 4.3 provides our experimental evaluation. Finally, Section 4.4 concludes our work.

4.1 Entropy-Based Complexity Measures

To compute the depth-based complexity traces for graphs, we need to compute the entropy-based complexity measures for (sub)graphs. In Chapter 3, we have introduced two entropy measures for graphs. In this chapter, we introduce an alternative approach to measuring graph entropies. We review how the Shannon entropy of a graph can be computed using the information functionals proposed by Dehmer et al. [52, 53]. Unlike the classical Shannon entropies defined in [84, 85, 86], the information functionals can assign a probability value to each vertex in a graph, by using a functional that quantifies the structural information of the graph under consideration, it can hence be computed in a polynomial time (i.e., the time complexity of computing the entropy for a graph $G(V, E)$ is $O(|V|^3)$). By contrast, the classical Shannon entropies focus on obtaining the probability distribution by determining a partitioning of the underlying vertex set, and require burdensome computation (i.e., the time complexity is $O(|V|^5)$.

For computing the Shannon entropy of a graph $G(V, E)$, Dehmer et al. [53, 52] have defined two information functionals based on the metrical properties of graphs. They are as follows.

1-**Vertex Functional:** For the graph $G(V, E)$, let $\mathcal{M}_{v_i}^K$ be a subset of the vertices in V satisfying $\mathcal{M}_{v_i}^K := \{v_j \in V | S_G(i, j) = K\}$, where $S_G(i, j)$ is the shortest path length between vertex v_i and vertex v_j. Then for a vertex $v_i \in V$ of $G(V, E)$, the information functional $f^{V_e}, e = 1, 2$ is defined as

$$f^{V_1}(v_i) := \alpha^{c_1|\mathcal{M}_{v_i}^1|+\ldots+c_K|\mathcal{M}_{v_i}^K|+\ldots+c_\rho|\mathcal{M}_{v_i}^\rho|}, \tag{4.1}$$

and

$$f^{V_2}(v_i) := c_1|\mathcal{M}_{v_i}^1| + \ldots + c_K|\mathcal{M}_{v_i}^K| + \ldots + c_\rho|\mathcal{M}_{v_i}^\rho|, \tag{4.2}$$

where ρ is the greatest length of the shortest paths in S_G, $c_K > 0$, $1 \leq K \leq \rho$, $\alpha \leq 0$ and the c_K are arbitrary real positive coefficients. Details of selecting effective c_K can be found in [53].

2-**Path Functional:** For the vertex $v_i \in V$ of $G(V, E)$ and $K = 1, ..., \rho$, let $L_\varphi(v_i, K)$ denote the sum of the shortest paths from v_i to each vertex $v_\mu \in \mathcal{M}^K_{v_i}$. Note that, if there are more than one shortest path from v_i to v_μ, we only consider one random shortest path and we thus have

$$L_\varphi(v_i, K) := K|\mathcal{M}^K_{v_i}|. \tag{4.3}$$

Then the information functional $f^{P_e}, e = 1, 2$ is defined as

$$f^{P_1}(v_i) := \alpha^{b_1 L_\varphi(v_i,1)+...+b_K L_\varphi(v_i,K)+...+b_\rho L_\varphi(v_i,\rho)}, \tag{4.4}$$

and

$$f^{P_2}(v_i) := b_1 L_\varphi(v_i, 1) + \ldots + b_K L_\varphi(v_i, K) + \ldots + b_\rho L_\varphi(v_i, \rho), \tag{4.5}$$

where b_K are arbitrary real valued positive coefficients. Details of selecting effective b_K can be found in [53].

In order to deal with graphs with large sizes, we propose to use the information functionals f^{V_2} and f^{P_2} to construct graph entropies. The entropies of the sample graph $G(V, E)$ associated with the information functional f^{V_2} and f^{P_2} are defined as

$$H_{f^{V_2}}(G) = -\sum_{i=1}^{|V|} \frac{f^{V_2}(v_i)}{\Sigma_{j=1}^{|V|} f^{V_2}(v_j)} \log \frac{f^{V_2}(v_i)}{\Sigma_{j=1}^{|V|} f^{V_2}(v_j)}, \tag{4.6}$$

and

$$H_{f^{P_2}}(G) = -\sum_{i=1}^{|V|} \frac{f^{P_2}(v_i)}{\Sigma_{j=1}^{|V|} f^{P_2}(v_j)} \log \frac{f^{P_2}(v_i)}{\Sigma_{j=1}^{|V|} f^{P_2}(v_j)}. \tag{4.7}$$

Selection of parameters c_K and b_K: In this chapter, we follow Dehmer in [53] and choose the coefficients c_K and b_K of f^{V_e} and f^{P_e} as $\rho - K + 1$ with the objective of emphasizing certain structure characteristics of the underly graph, e.g., high vertex degrees.

Based on the statement of Dehmer et al. [53], for the graph $G(V, E)$ with $|V|$ vertices, the Shannon entropies $H_{f^{V_2}}(G)$ and $H_{f^{P_2}}(G)$ associated with information functionals require time complexity $O(|V|^3)$.

4.2 Depth-Based Complexity Traces of Graph Structures

In this section we combine the idea of graph entropies with that of using a depth-based representation to develop a novel depth-based complexity trace for a graph. Our idea is to decompose a graph into substructures (i.e., subgraphs) spanned from a root vertex to the remaining vertices with a minimal path length K. We then compute the complexities on these substructures as a measure of the information content flow over the substructures. To obtain a family of subgraphs capturing the fine structure of a graph, we identify a centroid vertex and use this as the root vertex. The terminology *centroid* is widely used in several fields including geometry and physics. In graph theory, the centroid of a graph is defined as a structure composed of vertices closest to all others [87, 88, 89]. Most of the classical definitions of a centroid focus on two specific tasks, namely 1) minimax location problems [90] and 2) minisum location problems [89]. The first task aims to locate the centroid by finding the vertices possessing the minimal longest distance to all the remaining vertices. The second task aims to locate the centroid by finding the vertices possessing the minimal sum of the shortest distance to the remainders. Here we present a novel method to identify the centroid vertex of a graph by evaluating the shortest path length distribution around a vertex. We select the vertex possessing the minimum variance of shortest path lengths to the remaining vertices as the centroid vertex. Since the vertices surrounding the centroid vertex in a graph lie along the different shortest paths from the vertex, the centroid vertex has a global view of the vertex path length distribution surrounding it.

4.2.1 Centroid Vertex and Centroid Expansion Subgraphs

The shortest path for a pair of vertices v_i and v_j in an undirected graph $G(V, E)$ can be obtained using Dijkstra's algorithm [91]. Let S_G be the matrix whose elements $S_G(i, j)$ represent the shortest path length between vertices v_i and v_j is the shortest path matrix for graph $G(V, E)$. The average-shortest-path vector S_V for $G(V, E)$ is a vector with the same vertex sequence as S_G and with element $S_V(i) = \sum_{j=1,i\neq j}^{|V|} S_G(i,j)/|V|$ representing the average shortest path length from vertex v_i to the remaining vertices. We then locate the centroid vertex $\hat{v}_i$ for $G(V,E)$ as follows

$$\hat{v}_i = \arg\min_i \sum_{j=1}^{|V|} [S_G(i,j) - S_V(i)]^2. \tag{4.8}$$

In other words, the centroid vertex $\hat{v}_i$ of $G(V,E)$ is located through selecting a vertex with a minimum variance of shortest path lengths for all vertices in $G(V,E)$. As a result, the shortest paths originating from the centroid vertex $\hat{v}_i$ form a *steady* path set that exhibits the least length variability compared with those path sets originating from the remaining vertices.

Let $N_{\hat{v}_C}^K$ be a subset of V satisfying the condition

$$N_{\hat{v}_C}^K = \{u \in V \mid S_G(\hat{v}_C, u) \leq K\}. \tag{4.9}$$

i.e., the set of vertices whose path lengths to the centroid vertex are less than or equal to K. For a graph $G(V,E)$ with the centroid vertex $\hat{v}_C$, the K-layer centroid expansion subgraph $\mathcal{G}_K(\mathcal{V}_K;\mathcal{E}_K)$ is

$$\begin{cases} \mathcal{V}_K &= V \cap N_{\hat{v}_C}^K; \\ \mathcal{E}_K &= \{(u,v) : u,v \in N_{\hat{v}_C}^K,\ (u,v) \in E\}. \end{cases} \tag{4.10}$$

Note that, if there are more than one vertex which satisfy Eq.(4.8), we can simply randomly choose one as the centroid vertex. Furthermore, the number of centroid expansion subgraphs is equal to the greatest length L^{max} of the shortest path from the centroid vertex to the remaining vertices of the graph. The L^{max}-layer expansion subgraph is the graph $G(V,E)$ itself. An example of the generation of a K-layer subgraph for a graph $G(V,E)$ is shown in Fig.4.1.

For the graph $G(V,E)$, which has $|V|$ vertices, constructing the family of centroid expansion subgraphs from the centroid vertex requires time complexity $O(L^{max}|V|^2)$,

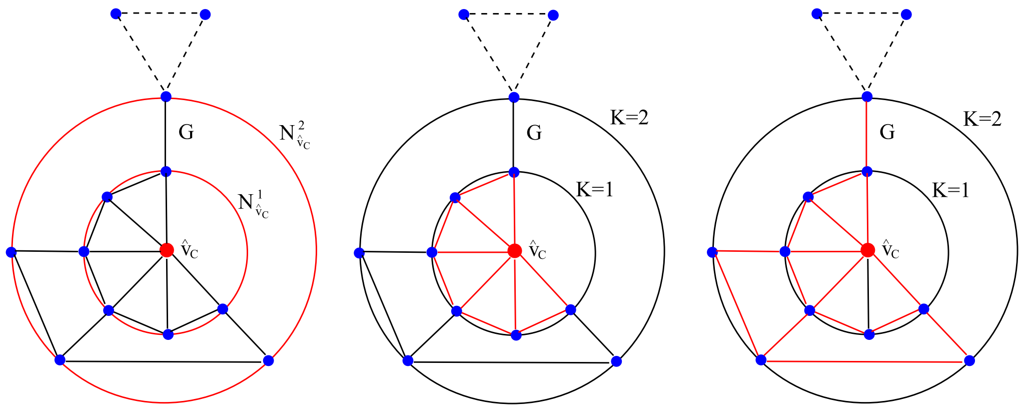


Figure 4.1: The left-most figure shows the determination of K-layer centroid expansion subgraphs for a graph $G(V, E)$ which hold $|N^1_{\hat{v}_C}| = 6$ and $|N^2_{\hat{v}_C}| = 10$ vertices. While the middle and the right-most figure show the corresponding 1-layer and 2-layer subgraphs regarding the centroid vertex $\hat{v}_C$, and are depicted by red-colored edges. In this example, the vertices of different K-layer subgraphs regarding the centroid vertex $\hat{v}_C$ are calculated by Eq.(4.9), and pairwise vertices possess the same connection information in the original graph $G(V, E)$.

because this follows the definitions in Eq.(4.9) and Eq.(4.10). For the graph $G(V, E)$, the Dijkstra algorithm requires time complexity $O(|V|^2)$. The enumeration of the centroid expansion subgraphs $\mathcal{G}_K$ ($K = 1, \ldots, L^{max}$) of $G(V, E)$ requires $O(L^{max})$ operations, in each of which the computation of the vertex degrees for a centroid expansion subgraph scales up to $O(|V|^2)$. As a result, constructing the centroid expansion subgraphs requires time complexity $O(L^{max}|V|^2)$.

4.2.2 Entropy Complexity Traces of Graphs

We first define a depth-based entropy complexity trace for a graph $G(V, E)$.

Definition 4.1 (Entropy complexity trace) For a graph $G(V, E)$ and its K-layer centroid expansion subgraphs, the entropy complexity trace CT^E is an $L^{\max}$ dimensional vector defined as

$$CT^E = [H_E(\mathcal{G}_1), \cdots, H_E(\mathcal{G}_K), \cdots, H_E(\mathcal{G}_{L^{\max}})]^T, \tag{4.11}$$

where $L^{\max}$ is the greatest length of shortest paths from the centroid vertex $\hat{v}_C$ to the remaining vertices in $G(V, E)$, $\mathcal{G}_K$ is the K-layer centroid expansion subgraph of $G(V, E)$, and $H(\mathcal{G}_K)$ is the entropy of $\mathcal{G}_K$.

Here the entropy function $H_E(\cdot)$ could be either the von Neumann entropy $H_{VN}(\cdot)$ defined in Eq.(3.7), the Shannon entropy $H_S(\cdot)$ defined in Eq.(3.9), or the information functional entropies $H_{f^{V_2}}(\cdot)$ and $H_{f^{P_2}}(\cdot)$ in Eq.(4.6) and Eq.(4.7).

4.2.3 Entropy Difference Complexity Traces of Graphs

In this subsection, we investigate how to use the entropy difference as a means of constructing the complexity trace for a graph. We define the depth-based entropy difference complexity trace for a graph.

Definition 4.2 (Entropy difference complexity trace) For a graph $G(V, E)$ and its K-layer centroid expansion subgraphs, the entropy difference complexity trace CT^{ED} is an $L^{\max}$ dimensional vector defined as

$$\begin{aligned} CT^{ED} =& [ED(\mathcal{G}_1, \mathcal{G}_{L^{\max}}), ..., ED(\mathcal{G}_K, \mathcal{G}_{L^{\max}}), ..., \\ & ED(\mathcal{G}_{L^{\max}}, \mathcal{G}_{L^{\max}})]^T, \end{aligned} \tag{4.12}$$

where $L^{\max}$ is the greatest length of shortest paths from the centroid vertex $\hat{v}_C$ to the remaining vertices in $G(V, E)$, and $ED(\mathcal{G}_K, \mathcal{G}_{L^{\max}})$ is the entropy difference between the K-layer centroid expansion subgraph and the L^{max}-layer centroid expansion subgraph (i.e., graph $G(V, E)$).

Unlike the entropy complexity trace defined in Definition 4.1 that only records how the entropies vary from the smaller subgraph to the global graph, the entropy difference complexity trace gauges how the entropy difference varies between the subgraphs and the global graph, and thus also reflects the relationship between each subgraph and the global graph. In other words, this complexity trace encapsulates an information-based interior dissimilarity transformation between the graph $G(V, E)$ and its K-layer centroid expansion subgraphs associated with their entropy differences. Here the layer K runs from 1 to $L^{\max}$. The entropy difference $ED(\mathcal{G}_K, \mathcal{G}_{L^{\max}})$ is defined as

$$ED(\mathcal{G}_K, \mathcal{G}_{L^{\max}}) = H_E(\mathcal{G}_{L^{\max}}) - H_E(\mathcal{G}_K). \tag{4.13}$$

Similar to the entropy complexity traces of graphs, here the entropy function $H_E(\cdot)$ could be the von Neumann entropy $H_{VN}(\cdot)$ defined in Eq.(3.7), the Shannon entropy $H_S(\cdot)$ defined in Eq.(3.9), or the information functional entropies $H_{f^{V_2}}(\cdot)$ and $H_{f^{P_2}}(\cdot)$ defined in Eq.(4.6) and Eq.(4.7).

4.2.4 Graphs of Different Sizes

For a graph $G(V, E)$, the dimension of the complexity trace vectors are equal to the size of the largest layer L^{max} which is the length of the greatest shortest path from the centroid vertex to the remaining vertices. Since the lengths of the greatest shortest paths from the centroid vertices for different graphs of different sizes may be different, hence the complexity trace vectors for graphs of different sizes may exhibit different dimensions. To compare these graphs by using complexity trace vectors, we need to make the vector dimensions uniform. This is achieved by padding out the dimensions of the complexity trace vectors. For entropy or entropy difference complexity trace vectors CT_x and CT_y of two graphs G_x and G_y with dimensions L_x^{max} and L_y^{max} respectively, where $L_x^{max} > L_y^{max}$, we use the L_y^{max}-th element value of CT_y as the padding values for the extended $L_y^{max}+1$-th to L_x^{max}-th elements of CT_x.

4.2.5 Computational Complexity Evaluation

The computational complexity of the proposed complexity traces are governed by the following computational steps. Consider a sample graph $G(V, E)$ with $|V|$ vertices and the longest shortest path length L^{max} rooted from the centroid vertex. Constructing the centroid expansion subgraphs derived from the centroid vertex requires time complexity $O(L^{max}|V|^2)$. In our framework, the computations of the required approximated von Neumann entropy, the Shannon entropy associated with the steady state random walk, and the Shannon entropy associated with the information functionals of the centroid expansion subgraphs require time complexities $O(L^{max}|V|^2)$, $O(L^{max}|V|^2)$ and $O(L^{max}|V|^3)$ respectively. As a result, the whole time complexity of the complexity traces using the three different kinds of entropies are $O(L^{max}|V|^2)$, $O(L^{max}|V|^2)$ and $O(L^{max}|V|^3)$ respectively. L^{max} is approximately equal to $\sqrt[3]{|V|}$.

Therefore, our depth-based complexity traces can be computed in polynomial time. The reason for this is that we efficiently compute the required entropies on a small set of expansion subgraphs rooted at the centroid vertex of a graph. By contrast, the depth-based complexity measure described in [17] establishes expansion subgraphs from each vertex of a given graph (e.g., a graph having $|V|$ vertices) and then computes the burdensome intrinsic complexities on the subgraphs. It hence requires time complexity $O(|V|^7)$ [17].

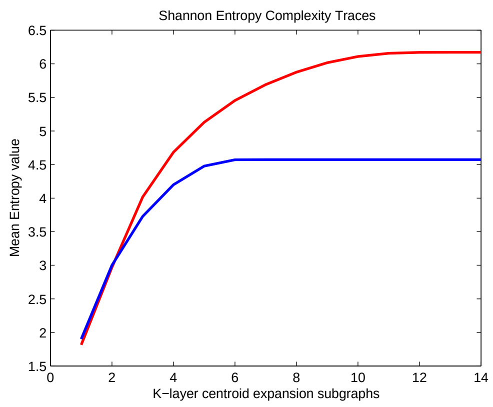
4.3 Experimental Results

In this section, we provide experimental evaluation of the proposed depth-based complexity traces of graphs. First, we illustrate the discriminative power of the complexity traces. Second, we present experiments on synthetic data with the aim of evaluating the stability of our methods and their ability to distinguish graphs under controlled structural-errors. Third, we evaluate the computational efficiency of the complexity traces. Finally, we focus on real-world graph data and assess the performance of the proposed complexity traces of graphs for graph classification problems. The graphs for testing are abstracted from a) a real-world image database and b) bioinformatics databases. We provide comparisons between the proposed methods and several alternative methods reported in the literatures.

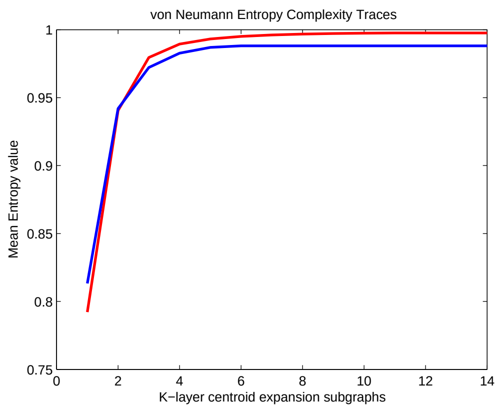
4.3.1 Evaluation of Interior Complexity Traces

We commence by illustrating the representational power of the proposed complexity traces for graphs, and demonstrate that they can be used to distinguish different graphs. The evaluation utilizes graphs abstracted separately from images of a box and a cup in the Columbia object image library (COIL) database [92]. For each object we use 18 images captured from different viewpoints. For each image we first extract corner points using the Harris detector [93], and then establish Delaunay graphs based on the corner points as vertices. Each vertex is used as the seed of a Voronoi region, which expands radially with a constant speed. The linear collision fronts of the regions delineate the image plane into polygons, and the Delaunay graph is the region adjacency graph for the Voronoi polygons. Details of constructing a Delaunay graph of an image can be found in [3].

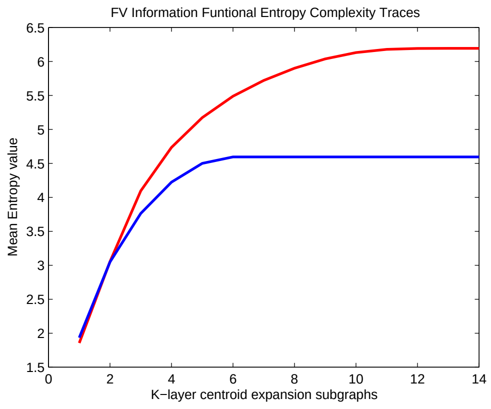
For the Delaunay graph of each image, we locate the centroid vertex and construct the centroid expansion subgraphs. We then construct the proposed entropy and entropy difference complexity traces. Fig.4.2 and Fig.4.3 show the mean complexity trace distributions of graphs for the entropy complexity trace and the entropy difference complexity trace respectively. The subfigures (a)-(d) of Fig.4.2 and 4.3 show the corresponding mean complexity trace distributions using the Shannon entropy, von Neumann entropy, and entropies associated with information functionals f^{V_2} and f^{P^2} respectively. In Fig.4.2 and Fig.4.3, the x-axis shows the order of the K-layer centroid expansion subgraph for each individual graph, while the y-axis shows the mean entropy value or the mean entropy difference as a function of the expansion order. Here the blue line represents the mean complexity trace of the graphs abstracted from the box object, while the red line represents that of the graphs abstracted from the cup object. The main feature to note is that the mean complexity trace distributions of graphs abstracted from the different objects



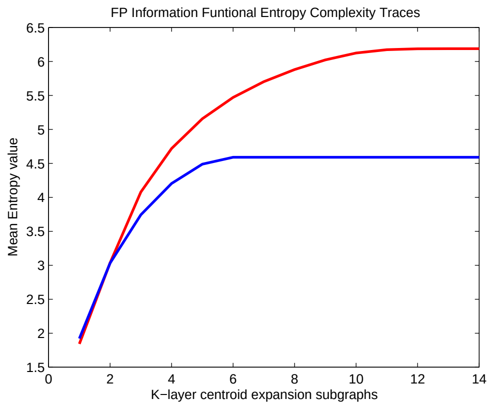
(a)



(b)

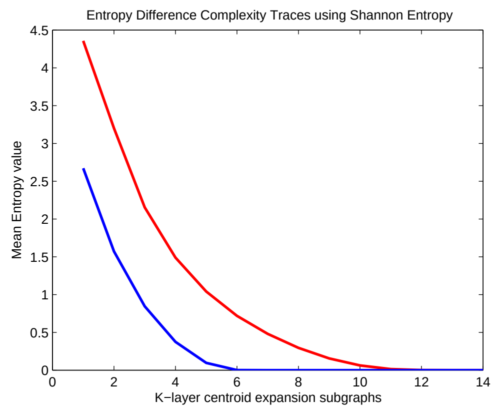


(c)

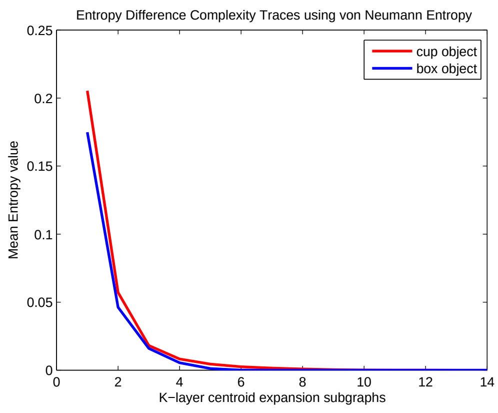


(d)

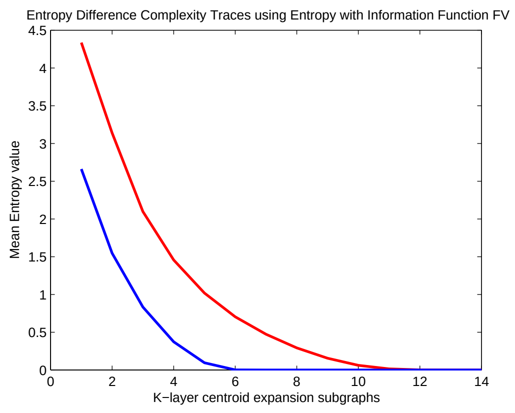
Figure 4.2: (a)-(d): Entropy complexity trace distributions using the a) Shannon entropy, b) von Neumann entropy, and entropies computed using information functionals c) f^{V_2} and d) f^{P_2}.



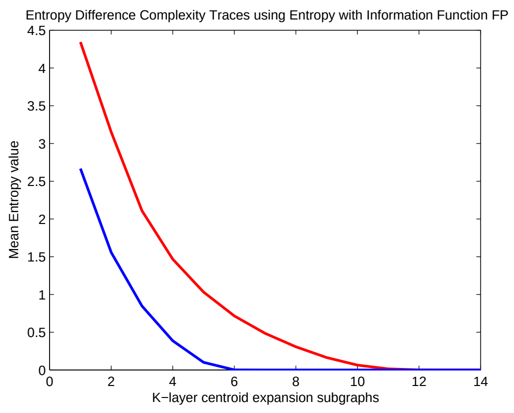
(a)



(b)



(c)



(d)

Figure 4.3: (a)-(d): Entropy difference complexity trace distributions using the a) Shannon entropy, b) von Neumann entropy, and entropies computed using information functionals c) f^{V_2} and d) f^{P_2}.

are dissimilar. The mean complexity traces of graphs abstracted from the different objects using the entropy difference are better separated than that abstracted using the graph entropy. The mean entropy and the entropy difference complexity traces of graphs computed using the von Neumann entropy show the best separation when compared to the other alternative entropies.

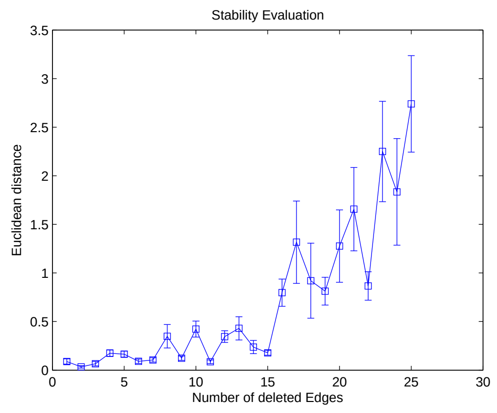
4.3.2 Stability Evaluation of Complexity Traces

To evaluate the stability of our proposed complexity traces from the centroid vertex, we explore the relationship between the graph edit distance and the Euclidean distance between pattern vectors encoding the complexity traces. The edit distance between two graphs G_x and G_y is the minimum edit cost taken over all sequences of edit operations that transform G_x into G_y [94, 95]. In our experimental evaluation, we establish a new graph by deleting a number of edges from a seed graph. Compared to the seed graph, the location of the centroid vertex of the new graph may produce changes. The evaluation utilizes 100 randomly generated seed graphs. Each seed graph has 100 vertices and 200 edges. For each seed graph, we randomly delete a predetermined number of edges to simulate the effects of noise. We continuously apply the edge deletion edit operation to the seed graph 25 times. We delete 1 edge for each time and thus generate 25 edit-operated graphs as the noise corrupted counterparts, for each of the seed graphs.

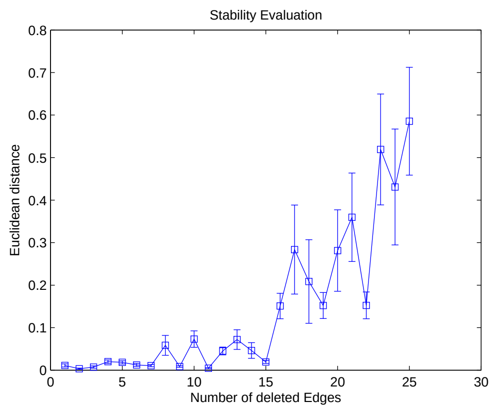
The Euclidean distance between the complexity traces of the original seed graph G_S and a noise corrupted counterpart G_E obtained by the edit operation is

$$d_{S,E} = \sqrt{(CT_S - CT_E)^T (CT_S - CT_E)}, \tag{4.14}$$

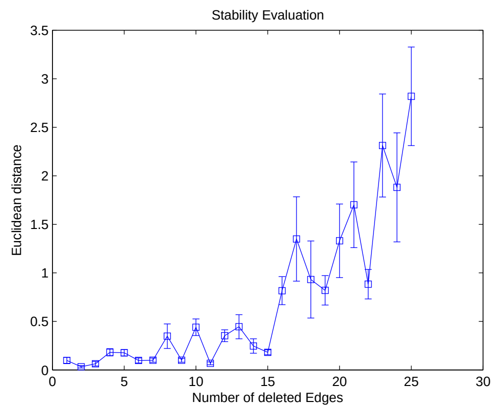
where CT_S and CT_E are their entropy or entropy difference complexity traces of G_S and G_E from their centroid vertex. The results of these experiments are shown in Fig.4.4 and Fig.4.5, which show the the mean effects of edge deletion on the entropy and entropy difference complexity traces respectively. The subfigures (a)-(d) of both Fig.4.2



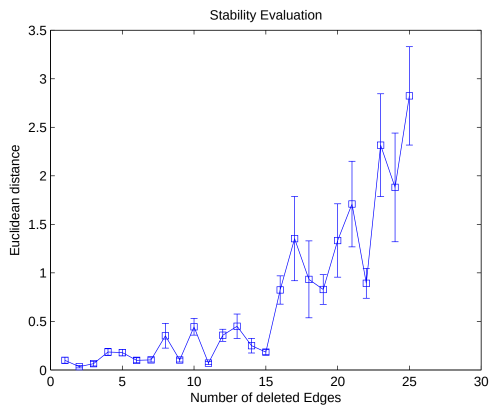
(a)



(b)



(c)



(d)

Figure 4.4: (a)-(d): Distance distributions of entropy complexity traces using the a) Shannon entropy, b) von Neumann entropy, and entropies computed using information functionals c) f^{V_2} and d) f^{P_2}.

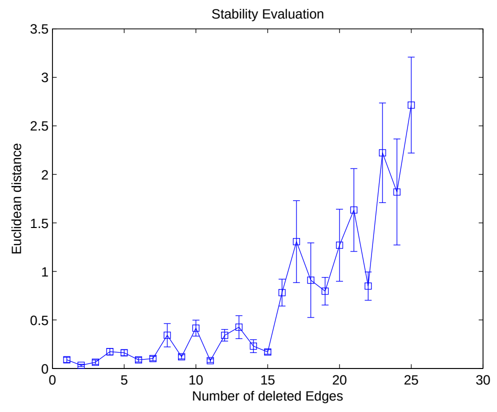
and 4.3 represent the corresponding complexity traces computed using the Shannon entropy, von Neumann entropy, and entropies with information functionals f^{V_2} and f^{P_2} respectively. The x-axis shows the number of edges randomly deleted (i.e., the 1-th to 25-th edit-operated graphs), and the y-axis shows the mean value of the Euclidean distances $d_{S,E}$ between the 100 original seed graphs and their corresponding noise corrupted counterparts. Moreover, we also draw the stand errors for each of the mean Euclidean distances.

It is clear that when less than 15 edges are deleted the fluctuation is small, and when around 20 edges are deleted the fluctuation becomes moderate. On the other hand, when less than 5 edges are deleted the stand errors are small, when less than 15 edges are deleted the stand errors are moderate, and when more than 15 edges are deleted the stand errors tend to be larger. The evaluation results imply that the proposed complexity traces are robust even when each of the seed graph structures and its identification of the centroid vertex undergo relatively large perturbations. Since the location of the centroid vertices of the edit-operated graphs may gradually get far away from that of a seed graph with the increasing number of deleted edges, the evaluation results also reveal that for a graph the complexity traces from other vertices tend to be similar to those from the centroid vertex if the vertices are near to the centroid vertex. Furthermore, there is an approximately linear relationship between the graph edit distance and the Euclidean distance. This implies that the proposed complexity traces possess the ability to distinguish graphs under controlled structural-errors.

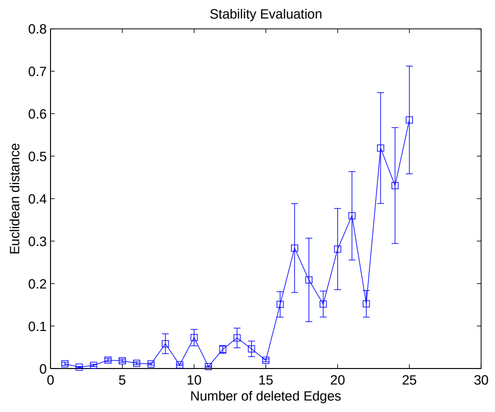
4.3.3 Computational Evaluation of The Proposed Complexity Traces

In this subsection, we evaluate the computational efficiency (i.e., the CPU runtime) of the complexity traces, and reveal the relationship between their computational overheads and the structural complexity or number of the associated graphs.

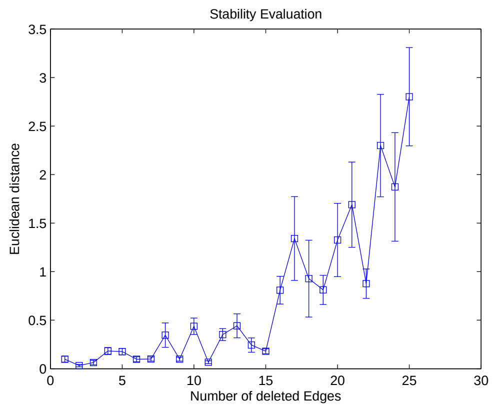
Experimental setup: We evaluate the computational efficiency on randomly generat-



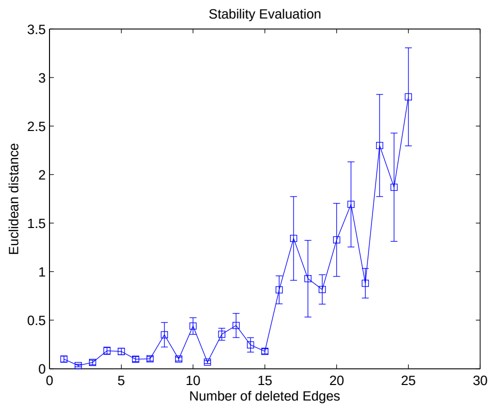
(a)



(b)



(c)



(d)

Figure 4.5: (a)-(d): Distance distributions of entropy difference complexity traces using the a) Shannon entropy, b) von Neumann entropy, and entropies computed using information functionals c) f^{V_2} and d) f^{P_2}.

ed graphs with respect three parameters: a) the graph size n, b) the greatest expansion subgraph layer L^{max} of a graph, and c) the dataset size N. Separately, we vary $n = \{100, 200, \ldots, 2000\}$, $L^{max} = \{1, 2, \ldots, 41\}$ and $N = \{1, 2, \ldots, 300\}$.

For the experiments with graph size n, we generate 20 graphs having increasing number of vertices. We report the runtime of computing the complexity traces for each graph. For the experiments with greatest expansion subgraph layer L^{max}, we randomly generate a graph with 8500 vertices. We report the runtime of computing the complexity traces of the graph with an increasing expansion subgraph layer L^{max}. For the experiments with graph dataset size N, we generate 300 graph datasets. The datasets have an increasing number of test graphs. For each dataset, each graph has 100 vertices. We report the runtime of computing the complexity traces of graphs from each dataset.

Note that, in the computational evaluation each test graph is a scale-free graph. For generating a random scale-free graph, we commence by generating a graph having 5 vertices as the initialized graph. In the initialized graph, each vertex is connected to all other vertices. With the initialized graph, we gradually add some new vertices to generate a required scale free graph. We add 5 new vertices each time, and each new vertex is randomly connected to 5 vertices of existing vertices. The CPU runtime is reported in seconds in Fig.6.4, as measured in Matlab R2011a on a 2.5GHz Intel 2-Core processor (i.e., i5-3210m).

Experimental results: In Fig.6.4, the subfigures from the first row to the fourth row represent the experiments of the complexity traces using the a) Shannon entropy, b) the approximated von Neumann entropy, c) the vertex-information functional entropy (i.e., f^{V_2}), and d) the path-information functional entropy (i.e., f^{P_2}) respectively. The subfigures from the first column to the third column represent the experiments of the complexity traces varying the parameters n, L^{max} and N respectively. Note that, for each parameter α of the information functionals the experiments using the different information functional entropies are the same.

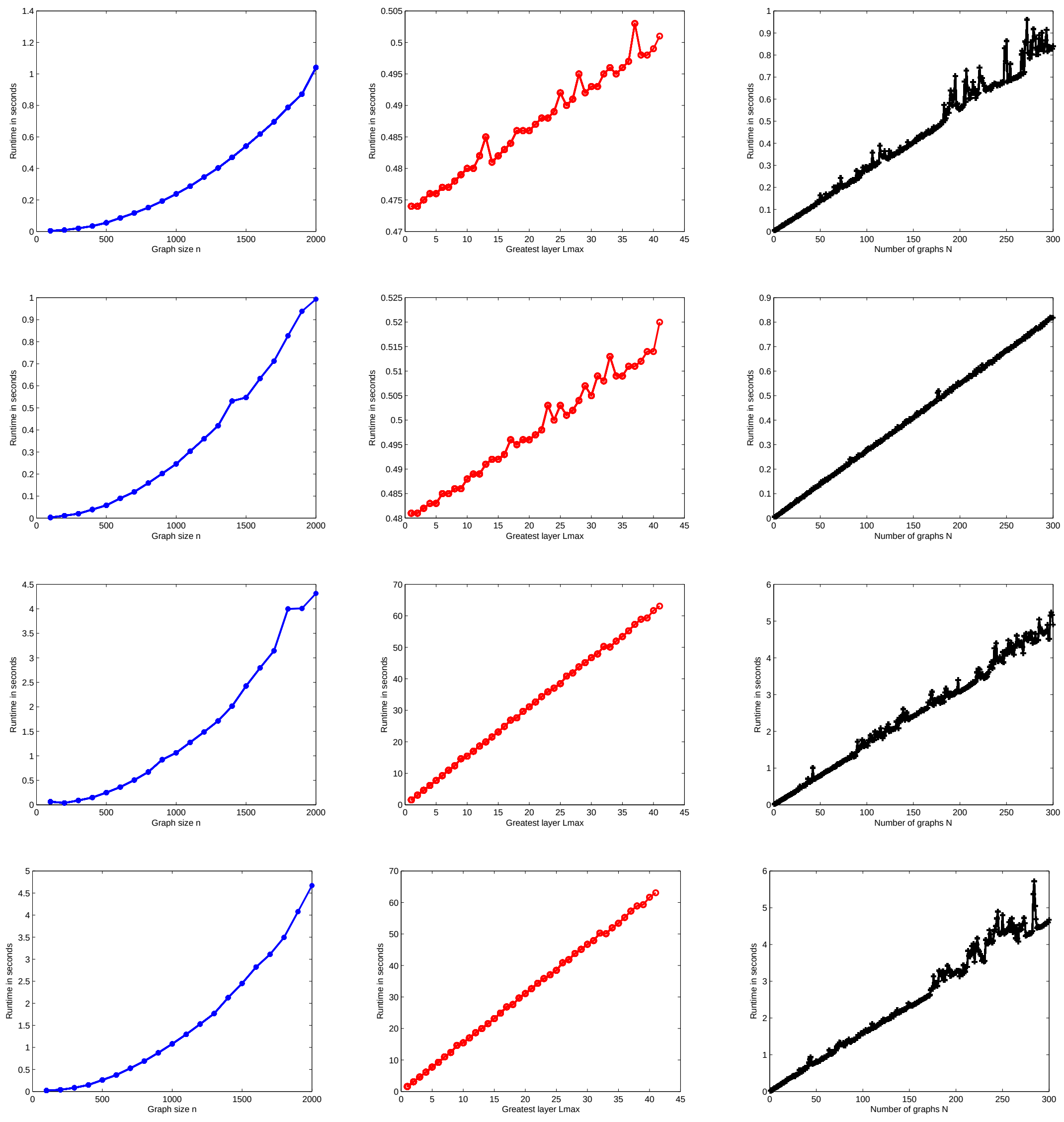


Figure 4.6: Runtime evaluations.

From these plots, we can draw the following conclusions. a) When varying the number of vertices n of a graph, we observe that the runtime of the complexity traces using the different entropies scales quadratically or cubically with n. b) When varying the greatest expansion subgraph layer L^{max} of a graph, we observe that the runtime of the complexity traces using different entropies scales linearly with L^{max}. c) When varying the dataset size N, we observe that the runtime of the complexity traces using the different entropies scales linearly with N. d) For any parameter, the complexity traces using the Shannon entropy and the approximated von Neumann entropy are computed more rapidly than those using the information functional entropies.

These computational experiments verify that our complexity traces can be computed in polynomial time. The complexity traces using the Shannon entropy and the approximated von Neumann entropy are more efficient than those using the information functional entropies. The reason for this is that the computations of the Shannon entropy and the approximated von Neumann entropy are quadratic in the number of vertices of a graph. On the other hand, the computations of the information functional entropies are cubic in the number of vertices of a graph.

4.3.4 Real-world Datasets

We compare our proposed complexity trace methods with several state of the art methods. The methods for comparisons are a) graph complexity based methods, b) a graph embedding method, and c) graph kernel methods. The graph complexity based methods include 1) the von-Neumann thermodynamic depth complexity (VNTD) [15, 17], 2) the von-Neumann graph entropy (VNGE) [15], 3) the Shannon graph entropy associated with the steady state random walk (SGE), 4) the information functionals f^{V_1} (FV1), f^{V_2} (FV2), f^{P_1} (FP1) and f^{P_2} (FP2) [52]. Here we set the parameters α of FV1 and FP1 as 2. The graph embedding method is the coefficients from the Ihara zeta function for graphs (CIZF) [4]. The graph kernel methods include 1) the Weisfeiler-Lehman subtree kernel (WL) [7], 2) the random walk graph kernel (random walk) [22], 3) the shortest path graph kernel (SPGK) [39], and 4) the graphlet count graph kernel (graphlet count) with size 4. We use eight standard graph based datasets abstracted from the bioinformatics database for experimental evaluation [17, 77, 78, 79, 80, 96]. These datasets include: MUTAG, NCI1, NCI109, ENZYMES, D&D, PPIs, CATH1 and CATH2. The MUTAG, NCI1, NCI109, ENZYMES, D&D and PPIs datasets have been introduced in Chapter 3. The detail information of the CATH1 and CATH2 datasets is shown as below.

CATH1 and CATH2: The CATH1 dataset consists of proteins in the same class (i.e., Mixed Alpha-Beta), but the proteins have different architectures (i.e., Alpha-Beta Barrel vs. 2-layer Sandwich). CATH2 has proteins in the same class (i.e., Mixed Alpha-Beta), architecture (i.e., Alpha-Beta Barrel), and topology (i.e., TIM Barrel), but in different homology classes (i.e., Aldolase vs. Glycosidases). The CATH2 dataset is harder to classify, since the proteins in the same topology class are structurally similar. The protein graphs are 10 times larger in size than chemical compounds, with $200 - 300$ vertices. There are 712 and 190 testing graphs in the CATH1 and CATH2 datasets.

More information concerning all these datasets for evaluation are summarized in Table.4.1.

Table 4.1: Information of the graph based bioninformatics datasets

Datasets	MUTAG	NCI1	NCI109	ENZYMES	D&D	PPIs	CATH1	CATH2
Max # vertices	28	111	111	126	5748	232	568	568
Min # vertices	10	3	4	2	30	3	44	143
Mean # vertices	17.93	29.87	29.68	32.63	284.32	109.60	205.70	308.03
Number of graphs	188	4110	4127	600	1178	86	712	190
Number of disjoint graphs	0	580	608	31	21	0	18	7
Proportion of disjoint graphs	0%	14.11%	14.73%	5.16%	1.7%	0%	2.53%	3.68%

4.3.5 Experiments on Bioinformatics Datasets

Experimental setup: We evaluate the performance of our proposed depth-based complexity traces on the graph datasets abstracted from the bioinformatics database. We also compare them with alternative state-of-the-art graph based learning methods mentioned in Section 4.3.4. For our proposed methods, the graph complexity based methods and the CIZF, we calculate the vectors or characterization values of graphs as features. We then perform 10-fold cross-validation using the Support Vector Machine Classification (SVM) associated with the Sequential Minimal Optimization (SMO) [97] and the Pearson VII universal kernel (PUK) [98] to evaluate the performance of our methods and the alternative methods. We use nine folds for training and one fold for testing. For each method, we repeat the experiments 10 times. All parameters of the SMO-SVMs were optimized for each method on different datasets on a Weka workbench. We report the average classification accuracies of each method in Tables 4.2 and 4.3. The runtime is measured under a Matlab R2011a running on a ThinkPad T61p with an Intel(T7500) 2.2GHz 2-Core processor and 2GB RAM. The runtime of each method is shown in Tables 4.5 and 4.6. Here ECTS, ECTV, ECTFV and ECTFP represent the entropy complexity traces associated the Shannon entropy, von Neumann entropy, and Shannon entropies associated with information functionals f^{V_2} and f^{P_2} respectively, while EDCTS, EDCTV, EDCTFV and EDCTFP represent the entropy difference complexity trace using the same entropies respectively. Note that, for a sample graph the entropy difference complexity trace can be seen as a lineal transformation from the entropy complexity trace, by just adding the negative largest layer expansion subgraph entropy value for each element of the entropy complexity trace. This can be observed from Eq.(4.11), Eq.(4.12) and Eq.(4.13). However, the experimental results for both the entropy and entropy difference complexity traces will still be different. The reason for this is that the entropy values of the largest layer expansion subgraphs from different sample graphs may be different, since each largest layer expansion subgraph is a corresponding sample graph itself. In other words, for different entropy traces, the entropy difference complexity traces are not computed by shifting the same distance for the entropy complexity traces in the original principle space.

We also compare our proposed methods with several state-of-the-art graph kernels mentioned in Section 4.3.4. For each graph kernel method, we compute the kernel matrix on each dataset. For each kernel matrix we perform Principle Component Analysis (PCA) [25] to embed the graphs into a feature space as vectors. Any standard machine learning algorithm can hence be performed to classify the graphs in the principle component feature space. We also perform 10-fold cross-validation using the SMO-SVM Classification to evaluate the performance of the kernels. We use nine folds for training and one fold for testing. We repeat the experiments for 10 times. We report the average accuracies of different kernels on different datasets. The runtime of these methods were measured under a Matlab R2011a on a 2.5GHz Intel 2-Core processor (i.e., i5-3210m). We report these accuracies and runtime in Tables 4.4 and 4.7.

Table 4.2: Performance of proposed complexity traces

Datasets	ECTS	ECTV	ECTFV	ECTFP	EDCTS	EDCTV	EDCTFV	EDCTFP
MUTAG	85.10	**88.29**	85.63	85.63	85.63	82.44	81.38	81.38
NCI1	68.32	69.74	66.95	67.73	67.49	67.95	66.37	66.13
NCI109	68.96	69.81	68.13	67.91	66.94	65.93	65.44	64.33
ENZYMES	38.00	**38.83**	35.00	35.33	29.00	32.16	27.17	28.33
D&D	76.49	75.89	**77.58**	77.00	75.32	76.15	73.09	73.26
PPIs	73.25	76.74	73.25	75.58	76.74	77.90	79.07	79.07
CATH1	**98.87**	98.73	97.75	97.75	94.80	88.76	94.10	94.38
CATH2	78.42	**80.47**	78.94	**80.47**	77.89	72.10	77.36	77.36

The unit of an accuracy value is %.

Experimental Results: The graphs in the D&D dataset have on average more than 284 vertices and at maximum 5748 vertices. The result for the D&D dataset from our entropy complexity trace with the entropy associated with the information functional f^{V_2} achieves the highest accuracy for the proposed complexity traces. The accuracy of the complexity

Table 4.3: Performance comparisons of graph complexity based methods and the CIZF

Datasets	VNTD	VNGE	SGE	FV1	FV2	FP1	FP2	CIZF
MUTAG	83.51	85.10	87.76	84.57	84.57	85.63	85.63	80.85
NCI1	$--$	62.15	61.84	62.04	62.04	62.09	62.09	60.05
NCI109	$--$	62.05	62.05	62.15	62.15	62.37	62.37	62.79
ENZYMES	30.50	22.33	23.16	24.17	24.17	23.33	23.33	32.00
D&D	$--$	74.70	75.46	$--$	76.31	$--$	75.97	$--$
PPIs	67.44	63.95	67.44	70.93	70.93	70.93	70.93	70.93
CATH1	$--$	98.48	**98.87**	$--$	96.91	$--$	96.91	$--$
CATH2	$--$	75.78	76.31	$--$	76.31	$--$	76.31	$--$

The unit of an accuracy value is %.

$--$: can not be finished in one day or the feature values are infinite.

trace outperforms that of all the graph complexity based methods, the coefficients from the Ihara zeta function (CIZF) method, and all the graph kernel methods for comparisons. The SPGK kernel, the random walk graph kernel, VNTD, FV1, FV2 and CIZF cannot finish the computation on the dataset, since they generate burdensome computation. The accuracies of the other proposed complexity traces outperform or are competitive to those of these alternative methods. The runtime of our proposed complexity traces outperforms or is competitive to that of all the alternative methods.

The graphs in the MUTAG dataset are of similar sizes, but correspond to very different structures. On this dataset, the entropy complexity trace with the von Neumann entropy achieves the highest accuracy of our proposed complexity traces. The accuracy of the complexity trace outperforms that of all the alternative methods. The accuracies of the other proposed complexity traces outperform or are competitive to those of these alternative methods. The runtime of our proposed complexity traces outperforms or is competitive to that of all the alternative methods.

The graphs in the NCI1 and NCI109 datasets are of similar sizes, but exhibit very different structures. The entropy complexity trace with the von Neumann entropy achieves the highest accuracies of our proposed complexity traces on the two datasets respectively.

Table 4.4: Performance comparisons of graph kernel methods

Datasets	WL	random walk	SPGK	graphlet count
MUTAG	84.57	86.17	87.23	84.04
NCI1	**73.00**	−−	70.61	67.71
NCI109	**73.28**	−−	70.93	67.32
ENZYMES	38.50	25.33	31.16	34.00
D&D	75.63	−−	−−	77.33
PPIs	73.25	53.48	67.44	**82.55**
CATH1	98.17	−−	98.73	98.73
CATH2	73.15	−−	75.26	74.73

The unit of an accuracy value is %.

−−: can not be finished in one day.

The accuracies of the complexity trace on the two datasets outperform those of all the graph complexity based methods, the CIZF method and most of the graph kernel methods. The accuracies are lower than those of the WL kernel and the SPGK kernel. The VNTD and the random walk kernel were too computationally burdensome to apply on these datasets. The accuracies of the other proposed complexity traces outperform or are competitive to those of these alternative methods exclude the WL kernel. The runtime of our proposed complexity traces outperforms or is competitive to that of all the alternative methods.

The graphs in the ENZYMES dataset are of variable sizes. On this dataset, the entropy complexity trace with the von Neumann entropy achieves the highest accuracy of our proposed complexity traces. The accuracy of the complexity trace outperforms that of all the graph complexity based methods, the CIZF method and the graph kernel methods. The accuracies of the other proposed complexity traces outperform or are competitive to those of all the alternative methods. The runtime of our proposed complexity traces outperforms or is competitive to that of all the alternative methods.

The graphs in the PPIs dataset are of variable sizes. On this dataset, the entropy difference complexity trace with the entropy associated with the information functionals

f^{V_2} or f^{P_2} achieves the highest accuracy of our proposed complexity traces. The accuracy of the complexity trace outperforms that of all these alternative methods excluding the graphlet count graph kernel. The accuracies of the other complexity traces outperform or are competitive to those of other alternative methods. The runtime of our proposed complexity traces outperforms or is competitive to that of all these alternative methods.

The graphs in the CATH1 dataset are of variable sizes. On this dataset, the entropy complexity trace with the Shannon entropy associated with the steady state random walk achieves the highest accuracy of our proposed complexity traces. The accuracy of the complexity trace outperforms that of all these alternative methods. The VNTD and the random walk kernel were too computationally burdensome to apply on the dataset. The accuracies of the other complexity traces outperform or are competitive to those of all the alternative methods. The runtime of our proposed complexity traces outperforms or is competitive to that of all these alternative methods.

The graphs in the CATH2 dataset are of variable sizes. On this dataset, the entropy complexity traces with the von Neumann entropy and the entropy associate with the information functional f^{P_2} achieve the highest accuracies of our proposed complexity traces. The accuracies of the complexity traces outperform those of all these alternative methods. The VNTD and the random walk kernel were too computationally burdensome to apply on the dataset. The accuracies of the other complexity traces outperform those of all the alternative methods. The runtime of our proposed complexity traces outperforms or is competitive to that of all these alternative methods.

Discussions on Experimental Results: On the whole, our proposed complexity traces can easily scale up to large graphs with even thousands of vertices. The accuracies of our proposed complexity traces outperform most of the alternative methods on most datasets used in the experiments. Only on the NCI1 and NCI109 datasets, the accuracies of the WL kernel and the SPGK kernel are competitive to or better than our methods. This is caused by the large proportion of disjoint graphs in the two datasets. From Table 4.1,

Table 4.5: Runtime of proposed complexity traces

Datasets	ECTS	ECTV	ECTFV	ECTFP	EDCTS	EDCTV	EDCTFV	EDCTFP
MUTAG	1"	1"	1"	1"	1"	1"	1"	1"
NCI1	4"	4"	40"	40"	4"	4"	40"	40"
NCI109	4"	4"	40"	40"	4"	4"	40"	40"
ENZYMES	1"	1"	4"	4"	1"	1"	4"	4"
D&D	42"	44"	$23'29$"	$23'29$"	42"	44"	$23'29$"	$23'29$"
PPIs	1"	1"	4"	4"	1"	1"	4"	4"
CATH1	5"	5"	50"	50"	5"	5"	50"	50"
CATH2	2"	2"	25"	25"	2"	2"	25"	25"

Table 4.6: Runtime comparisons of graph complexity based methods and the CIZF

Datasets	VNTD	VNGE	SGE	FV1	FV2	FP1	FP2	CIZF
MUTAG	$19'21$"	1"	1"	1"	1"	1"	1"	1"
NCI1	$> 1day$	1"	1"	4"	4"	4"	4"	26"
NCI109	$> 1day$	1"	1"	4"	4"	4"	4"	26"
ENZYMES	$4h37$"	1"	1"	1"	1"	1"	1"	1"
D&D	$> 1day$	4"	3"	$--$	$1'22$"	$--$	$1'22$"	$--$
PPIs	$52'27$"	1"	1"	1"	1"	1"	1"	55"
CATH1	$> 1day$	1"	1"	$--$	8"	$--$	8"	$--$
CATH2	$> 1day$	1"	1"	$--$	3"	$--$	3"	$--$

$--$: the feature values are infinite.

there are 14.11% and 14.73% of graphs are disjoint for the three datasets. According to the definitions of our proposed complexity traces, the centroid vertex is identified by computing the minimum variance of its shortest path lengths, and the complexity trace for a graph is constructed on the centroid expansion subgraphs from the centroid vertex. Since there are no connections (e.g., infinite path lengths) between some vertices in the disjoint graphs, the identification of the centroid vertex in each disjoint graph is unstable. Moreover, some vertices of a disjoint graph are not included in the centroid expansion subgraphs. For a graph having disjoint structure, our complexity traces can only utilize the biggest component of the graph. In other word, the other smaller components of a

Table 4.7: Runtime comparisons of graph kernel methods

Datasets	WL	random walk	SPGK	graphlet count
MUTAG	1”	8”	1”	3”
NCI1	$2'27$”	$> 1days$	9”	$1'35$”
NCI109	$2'27$”	$> 1days$	9”	$1'35$”
ENZYMES	20”	$9'52$”	4”	33”
D&D	$11'$	$> 1days$	$> 1day$	$21'51$”
PPIs	8”	14”	7”	$3'20$”
CATH1	$2'41$”	$> 1day$	$6'$	$19'9$”
CATH2	12”	$> 1day$	$2'55$”	$7'42$”

disjoint graph will be discarded. However, even under such a disadvantageous situations, our proposed complexity traces can still outperform or be competitive to most of the alternative methods except the WL and SPGK kernels on the two datasets. For the MUTAG, ENZYMES, D&D, PPIs, CATH1 and CATH2 datasets each of which has no or minor disjoint graphs, our complexity traces can outperform the WL and SPGK kernels. Furthermore, on all datasets used in our experiments, the runtime of our proposed complexity traces outperforms or is competitive to that of the graph complexity based methods and the CIZF method, and also outperforms that of all the graph kernel methods.

Furthermore, the accuracies of proposed complexity traces with different entropies are obviously higher than those of the original entropies. This reveals that our proposed depth-based representations of graphs can capture richer structures of graphs. Compared to the single value based complexity measure methods of graphs (i.e., VNTD, VNGE, SGE, FV1, FV2, FP1 and FP2), our complexity traces of graphs can reflect a high dimensional and comprehensive complexity information of graphs. Through the experimental evaluations, we also observe that the performance of the proposed depth-based complexity traces of graphs are related to that of different entropies. The higher accuracy of an entropy can achieve, the higher accuracy of the entropy or entropy difference complexity trace associated with the entropy can achieve. The performance of the von Neumann entropy is better than the other entropies. The accuracies of the entropy complexity traces of graphs are higher than that of the entropy difference complexity traces of graphs, but the runtime of them is the same.

4.4 Conclusion

In this chapter, we have shown how to construct depth-based complexity traces for a graph. Our methods were motivated by the ideas of the entropy based graph complexity measure and the depth-based representations of graphs. We have identified a centroid vertex by computing the minimum variance of its shortest path lengths, and thus obtained a family of expansion subgraphs with increasing layer. The complexity traces of a graph have been constructed by measuring how the graph entropies or the entropy differences vary with the subgraphs of increasing layer. Experiments on graph datasets abstracted from bioinformatics and image data demonstrate the effectiveness and efficiency of our complexity traces in graph classification.

Chapter 5

A Depth-Based Matching Kernel for Unattributed Graphs

In this chapter, we present our third contribution. We develop a novel unattributed graph kernel by matching the depth-based substructures in graphs. We theoretically show the relationship of the depth-based graph kernel and the all subgraph kernel, and then we explain the reasons for the effectiveness of the new graph kernel. The depth-based matching kernel significantly overcomes the shortcoming of neglecting structural correspondence that arises in the all subgraph kernel (or the R-convolution kernels). We explore our depth-based matching kernel on several graph datasets abstracted from computer vision databases. The experimental results demonstrate that our new kernel can easily outperform the existing state-of-the-art graph kernels in terms of the classification accuracy.

Chapter outline

The remainder of this chapter is organized as follows. Section 5.1 presents the definition of h-layer depth-based representation around each vertex for a graph. Section 5.2 presents the definition of the new graph matching kernel. Section 5.3 provides the experimental evaluation. Finally, Section 5.4 concludes our work.

5.1 h-layer Depth-based Representations

In this section, we show how to compute an h-layer depth-based representation around each vertex of a graph. We commence by generalizing the depth-based complexity trace around the centroid vertex developed in Chapter 4.

For an undirected graph $G(V, E)$ and its shortest path matrix S_G, let N_v^K be defined as $N_v^K = \{u \in V \mid S_G(v, u) \leq K\}$, where $S_G(v, u)$ is the shortest path length between vertices v and u. For $G(V, E)$, the K-layer expansion subgraph $\mathcal{G}_v^K(\mathcal{V}_v^K; \mathcal{E}_v^K)$ around vertex v is

$$\begin{cases} \mathcal{V}_v^K &= \{u \in N_v^K\}; \\ \mathcal{E}_v^K &= \{u, v \in N_v^K,\ (u, v) \in E\}. \end{cases} \tag{5.1}$$

Assume L_{max} is the greatest length of the shortest paths from v to the remaining vertices of $G(V, E)$. If $L_v \geq L_{max}$, the L_v-layer expansion subgraph is $G(V, E)$ itself.

Definition 5.1 (h-layer depth-based representation) For a graph $G(V, E)$ and a vertex $v \in V$, the h-layer depth-based representation around the vertex v of $G(V, E)$ is a h-dimensional vector

$$D_G^h(v) = [H_S(\mathcal{G}_v^1), \cdots, H_S(\mathcal{G}_v^K), \cdots, H_S(\mathcal{G}_v^h)]^T \tag{5.2}$$

where K ($K \leq h \leq L_v$) is the length of the shortest path from the vertex v to the remaining vertices in $G(V, E)$, $\mathcal{G}_v^K(\mathcal{V}_v^K; \mathcal{E}_v^K)$ is the K-layer expansion subgraph of $G(V, E)$ around the vertex v, and $H_S(\mathcal{G}_v^K)$ is the random walk Shannon entropy of $\mathcal{G}_v^K$ and is defined in Eq.(3.9). □

For a graph $G(V, E)$ and a vertex $v \in V$, computing the h-layer depth-based representation $D_G^h(v)$ of $G(V, E)$ around v requires time complexity $O(h|V|^2)$. This follows the definition in Eq.(5.1). For the graph $G(V, E)$, computing the shortest path matrix using the Dijkstra's algorithm requires time complexity $O(|V|^2)$. Computing the Shannon entropies of the h K-layer expansion subgraphs, which are derived from v, requires time complexity $O(h|V|^2)$. Hence, the whole time complexity is $O(h|V|^2)$. This indicates that the h-layer depth-based representation around a vertex of a graph can be efficiently computed. Key to this efficiency is that the Shannon entropy on an expansion subgraph only requires time complexity $O(|V|^2)$. By contrast, in [17] the intrinsic complexity measure of an expansion subgraph for measuring the depth-based complexity requires time complexity $O(|V|^5)$.

The h-layer depth-based representation $D_G^h(v)$ characterizes the depth-based complexity of $G(V, E)$ with regard to the vertex v in a h dimensional feature space. It captures the rich depth-based complexity characteristics of substructures around the vertex v in terms of the entropies of the K-layer expansion subgraphs with K increasing from 1 to h. In contrast, the existing graph kernels in the literatures [75, 76, 108] tend to compute similarities on global subgraphs of limited sizes and can only capture restricted characteristics of graphs.

5.2 Depth-based Graph Matching Kernel

We describe how the depth-based representations for graphs can be used for graph matching. We measure graph similarities based on the proposed graph matching method and thus define a novel depth-based graph kernel.

5.2.1 Depth-based Graph Matching

We develop a matching method similar to that introduced in [19] for point set matching, which computes an affinity matrix in terms of the distances between points. In our work, for a vertex v of $G(V, E)$, we treat the h-layer depth-based representations $D_G^h(v)$ as the point coordinate associated with v. Let $G_p(V_p, E_p)$ and $G_q(V_q, E_q)$ be a pair of graphs, we use the Euclidean distance between the depth-based representations $D_{G_p}^h(v_i)$ and $D_{G_q}^h(u_j)$ as the distance measure of pairwise vertices $v_i \in V_p$ and $u_j \in V_q$. The affinity matrix element $R(i,j)$ is defined as

$$\begin{aligned} R(i,j) &= \sqrt{[D^h_{G_p}(v_i) - D^h_{G_q}(u_j)]^T[D^h_{G_p}(v_i) - D^h_{G_q}(u_j)]} \\ &= \parallel D^h_{G_p}(v_i) - D^h_{G_q}(u_j) \parallel_2. \end{aligned} \tag{5.3}$$

where R is a $|V_p| \times |V_q|$ matrix. The element $R(i,j)$ represents the dissimilarity between the vertex v_i in $G_p(V_p, E_p)$ and the vertex u_j in $G_q(V_q, E_q)$. The rows of $R(i,j)$ index the vertices of $G_p(V_p, E_p)$, and the columns index the vertices of $G_q(V_q, E_q)$.

If $R(i,j)$ is the smallest element both in row i and in column j, there should be a one-to-one correspondence between the vertex v_i of $G_p(V_p, E_p)$ and the vertex u_j of $G_q(V_q, E_q)$. We record the state of correspondence using the correspondence matrix $C \in \{0,1\}^{|V_p|\times|V_q|}$ satisfying

$$C(i,j) = \begin{cases} 1 & \text{if } R(i,j) \text{ is the smallest element} \\ & \text{both in row } i \text{ and in column } j; \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

Eq. (5.4) implies that if $C(i,j) = 1$, the vertices v_i and v_j are matched.

Note that, in row i or column j there may be two or more elements satisfying Eq.(5.4). In other words, for a pair of graphs, a vertex from a graph may have two or more than two matched vertices from the other graph. To assign a vertex one matched vertex at most, we update the matrix C by employing the Hungarian method that is widely used for solving the assignment problem (e.g., the bipartite graph matching problem) in polynomial time [99]. Here the matrix $C \in \{0,1\}^{|V_p|\times|V_q|}$ can be seen as the incidence matrix of a bipartite graph $G_{pq}(V_p, V_q, E_{pq})$, where V_p and V_q are the two sets of partition parts and E_{pq} is the edge set. By performing the Hungarian algorithm on the incidence matrix $C \in \{0,1\}^{|V_p|\times|V_q|}$ (i.e., the correspondence matrix of G_p and G_q) of the bipartite graph G_{pq}, we assign each vertex from G_p or G_q at most one matched vertex from the other graph G_q or G_p. Note finally that, directly performing the Hungarian algorithm on the matrix R can also assign each vertex from G_p or G_q an unique matched vertex. However, it cannot guarantee that each identified element is the smallest both in the row and column in R. This is because some vertices will not have matched vertices.

For a pair of graphs $G_p(V_p, E_p)$ and $G_q(V_q, E_q)$ ($|V_p| = |V_q| = n$). Computing the correspondence matrix $C \in \{0,1\}^{|V_p|\times|V_q|}$ (i.e., the final correspondence matrix updated by the Hungarian algorithm) requires time complexity $O(hn^3)$. This follows the definition in Section 5.2.1. For G_p, computing its n h-layer depth-based representations derived from each of its vertices requires time complexity $O(hn^3)$, and it is the same for G_q. Computing each element of the affinity matrix R requires time complexity $O(h)$, and hence computing the whole affinity matrix R requires time complexity $O(hn^2)$. The computation of the correspondence matrix C need to enumerate all the n^2 pairs of elements in R and thus requires time complexity $O(n^2)$. The Hungarian algorithm on the matrix C requires time complexity $O(n^3)$. As a result, the whole time complexity is $O(hn^3)$.

5.2.2 A Depth-based Graph Kernel

Based on the graph matching strategy described in Section 5.2.1, we define a depth-based graph kernel function.

Definition 5.2 (The depth-based graph kernel) Consider a pair of graphs $G_p(V_p, E_p)$ and $G_q(V_q, E_q)$. Based on the definitions in Eq.(5.2), Eq.(5.3) and Eq.(5.4), and the Hungarian algorithm, we compute the correspondence matrix C. The depth-based graph kernel $k_{DB}^{(h)}$ using the h-layer depth-based representations of the graphs is

$$k_{DB}^{(h)}(G_p, G_q) = \sum_{i=1}^{|V_p|}\sum_{j=1}^{|V_q|} C(i,j). \tag{5.5}$$

which counts the number of matched vertex pairs between $G_p(V_p, E_p)$ and $G_q(V_q, E_q)$. □

Lemma 5.1 *The depth-based graph kernel $k_{DB}^{(h)}$ is positive definite (**pd**).* □

Proof Intuitively, the proposed depth-based graph kernel is **pd** because it counts pairs of matched vertices (i.e., the number of smallest isomorphic subgraphs). More formally, let the base kernel k be a function counting pairs of matched vertices in the pair of graphs $G_p(V_p, E_p)$ and $G_q(V_q, E_q)$

$$k(G_p, G_q) = k_{DB}^{(h)}(G_p, G_q) = \sum_{v_i \in V_p} \sum_{u_j \in V_q} \delta(v_i, u_j). \tag{5.6}$$

where

$$\delta(v_i, u_j) = \begin{cases} 1 & \text{if } C(i,j) = 1; \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

where δ is the Dirac kernel, that is, it is 1 if the arguments are equal, and 0 otherwise (i.e., it is 1 if a pair of vertices are matched and 0 otherwise). Hence the proposed kernel function $k_{DB}^{(h)}$ is the sum of several positive definite Dirac kernels, and is thus **pd**. ■

Time Complexity: The depth-based graph kernel $k_{DB}^{(h)}$ on a pair of graphs $G_p(V_p, E_p)$ and $G_q(V_q, E_q)$ $(|V_p| = |V_q| = n)$ requires time complexity $O(hn^3)$. This follows the definition in Section 5.2.1. For the pair of graphs $G_p(V_p, E_p)$ and $G_q(V_q, E_q)$, computing their correspondence matrix C in terms of h-layer depth-based representations requires time complexity $O(hn^3)$, and counting the number of matched vertex pairs from the matrix C needs to enumerate all the n^2 pairs of elements in C. Hence, the whole time complexity of the depth-based graph kernel $k_{DB}^{(h)}$ is $O(hn^3)$.

The time Complexity analysis indicates that our depth-based graph kernel $k_{DB}^{(h)}$ between pairs of graphs can be computed in polynomial time. Key to this efficiency is that the required h-layer depth-based representations and the corresponding matching can be efficiently computed.

The depth-based graph kernel is related to the depth-based representation defined in Chapter 4. However, there are two significant differences. First, the depth-based representation in Chapter 4 is computed by measuring the complexities of subgraphs from the centroid vertex, which is identified by evaluating the minimum shortest path length variance to the remaining vertices. By contrast, we compute the h-layer depth-based representation for each vertex as a point coordinate. Second,the depth-based representation from the centroid vertex can be seen as an embedding vector. Embedding a graph into a vector tends to approximate the structural correlations in a low dimensional space, and thus leads to information loss. By contrast, the depth-based graph kernel computed by matching the h-layer depth-based representation characterizes graphs in a high dimensional space and thus better preserves graph structure.

5.2.3 The Depth-based Graph Kernel on N Graphs

For a graph dataset having N graphs, the kernel matrix of the proposed kernel can be computed using the following computational steps: 1) For each graph, compute the h-layer depth-based representation around each vertex. 2) Compute the corresponding matrix for each pair of graphs based on their depth-based representations. 3) Compute the kernel value for each pair of graphs based on their corresponding matrix.

For N graphs each of which has n vertices, the depth-based graph kernel on all pairs of these graphs requires time complexity $O(Nhn^3 + N^2hn^2)$. This is because the step 1 requires time complexity $O(Nhn^3)$. The step 2 requires time complexity $O(N^2hn^2 + N^2n^2)$, and step 3 requires time complexity $O(N^2n^2)$. As a result, the whole time complexity is $O(Nhn^3 + N^2hn^2)$.

5.2.4 Linkage to the All Subgraph Kernel

The depth-based graph kernel can be defined in another manner that elucidates its relation to the all subgraph kernel. Let $G_p(V_p, E_p)$ and $G_q(V_q, E_q)$ be two graphs. The all subgraph kernel [108] is defined as

$$k_{sub}(G_p, G_q) = \sum_{S_p \sqsubseteq G_p} \sum_{S_q \sqsubseteq G_q} k_{iso}(S_p, S_q), \tag{5.8}$$

where

$$k_{iso}(S_p, S_q) = \begin{cases} 1 & \text{if } S_p \simeq S_q, \\ & \text{i.e., S}_\text{p} \text{ and s}_\text{q} \text{ are isomorphic} \\ 0 & \text{otherwise.} \end{cases} \tag{5.9}$$

Theorem 5.1 *The depth-based graph kernel k_{DB}^h is equivalent to the all subgraph kernel.*

Proof. We prove this theorem by revealing the relationship between the depth-based representation and the subgraph isomorphism. Based on Eq.(5.2) and Definition 5.1, the h-layer depth-based representations around a vertex v of $G_p(V_p, E_p)$ and a vertex u of $G_q(V_q, E_q)$ are

$$D_{G_p}^h(v) = \{H_S(\mathcal{G}_{p;v}^1), \cdots, H_S(\mathcal{G}_{p;v}^K), \cdots, H_S(\mathcal{G}_{p;v}^h)\},$$

and

$$D_{G_q}^h(u) = \{H_S(\mathcal{G}_{q;u}^1), \cdots, H_S(\mathcal{G}_{q;u}^K), \cdots, H_S(\mathcal{G}_{q;u}^h)\}.$$

Clearly, the expansion subgraphs $\mathcal{G}_{p;v}^h$ and $\mathcal{G}_{q;u}^h$ encapsulate other smaller subgraphs $\mathcal{G}_{p;u}^K$ and $\mathcal{G}_{q;u}^K$ respectively. According to the observations in [17], each depth-based representation can be seen as a casual trajectory leading v to $\mathcal{G}_{p;v}^h$ or u to $\mathcal{G}_{q;u}^h$. Based on the depth-based matching defined in Section 5.2.1, if the vertices v and u are matched the two trajectories are close together on a principle space. The h-layer expansion subgraphs $\mathcal{G}_{p;v}^h$ and $\mathcal{G}_{q;u}^h$ can be seen as approximate isomorphism, i.e., $\mathcal{G}_{p;v}^h \simeq \mathcal{G}_{q;u}^h$. As a result, the proposed kernel k_{DB}^h can be re-written as

$$k_{DB}^h(G_p, G_q) = \sum_{S_p \sqsubseteq G_p} \sum_{S_q \sqsubseteq G_q} k_{iso}(S_p, S_q), \tag{5.10}$$

where

$$k_{iso}(S_p, S_q) = \begin{cases} 1 & \text{if } S_p = \mathcal{G}_{p;v}^h \text{ and } S_q = \mathcal{G}_{q;u}^h, \\ & \text{and } v \text{ and } u \text{ are matched,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.11}$$

Eqs.(5.8) and (5.10) indicate that both the kernels k_{sub} and k_{DB}^h need to identify all pairs of isomorphic subgraphs. For k_{sub} and k_{DB}^h, each isomorphic subgraph pair adds an unit value to the kernel value. Thus, both the depth-based kernel and the all subgraph kernel count the number of isomorphic subgraph pairs and are thus equivalent. ■

Theorem 5.1 and its proof highlight the following difference between the depth-based graph kernel and the all subgraph kernel. **a)** For the depth-based graph kernel, only the subgraphs around a pair of matched vertices having a maximum topology distance K are evaluated with respect to isomorphism. While for the all subgraph kernel, any pair of subgraphs are evaluated for identifying the isomorphism. **b)** The depth-based graph kernel overcomes the NP-hard problems of measuring all possible pairs of subgraphs arising in the all subgraph kernel. **c)** For the depth-based graph kernel, any pair of isomorphism subgraphs are identified by a pair of matched vertices. Hence there is a locational correspondence between the isomorphism subgraphs with respect to global graphs. On the other hand, for the all subgraph kernel, a pair of subgraphs having no location correspondence may also be seen as isomorphism.

5.3 Experimental Results

In this section, we empirically compare our new depth-based matching kernel with several alternative state-of-the-art graph kernels on several standard graph datasets. Unlike other chapters (i.e., Chapter 3, 4 and 6) that mainly use graph datasets abstracted from bioinformatics, in this chapter, we use graph datasets abstracted from computer vision. The reasons of using computer vision datasets are twofold. First, many computer vision applications require the correspondence information between pairwise feature points that are abstracted from images or 3D shapes, for the objective of similarity measure [102]. For an instance, one has two graphs abstracted from two digital images both containing the same object, based on different viewpoints. Here, each vertex represents a feature point. Identifying the correspondence information between pairwise vertices or substructures from the identical region is our concern, and can provide us an elegant way of reflecting precise similarity between the images or shapes (e.g., the similarity measure from the entropic matching that is developed by Escolano et al. in [102]). Second, graph matching has been proven a powerful tool for identifying the correspondence information between pairwise vertices (i.e., feature points) in computer vision applications. As a result, the

Table 5.1: Classification accuracy (in % ± standard error) comparisons using C-SVM.

Datasets	DB	WLSK	SPGK	GCGK
COIL5	**74.22** ± .41	33.16 ± 1.01	69.97 ± .92	67.00 ± .55
BAR31	**69.40** ± .56	58.53 ± .53	55.73 ± .44	22.96 ± .65
BSPHERE31	**56.43** ± .69	42.10 ± .68	48.20 ± .76	17.10 ± .60
GEOD31	**42.83** ± .50	38.20 ± .68	38.40 ± .65	15.30 ± .68

new matching kernel can easily indicate its main advantage of identifying the correspondence information, on computer vision datasets. The advantage is unavailable for most existing graph kernels from R-convolution. Finally, in this section, we also evaluate the computational efficiency of our new kernel.

5.3.1 Datasets

We explore the performance of our kernel on computer vision datasets. These datasets are COIL5, BAR31, BSPHERE31 and GEOD31. For the COIL5 dataset, each graph represents an image. For the BAR31, BSPHERE31 and GEOD31 datasets, each graph represents a 3D shape. Note that, each graph in these dataset is unattributed, for some kernels which can accommodate the attributed graphs we use the vertex degree as the label of a vertex. Details of these datasets are described as follows.

COIL5: We establish a COIL5 dataset from the COIL database. The COIL image database consists of images of 100 3D objects. We use the images for the first five objects. For each object we employ 72 images captured from different viewpoints. For each image we first extract corner points using the Harris detector, and then establish Delaunay graphs based on the corner points as vertices. As a result, in the dataset there are 5 classes of graphs, and each class has 72 testing graphs. The number of maximum, minimum and average vertices for the dataset are 241, 72 and 144.90 respectively.

BAR31, BSPHERE31 and GEOD31: The SHREC 3D Shape database consists of 15 classes and 20 individuals per class, that is 300 shapes [100]. This is a usual benchmark in 3D shape recognition. From the SHREC 3D Shape database, we establish three

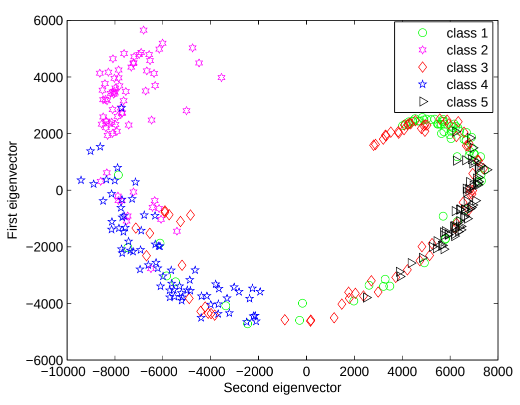
Table 5.2: Rand index for K-means method.

Datasets	DB	WLSK	SPGK	GCGK
COIL5	**0.4436**	0.3503	0.4124	0.4119
BAR31	**0.2319**	0.2047	0.1734	0.1638
BSPHERE31	**0.1615**	0.1304	0.1582	0.1210
GEOD31	**0.1502**	0.1136	0.1142	0.1002

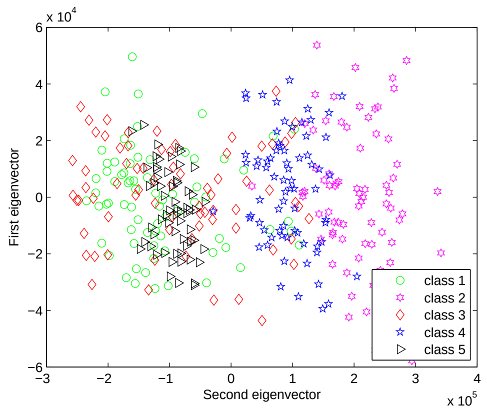
graph datasets named BAR31, BSPHERE31 and GEOD31 datasets through three mapping functions. These functions are a) ERG barycenter: distance from the center of mass/barycenter, b) ERG bsphere: distance from the center of the sphere that circumscribes the object, and c) ERG integral geodesic: the average of the geodesic distances to the all other points. The number of maximum, minimum and average vertices for the three datasets are a) 220, 41 and 95.42 (for BAR31), b) 227, 43 and 99.83 (for BSPHERE31), and c) 380, 29 and 57.42 (for GEOD31), respectively.

5.3.2 Experiments on Graph Datasets

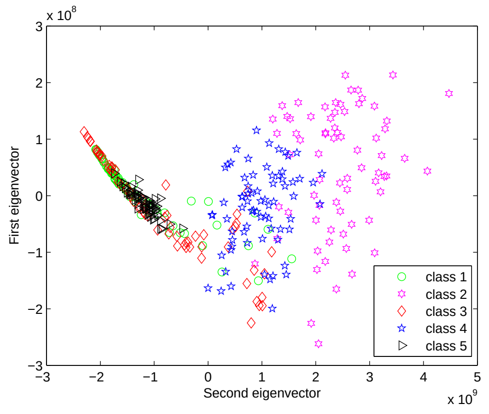
Experimental Setup: a) First, we evaluate the performance of our depth-based matching kernel ($h = 10$) (DB) on classification problems. We also compare our kernel with several alternative state-of-the-art graph kernels. These graph kernels include 1) the Weisfeiler-Lehman subtree kernel (WLSK) [7], 2) the shortest path graph kernel (SPGK) [39], and 3) the graphlet count graph kernel (GCGK) [82]. For our DB kernel, we set h as 10. For the WLSK kernel, we set the highest dimension (i.e., the highest height of subtrees) of the Weisfeiler-Lehman isomorphism as 10. For the GCGK graph kernel, we set the size of a graphlet as 3. For each kernel, we compute the kernel matrix on each graph dataset. We perform 10-fold cross-validation using the C-Support Vector Machine (C-SVM) Classification to compute the classification accuracies, using LIBSVM [83]. We use nine folds for training and one for testing. All the C-SVMs were performed along with their parameters optimized on each dataset. We report the average classification accuracies and standard errors for each kernel in Table.5.1.



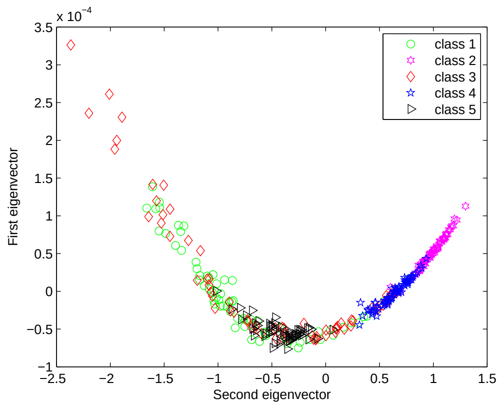
(a) For DB kernel.



(b) For WLSK kernel.



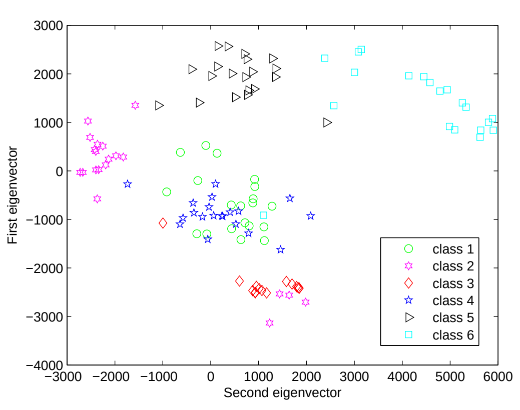
(c) For SPGK kernel



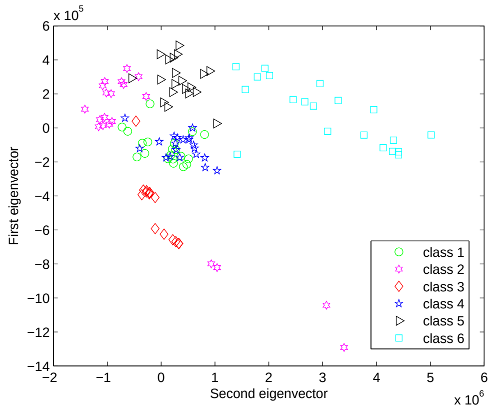
(d) For GCGK kernel

Figure 5.1: Embedding from kPCA for the COIL5 dataset.

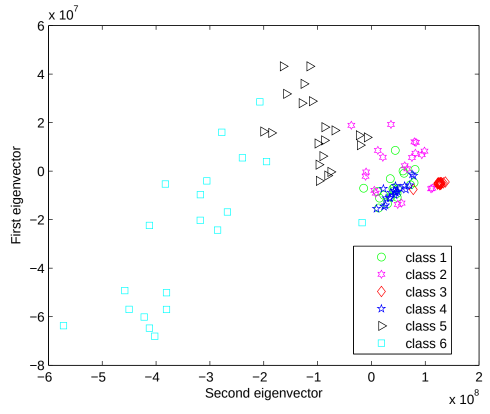
b) Second, we evaluate the performance of different kernels on clustering problems. We commence by performing the kernel Principle Component Analysis (kPCA) on the kernel matrix to embed graphs into a 2-dimensional principal space. We visualize the embedding results of each kernel using the first two principal components. The embedding results on the COIL5, BAR31, BSPHERE31 and GEOD31 datasets are shown in Fig.5.1, Fig.5.2, Fig.5.3 and Fig.5.4, respectively. Moreover, we also show the Euclidean distance matrices of these embedding results in the principal space for each kernel. The Euclidean distance matrices on the COIL5, BAR31, BSPHERE31 and GEOD31 datasets are shown in Fig.5.5, Fig.5.6, Fig.5.7 and Fig.5.8, respectively. Note that, for the BAR31,



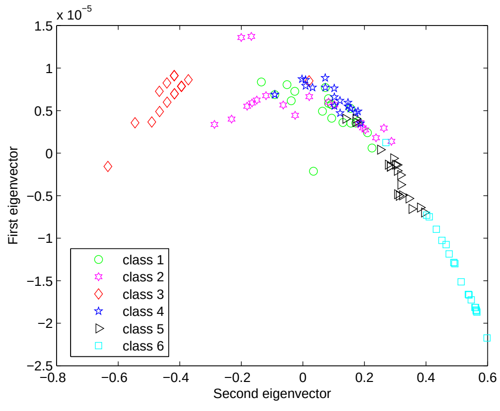
(a) For DB kernel.



(b) For WLSK kernel



(c) For SPGK kernel

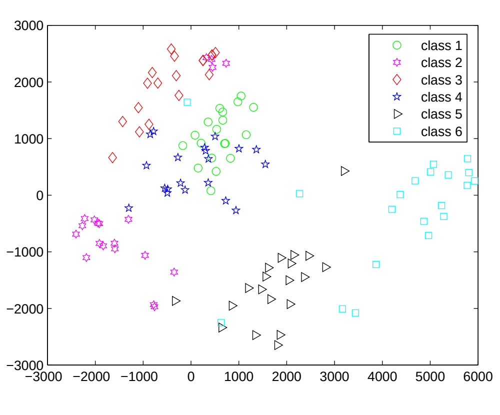


(d) For GCGK kernel

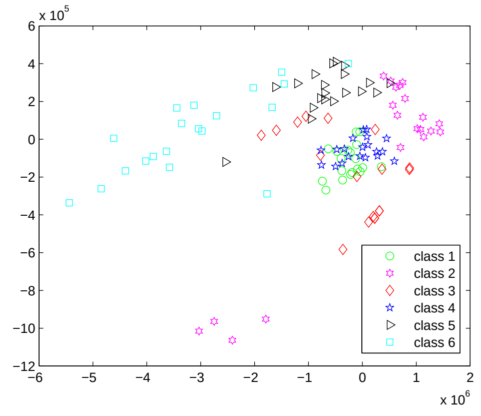
Figure 5.2: Embedding from kPCA for the BAR31 dataset.

BSPHERE31 and GEOD31 datasets, we only visualize the embedding points and Euclidean distance matrices of the first six classes of graphs. Finally, to place our analysis of graph clustering on a more quantitative footing, for each kernel we apply the K-means method to all the kernel embeddings. We calculate the Rand Index for the resulting clusters. The Rand indicating each kernel is listed in Table.5.2.

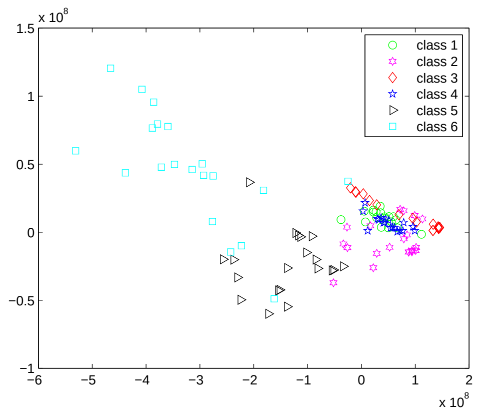
c) Third, we investigate whether the different kernels can learn the structural variation within a graph class. For each kernel and its embedding result on the COIL5 dataset, we mark the points of the first ten graphs in the embedding principal space. These graphs are abstracted from the first ten images of the first object in the COIL image database. Since



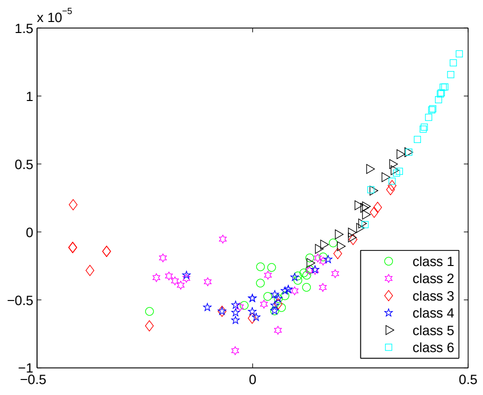
(a) For DB kernel.



(b) For WLSK kernel.



(c) For SPGK kernel.

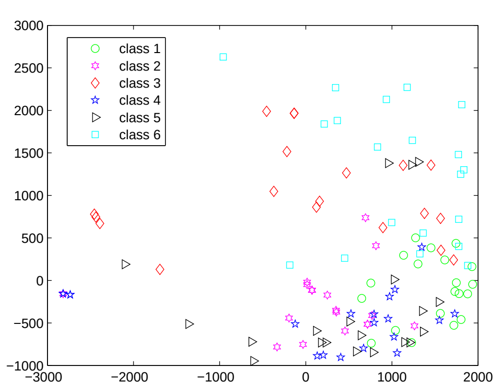


(d) For GCGK kernel.

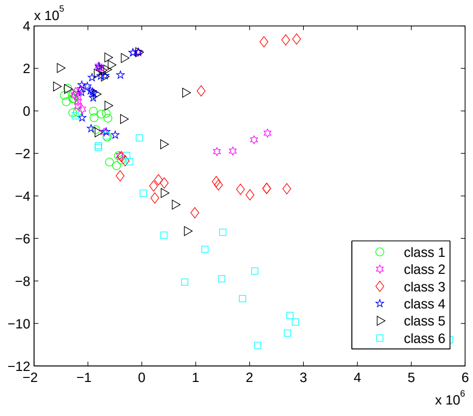
Figure 5.3: Embedding from kPCA for the BSPHERE31 dataset.

the ten images are captured from different viewpoints spaced at intervals of 5° around the object, the ten points are marked with the view number which corresponds to the camera angle. The results are shown in Fig.5.9.

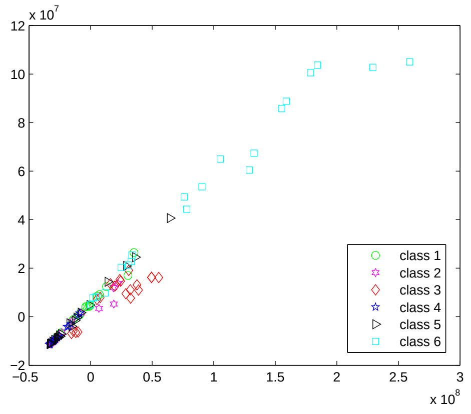
Experimental Results and Discussions: In terms of the classification and clustering accuracies, we make three observations. a) First, we observe that the accuracies of our DB kernel are the greatest for all datasets. The performance of our DB kernel obviously exceeds that of all other kernels. The reason for its effectiveness is that the required depth-based representations of graphs used for DB kernel establishes a substructure location correspondence through the depth-based matching. In contrast, the other alternative



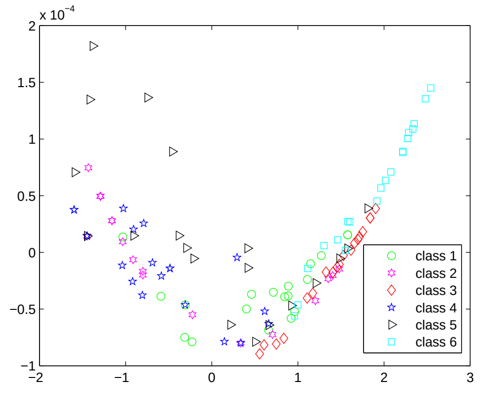
(a) For DB kernel.



(b) For WLSK kernel.



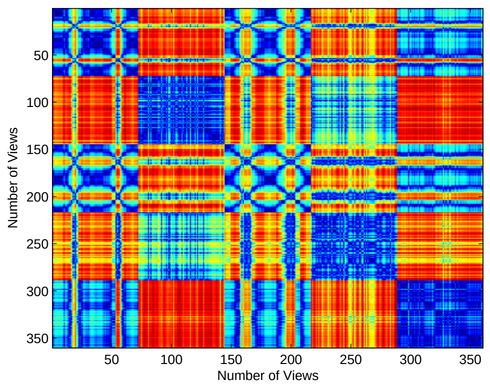
(c) For SPGK kernel.



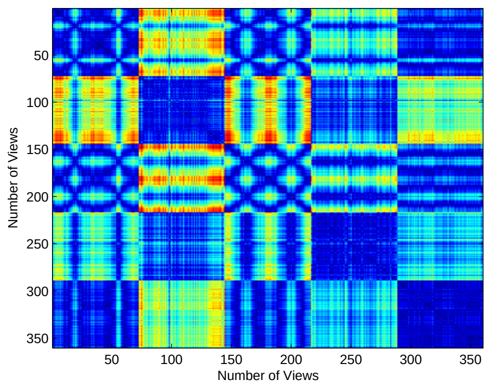
(d) For GCGK kernel.

Figure 5.4: Embedding from kPCA for the GEOD31 dataset.

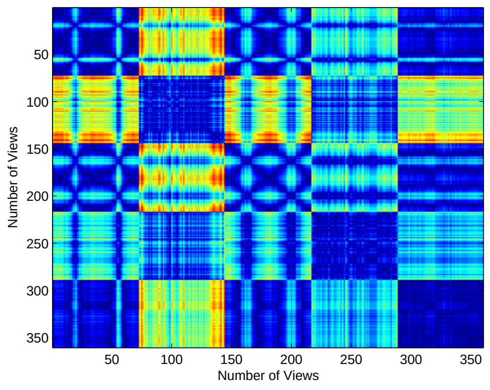
kernels cannot reflect the substructure location correspondence. Furthermore, we observe that the accuracy of WLSK kernel is obviously lower than other kernels on the COIL5 dataset. The reason for this is that the WLSK kernel requires a tree-index (TI) method that identifies a subtrees for each vertex by augmenting the label of the vertex using the labels of its neighbouring vertices. For the COIL5 dataset, each graph is a Delaunay graph. Through these graphs, we observe that the degree of each vertex (i.e., the label of each vertex) is very similar. As a result, the WLSK kernel can only identify few distinguishable subtrees. This reveals that the WLSK kernel based on the TI method is not suitable for Delaunay graphs.



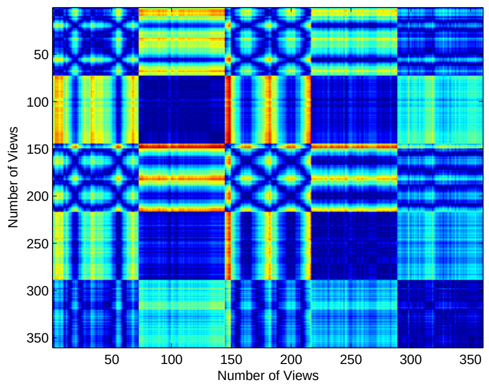
(a) For DB kernel.



(b) For WLSK kernel



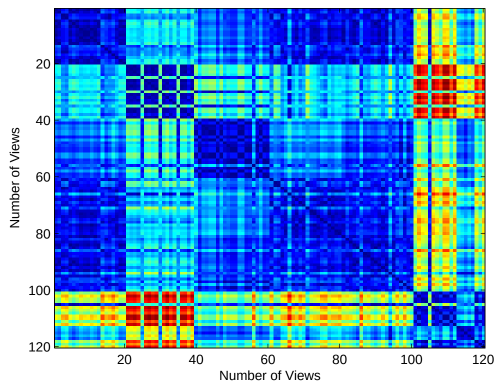
(c) For SPGK kernel



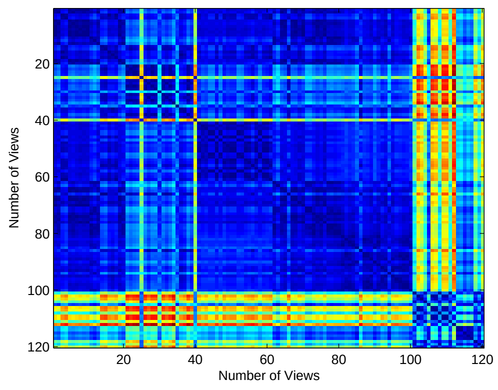
(d) For GCGK kernel

Figure 5.5: Distance matrix for the COIL5 dataset.

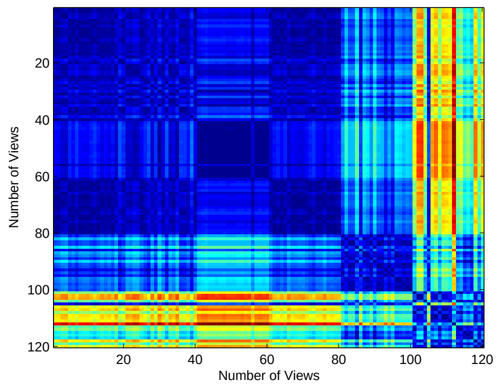
b) Second, in terms of the embedding results, it is clear that our DB kernel produces the best clusters. The different classes are separated better than other kernels on any dataset. Note that, for the COIL5 dataset the 72 images for each object are taken from different viewing directions spaced at intervals of 5° around the object. Hence, the embedded graphs for each class are expected to form a circular trajectory rather than a cluster in the feature space. In the light of this observation, our method shows a greater representational power in terms of giving a more trajectory-like embedding than the alternative methods. Moreover, in terms of the Euclidean distance matrices we observe that all the kernels have a well-defined block structure on the COIL5 dataset. However, it is clear that



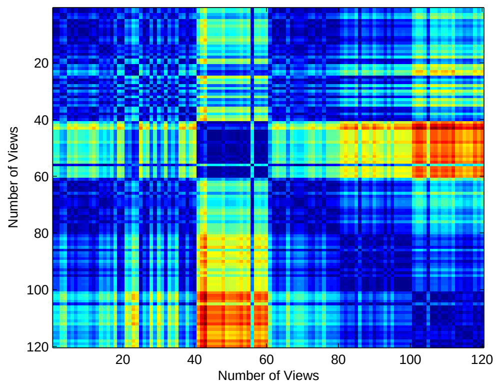
(a) For DB kernel.



(b) For WLSK kernel



(c) For SPGK kernel



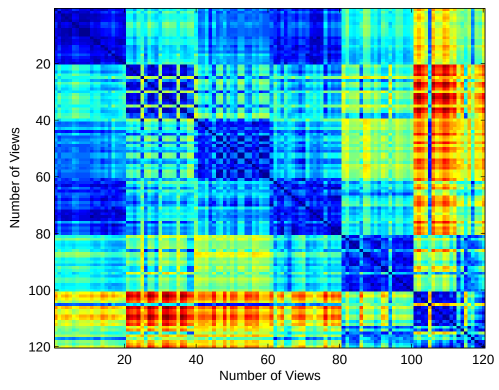
(d) For GCGK kernel

Figure 5.6: Distance matrix for the BAR31 dataset.

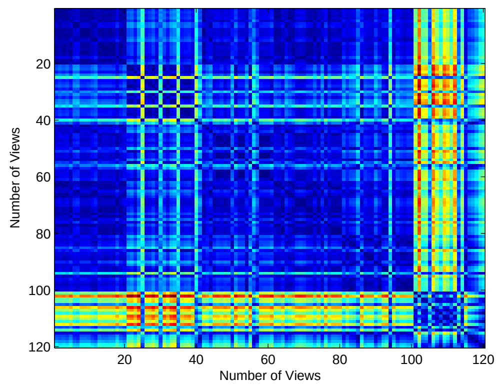
our DB kernel yields a stronger structure than other kernels on the BAR31, BSPHERE31 and GEOD31 datasets. Finally, Table 5.2 indicates that our DB kernel outperforms all the alternative kernels for all the object classes studied on any dataset. These observations verify that our DB kernel has better ability to distinguish different classes of graphs.

c) Finally, through Fig.5.9 we observe that our DB kernel produces a clear trajectory and the neighboring images in the sequence are close together in the embedding principal space. In contrast, all other kernels hardly result in a trajectory. This verifies that our DB kernel learns the structural variation within a graph class better than all other kernels.

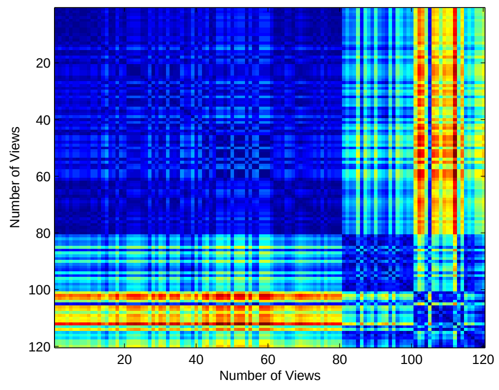
Comparisons with Increasing h: To take our study one step further, we evaluate the per-



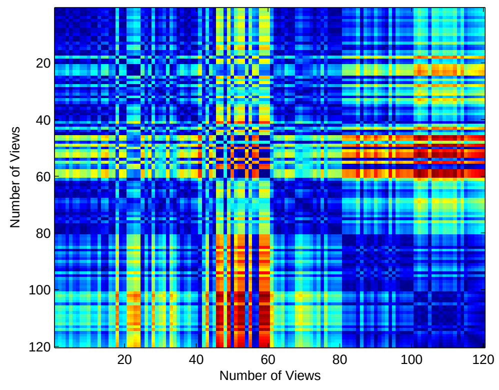
(a) For DB kernel.



(b) For WLSK kernel.



(c) For SPGK kernel.

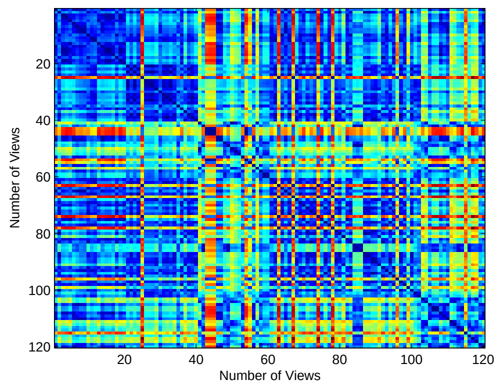


(d) For GCGK kernel.

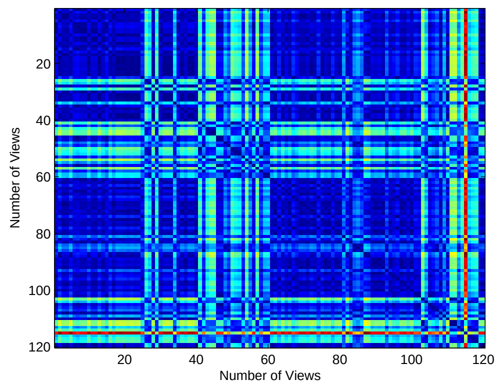
Figure 5.7: Distance matrix for the BSPHERE31 dataset.

formance of our DB graph kernel on graph datasets with increasing h. Here, we evaluate how the classification accuracies vary with increasing h (i.e., $h = 1, 2, \dots, 10$). We report the results in Fig.5.10, in which the x-axis gives the varying of h, and the y-axis gives the classification accuracies of our DB kernel. The lines of different colours represent the results on different datasets. Moreover, we also report the stand error in Table.5.3.

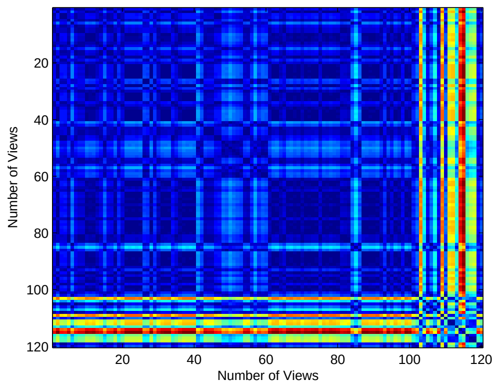
We make two observations through the experimental results. First, we observe that the classification accuracies tend to become greater with increasing h. This is because the greater the h, the higher dimensional depth-based complexity information of our kernel can be captured. Moreover, the accuracies tend to be stable when $h > 6$. Second, we



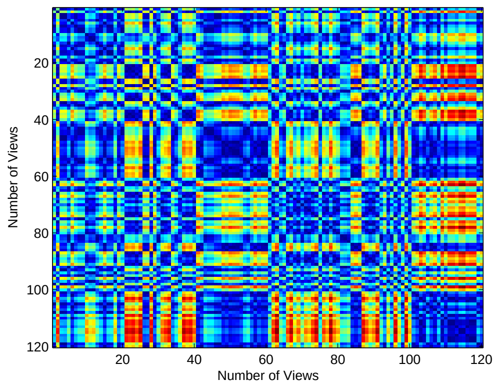
(a) For DB kernel.



(b) For WLSK kernel.



(c) For SPGK kernel.



(d) For GCGK kernel.

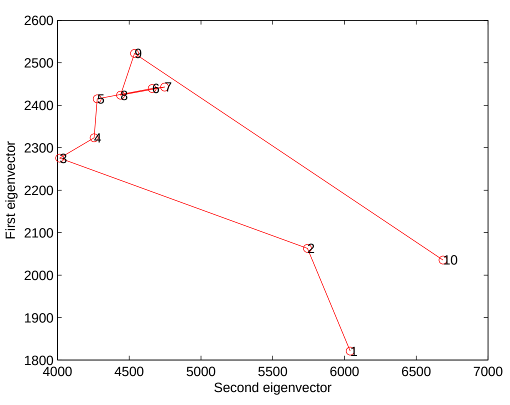
Figure 5.8: Distance matrix for the GEOD31 dataset.

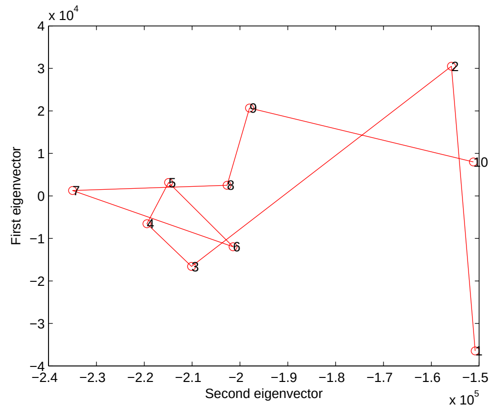
observe that the stand errors tend to become smaller with increasing h.

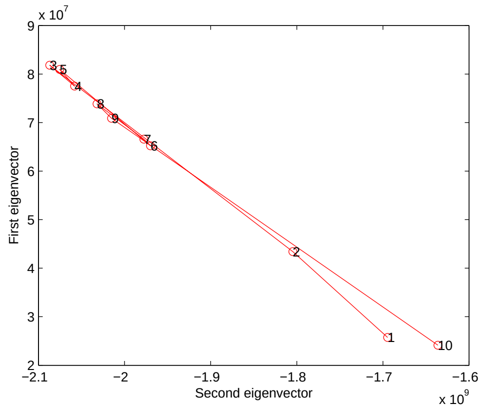
5.3.3 Computational Evaluation

Finally, we evaluate the computational efficiency (i.e., the CPU runtime) of our DB graph kernel, and reveal the relationship between the computational overheads and the structural complexity or number of the associated graphs.

Experimental setup: We evaluate the computational efficiency on randomly generated graphs with respect three parameters: the graph size n, the layer h of the depth-based rep-


(a) For DB kernel.


(b) For WLSK kernel.


(c) For SPGK kernel.

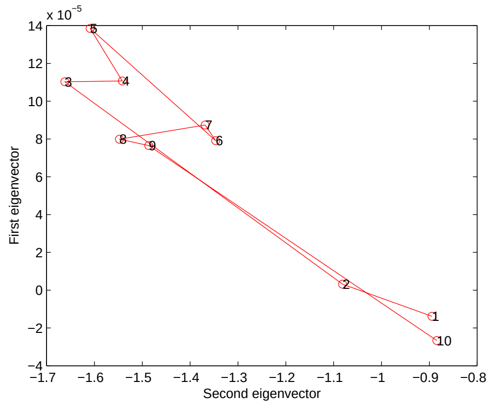

(d) For GCGK kernel.

Figure 5.9: Eigenprojection of graphs.

resentations of graphs, and the graph dataset size N. We vary $n = \{100, 200, \ldots, 2000\}$, $h = \{1, 2, \ldots, 50\}$ and $N = \{1, 2, \ldots, 500\}$, separately. a) For the experiments with graph size n, we generate 20 pairs of graphs with increasing number of vertices. We report the runtime for computing the kernel values between pairwise graphs ($h = 10$). b) For the experiments with the larger parameter h, we generate a pair of graphs each of which has 200 vertices. We report the runtime for computing the kernel values of the pair of graphs as a function of h. c) For the graph dataset size N, we generate 500 graph datasets with an increasing number of test graphs. In each dataset, one graph has 200 vertices. We report the runtime for computing the kernel matrices for each graph dataset

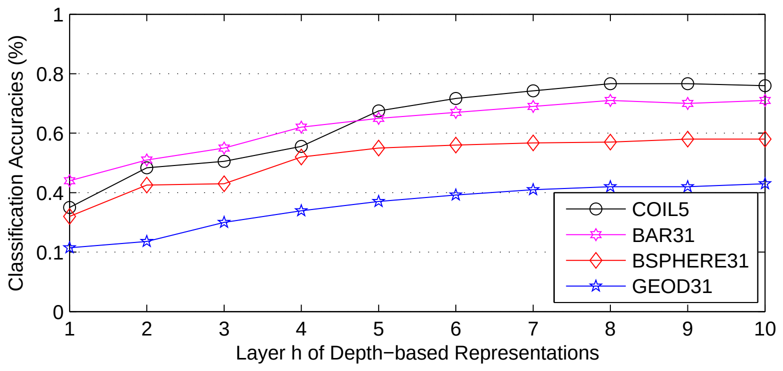


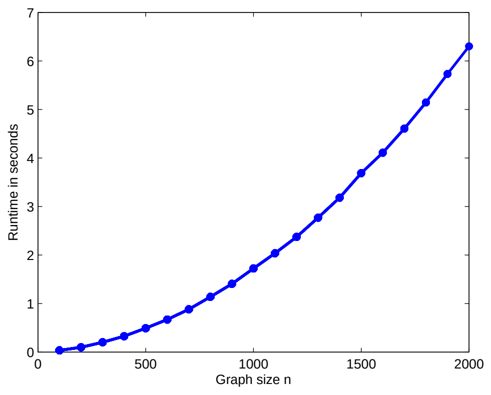
Figure 5.10: The accuracy with different h layer.

Table 5.3: The standard error with different h layers.

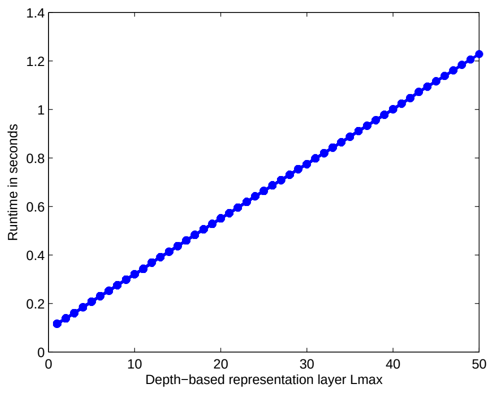
Datasets	COIL5	BAR31	BSPHERE31	GEOD31
h=1	±0.55	±0.66	±1.19	±0.67
h=2	±0.54	±0.61	±1.07	±0.69
h=3	±0.50	±0.64	±0.94	±0.68
h=4	±0.47	±0.63	±0.90	±0.63
h=5	±0.45	±0.66	±0.85	±0.64
h=6	±0.46	±0.57	±0.83	±0.62
h=7	±0.47	±0.58	±0.73	±0.60
h=8	±0.44	±0.57	±0.70	±0.53
h=9	±0.41	±0.56	±0.71	±0.54
h=10	±0.41	±0.56	±0.69	±0.50

($h = 10$). The CPU runtime is reported in Fig.5.11, as operated in Matlab R2011b on a 2.5GHz Intel 2-Core processor (i.e., i5-3210m).

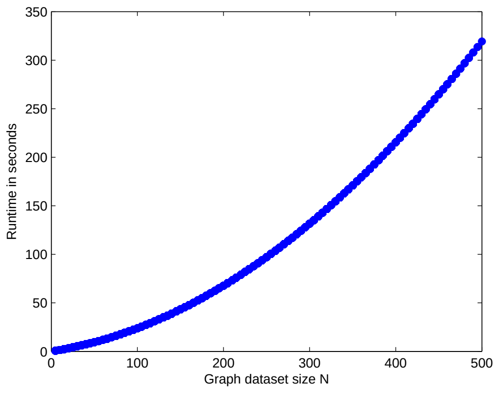
Experimental results: Figs.5.11 (a), (b) and (c) show the results for the DB kernel when varying the parameters n, h and N, respectively. We observe that the runtime scales quadratically with n, linearly with h, and quadratically with N. These results verify that our kernel can be computed in polynomial time.



(a) For graph size n



(b) For layer h



(c) For data size N

Figure 5.11: Runtime evaluations.

5.4 Conclusion

In this paper, we have described how to construct a depth-based graph kernel in terms of matching graphs based on the depth-based representations. The depth-based representations for graphs capture a high dimensional depth-based complexity information of graphs. Furthermore, our matching strategy incorporates structural correspondence into the kernel. The experimental results demonstrate the effectiveness and efficiency of our kernel.

Chapter 6

A Hypergraph Kernel from Subtree Isomorphism Tests

In this chapter, we present our forth contribution to the design of a hypergraph kernel based on substructure isomorphism tests. We commence by defining a new directed Weisfeiler-Lehman (WL) isomorphism test for directed graphs. Then, we define a new kernel for a pair of hypergraphs by counting the number of pairwise isomorphic substructures identified by the new WL algorithm from their directed line graphs. We show that our hypergraph kernel limits the tottering problem that arises in the existing walk and subtree based (hyper)graph kernels. We explore our new hypergraph kernel on either graph or hypergraph based datasets abstracted from bioinformatics and computer vision databases. We demonstrate the effectiveness and efficiency of our new hypergraph kernel.

Chapter outline

The remainder of this chapter is organized as follows. Section 6.1 presents the definition of the directed line graph for a hypergraph. Section 6.2 presents the definition of the new hypergraph kernel using the new developed directed Weisfeiler-Lehman isomorphism test. Section 6.3 provides the experimental evaluation. Finally, Section 6.4 concludes our work.

6.1 Directed Line Graphs

To develop a WL isomorphism test for a pair of hypergraphs, we need to establish a directed line graph for a hypergraph [12]. The directed line graph of a hypergraph is a dual representation in which each hyperedge is represented by a new vertex. The reasons for using this representation are twofold. First, a pairwise-order representations for hypergraphs may enable the graph based isomorphism test to be applied to hypergraphs. Second, the directed line graph will not lead to the order ambiguities that result from the straightforward expansion or a clique based graph representation of a hypergraph. For a hypergraph $HG(V_H, E_H)$, the directed line graph $G_D(V_D, \overrightarrow{E}_D)$ can be established using **Algorithm 2**.

6.1.1 Definitions and Notations

Note that, for step 1 there are potential multiple edges between two vertices in $GH(V_G, E_G)$ if the two vertices are encompassed by more than one common hyperedge in $HG(V_H, E_H)$. Suppose there are p hyperedges encompassing two vertices in $HG(V_H, E_H)$. The p hyperedges induce p edges separately between the two vertices in $GH(V_G, E_G)$. For step 3, it is important to stress that unlike the edge set E of an undirect graph $G(V, E)$, $\overrightarrow{E}_D$ is a set of directed edges of the directed graph $G_D(V_D, \overrightarrow{E}_D)$. The adjacency matrix T_H of $G_D(V_D, \overrightarrow{E}_D)$ is the Perron-Frobenius operator of the original hypergraph. For the (i, j)th entry of T_H, $T_H(i, j)$ is 1 if there is a simple edge directed from the vertex i to the vertex j in the directed line graph, and otherwise it is 0. Unlike the adjacency matrix of an undirected graph, the Perron-Frobenius operator for a hypergraph is not a symmetric matrix. This is because the constraint in Eq.(6.2) arises in the construction of directed edges. Specifically, any two directed edges induced by the same hyperedge in the original hypergraph are not allowed to establish a directed edge in the directed line graph.

An example of transforming a hypergraph into a directed line graph has been shown in

Algorithm 2: Establishing a directed line graph

Input: A hypergraph $HG(V_H, E_H)$. **Output:** A directed line graph $G_D(V_D, \overrightarrow{E}_D)$ for HG.

1. Establish the clique expansion graph $GH(V_G, E_G)$ for $HG(V_H, E_H)$ by connecting each pair of vertices in a hyperedge $e_i \in E_H$, the vertex and edge sets are

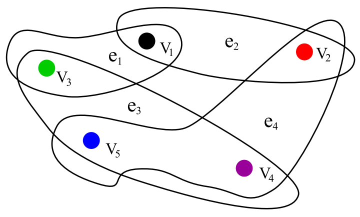
$$\begin{cases} V_G \;\; = V; \\ E_G = \{\{u, v\} \subset e_i \mid e_i \in E_H\}. \end{cases} \tag{6.1}$$

2. Establish the associated symmetric digraph $DGH(V_G, E_d)$ by replacing each edge of $GH(V_G, E_G)$ by a directed edge pair in which the two directed edges are inverse to each other.

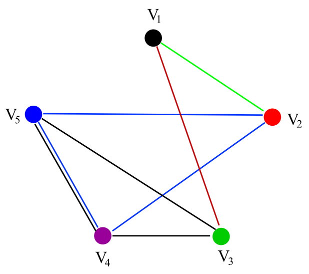
3. Establish the directed line graph $G_D(V_D, \overrightarrow{E}_D)$ of $HG(V_H, E_H)$ based on $DGH(V_G, E_d)$. The vertex set V_D and directed edge set $\overrightarrow{E}_D$ of G_D are

$$\begin{cases} V_D = E_d; \\ \overrightarrow{E}_D = \{(u, v)_i, (v, w)_j \in E_d \times E_d \mid i \neq j\}. \end{cases} \tag{6.2}$$

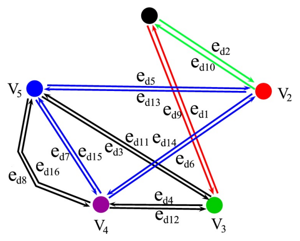
   where the subscripts i and j denote the indices of the hyperedges from which the directed edges (u, v) and (v, w) are induced respectively.



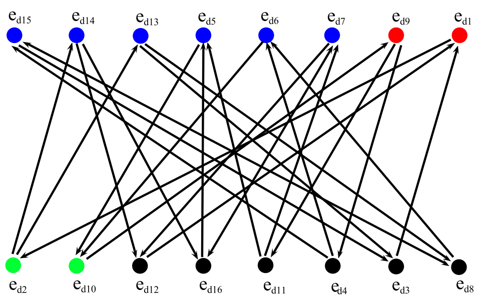
(a) A hypergraph



(b) Clique.



(c) Di-clique.



(d) Directed line graph.

Figure 6.1: An example of transformation a hypergraph into a directed line graph.

Fig.6.1. For the example hypergraph $HG(V_H, E_H)$ shown in Fig.6.1(a), the clique graph $GH(V_G, E_G)$ is shown in Fig.6.1(b). In $GH(V_G, E_G)$, the edges belonging to the common clique are indicated by the same colour while the different cliques are coloured differently. Furthermore, there are two different edges between v_4 and v_5, and these edges are induced by the hyperedge e_3 and e_4 of $HG(V_H, E_H)$, respectively. The associated symmetric digraph $DGH(V_G, E_d)$ of $GH(V_G, E_G)$ is shown in Fig.6.1(c), and the resulting directed line graph $G_D(V_D, \overrightarrow{E}_D)$ from $DGH(V_G, E_d)$ is shown in Fig.6.1(d).

The transformation from the hypergraph $HG(V_H, E_H)$ into the directed line graph $G_D(V_D, \overrightarrow{E}_D)$ requires time complexity $O(|V_D|^2)$. This is because the construction of the adjacency matrix of $G_D(V_D, \overrightarrow{E}_D)$ relies on visiting all the $|V_D|$ ($|V_D| = |E_d|$) edges in $DGH(V_G, E_d)$ and establishing all the $|V_D|^2$ entries in the incidence matrix of G_D.

6.1.2 Theoretical Properties

The directed line graph and its Perron-Frobenius operator have several interesting properties as follows.

1) First, comparing to the (hyper)graph adjacency or Laplacian matrix, the Perron-Frobenius operator spans a higher dimensional feature space where it may expose richer (hyper)graph characteristics. This property is a result of the fact that the cardinality of the vertex set for the directed line graph is much greater than, or at least equal to, that of the original (hyper)graph (see details in [12]). Hence, the adjacency matrix (i.e., the Perron-Frobenius operator) of the directed line graph is described in a high dimensional space than the original (hyper)graph.

2) Second, the directed line graph represents a (hyper)graph in a complete manner such that it naturally avoids the information loss arising in the spectral truncation [68] or the clique graph approximation [63]. This property is due to the constraint in Eq.(6.2) the connecting arc pair induced by the same hyperedge in the original hypergraph cannot establish a directed edge in the directed line graph. In other words, such a directed line graph can distinguish different edges derived from the same hyperedge. This property is illustrated in Fig.6.1(d). On the other hand, the clique expansion graph $GH(V_G, E_G)$ from the original hypergraph $HG(V_H, E_H)$ only records adjacency relationships between pairwise vertices of the hypergraph and cannot distinguish whether or not two edges are derived from the same hyperedge. This property is illustrated in Fig.6.1(b). Hence, for two different hypergraphs (e.g., the hypergraphs shown in Figs.6.2(a) and (b)) they may have the same clique expansion graph, and then the same resulted adjacency and Laplacian matrices defined from the clique expansion graph. But, the directed line graph defined in Eq.(6.2) may still produce total different structures for the two hypergraphs.

3) Finally, the directed line graph of a hypergraph is a backtrackless structure of the hypergraph. This property is due to the constraints imposed on a hypergraphs by the Perron-Frobenius operator (i.e., the adjacency matrix of the directed line graph for a hypergraph

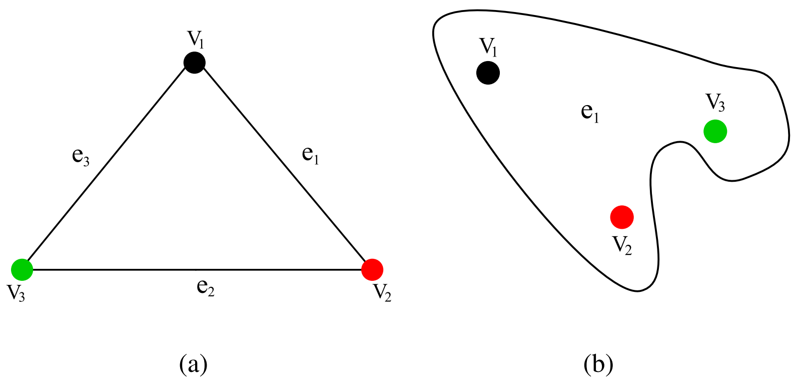


Figure 6.2: Hypergraph examples.

is not summetric). For the line graph of a hypergraph, a bidirected edge between a pair of vertices may be not included.

These properties indicate that the directed line graph and its Perron-Frobenius operator offers us an elegant way for hypergraph isomorphism analysis, which can not only capture precise hypergraph isomorphism information but can also reflect richer characteristics of hypergraphs.

6.2 A Hypergraph Kernel

In this section, we develop a new directed WL isomorphism test from undirected graphs into directed graphs. Finally, we define a new kernel for hypergraphs based on the new isomorphism test between their directed line graphs.

6.2.1 The Directed Weisfeiler-Lehman Isomorphism Test

WL Isomorphism Test on Undirected Graphs: We commence by reviewing the WL isomorphism test between undirected graphs [20], i.e., the 1-dimensional variant WL isomorphism test. The key idea of the WL isomorphism test is to first augment the label of a vertex by the labels of its neighbouring vertices, and compress these augmented labels into new set of labels. Then, the isomorphism between a pair of graphs can be tested by checking the identicalness between two sets of strengthened labels.

Suppose $G(V, E)$ is an undirected graph with vertex set V and undirected edge set $E \subseteq V \times V$. $f : \mathcal{L} \rightarrow \Sigma$ is a function that assigns vertices new labels from an alphabet Σ based on the existing vertex labels. The neighbourhood $\mathcal{N}(v) = \{u|(v, u) \in E\}$ of a vertex $v \in V$ is the set of vertices connected to v by an edge. The WL isomorphism test procedure between any pair of undirected graphs G_p and G_q with an increasing iteration h is as follows:

1. Set $h = 0$. Initialize the vertex labels. For G_p, compute the vertex degree $d_p(v_p)$ of each vertex v_p as the initial label $\mathcal{L}_{p;h}(v_p)$ of v_p. The set of vertex labels for G_p is $L_{p;h} = \{\mathcal{L}_{p;h}(v_p) = d_p(v_p)|v_p \in V_p\}$. Do the same computation for G_q.

2. For each vertex v_p of G_p, sort the labels of its neighbourhood $\mathcal{N}(v_p)$ in ascending order as $\mathcal{L}^h_{\mathcal{N};p}(v_p) = \{\mathcal{L}_{p;h}(u_p)|u_p \in \mathcal{N}_p(v_p)\}$. Do the same computation for G_q.

3. Set $h = h + 1$. For each vertex v_p of G_p, assign a new label as $\mathcal{L}_{p;h}(v_p) = \{\mathcal{L}_{p;h-1}(v_p), \mathcal{L}^{h-1}_{\mathcal{N};p}(v_p)\}$. Do the same computation for G_q.

4. Compress the label $\mathcal{L}_{p;h}(v_p)$ into a new short label for each vertex v_p of G_p, using the function $f : L_{p;h} \rightarrow \Sigma$. Do the same computation for G_p.

5. Check h. Repeat step 2, 3 and 4 until the iteration h achieves an expected value. Or terminate the procedure if the sets of newly generated labels are not identical in G_q and G_q (i.e., G_p and G_q are not isomorphic).

Note that, in step 4 we use the same vertex label function f for both G_p and G_q. This guarantees that all the identical labels of G_p and G_q are mapped into the same number. Furthermore, for each iteration h the initialized or compressed labels $\mathcal{L}_{p;h}(v_p)$ and $\mathcal{L}_{q;h}(v_q)$ correspond to subtrees of height h rooted from v_p and v_q respectively. If $\mathcal{L}_{p;h}(v_p) = \mathcal{L}_{q;h}(v_q)$, the subtrees of height h rooted from v_p and v_q are isomorphic. The undirected WL isomorphism test offers us an elegant way of defining a kernel of undirected graphs by counting the number of pairwise isomorphic subtrees. Unfortunately, straightforwardly measuring the undirected WL isomorphism between hypergraphs tends to be elusive since a hypergraph may exhibit various relational orders (i.e., a hyperedge can encompass an arbitrary number of vertices). The undirected WL isomorphism algorithm cannot distinguish a significant difference between adjacent vertices connected by different order hyperedges. For instance, for the two hypergraphs shown in Fig.6.2(a) and (b), they have the same adjacency matrix and the same degree for each vertex. As a result, the undirected WL isomorphism algorithm will assign all the vertices the same labels, though these vertices may be connected by different order hyperedges.

WL Isomorphism Test on Directed Graphs: To overcome the limitation of the original WL isomorphism test [20] on hypergraphs, we transform a hypergraph into a directed line graph. As we have stated in Section 6.1, the directed line graph can reflect precise topology information of a hypergraph. Furthermore, we generalize the isomorphism algorithm as a new directed WL isomorphism test on directed graphs. Suppose $G_D(V_D, \overrightarrow{E}_D)$ is a directed graph with vertex set V_D and directed edge set $\overrightarrow{E}_D \subseteq V_D \times V_D$, then the structure of this graph can be represented by a $|V_D| \times |V_D|$ adjacency matrix A_D as follows

$$A_D(v_D, u_D) = \begin{cases} 1 & \text{if } (v_D, u_D) \in \overrightarrow{E}_D \\ 0 & \text{otherwise.} \end{cases} \tag{6.3}$$

The in-degree and out-degree of a vertex v_D are

$$d^{in}(v_D) = \sum_{u_D=1}^{|V_D|} A_D(u_D, v_D), \tag{6.4}$$

and

$$d^{out}(v_D) = \sum_{u_D=1}^{|V_D|} A_D(v_D, u_D), \tag{6.5}$$

respectively. The in-neighbourhood of a vertex $v_D \in V_D$ is the set of vertices connecting v_D by a directed edge from the vertices, and is

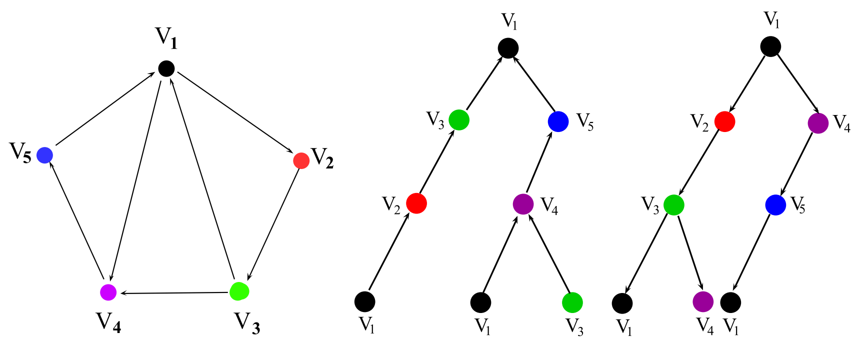
$$\mathcal{N}^{in}(v_D) = \{u_D | (u_D, v_D) \in \overrightarrow{E}_D\}. \tag{6.6}$$

The out-neighbourhood of a vertex $v_D \in V_D$ is the set of vertices connected by a directed edge from v_D, and is

$$\mathcal{N}^{out}(v_D) = \{u_D | (v_D, u_D) \in \overrightarrow{E}_D\}. \tag{6.7}$$

We develop the directed WL isomorphism test based on two steps. The first step is to assign a vertex a new in-label using the in-degree of the vertex and that of its in-neighbourhood. The second step is to assign a vertex a new out-label using the out-degree of the vertex and that of its out-neighbourhood. The directed WL isomorphism test between any pair of directed graphs $G_{D;p}$ and $G_{D;q}$ at an increasing iteration h can be measured using **Algorithm 3**.

Note that, in **Algorithm 3** the initialized or compressed in-label $\mathcal{L}^{in}_{p;h}(v_{D;p})$ usually corresponds an in-subtree (i.e., the root vertex $v_{D;p}$ ***(for the initialized label)***, or a directed subtree having shortest paths from other vertices to the root vertex $v_{D;p}$ ***(for the compressed label)***) of height h rooted at $v_{D;p}$. The initialized or compressed out-label $\mathcal{L}^{out}_{p;h}(v_{D;p})$ usually corresponds an out-subtree (i.e., the rooted vertex $v_{D;p}$ ***(for the initialized label)***, or a directed subtree having shortest paths to other vertices from the root vertex $v_{D;p}$ ***(for the compressed label)***) of height h rooted at $v_{D;p}$. See Fig.6.3 for an illustration of in-subtrees and out-subtrees of a directed graph. For $G_{D;p}$ and $G_{D;q}$, if $\mathcal{L}^{in}_{p;h}(v_{D;p}) = \mathcal{L}^{in}_{q;h}(v_{D;q})$ or $\mathcal{L}^{out}_{p;h}(v_{D;p}) = \mathcal{L}^{out}_{q;h}(v_{D;q})$, the in or out-subtrees of height h rooted from $v_{D;p}$ and $v_{D;q}$ are isomorphic. Furthermore, note finally that, for a directed graph G_D if a label $\mathcal{L}^{in}_{h}(v_D)$ or $\mathcal{L}^{out}_{h}(v_D)$ with an iteration h ($h \geq 0$) corresponds the highest in or out subtree rooted at v_D in the global directed graph G_D, then the label $\mathcal{L}^{in}_{h'}(v_D)$ or $\mathcal{L}^{out}_{h'}(v_D)$ with the iteration h' ($h' > h$) also corresponds the same highest in or out subtree. In other words, for some instances a higher in or out subtree may be a corresponding



(a) Directed graph. (b) In-subtree. (c) Out-subtree.

Figure 6.3: An in-subtree and out-subtree of height 3 rooted at the vertex 1 on a directed graph.

lower in or out subtree itself, and a compressed in or out label may also correspond a root vertex.

For a directed graph $G_D(V_D, \overrightarrow{E}_D)$ which has $|V_D|$ vertices, the directed Weisfeiler-Lehman isomorphism algorithm on G_D requires time complexity $O(H|V_D|^2)$. Here H is the largest number of iteration h for the directed WL isomorphism test. This is because the initialization of in-degree $d^{in}(v_D)$ and out-degree $d^{out}(v_D)$ requires to visit all the $|V_D|^2$ entries in the adjacency matrix of G_D, and requires time complexity $O(|V_D|^2)$. For each iteration, assigning the vertices new in-labels and out-labels needs to visit all the $|V_D|^2$ entries in the adjacency matrix, and compressing the new generated label of a vertex requires time complexity $O(|V_D|)$ (for the worst-case, i.e., all other vertices are neighbourhoods for a vertex). As a result, the whole time complexity is $O(H|V_D|^2)$.

6.2.2 An Isomorphism-Based Hypergraph Kernel

Based on the new directed isomorphism test described in Section 6.2.1, we define a new hypergraph kernel from the directed line graphs.

Definition 6.1 (The Hypergraph Kernel from Directed Line Graphs) For a pair of hy-

Algorithm 3: Measuring the directed WL isomorphism for a pair of directed graphs

1: Initialization.

- Input a pair of directed graphs $G_{D;p}(V_{D;p}, \overrightarrow{E}_{D;p})$ and $G_{D;q}(V_{D;q}, \overrightarrow{E}_{D;q})$.
- Set ***h=0***. Initialize the vertex labels. For a vertex $v_{D;p}$ of $G_{D;p}$, assign the in-degree $d^{in}(v_{D;p})$ as the initial in-label $\mathcal{L}^{in}_{p;h}(v_{D;p})$, and assign the out-degree $d^{out}(v_{D;p})$ as the initial out-label $\mathcal{L}^{out}_{p;h}(v_{D;p})$. The set of vertex in-labels and out-labels for $G_{D;p}$ is $L^{in}_{p;h} = \{\mathcal{L}^{in}_{p;h}(v_{D;p}) = d^{in}_{p}(v_{D;p}) | v_{D;p} \in V_{D;p}\}$ and $L^{out}_{p;h} = \{\mathcal{L}^{out}_{p;h}(v_{D;p}) = d^{out}_{p}(v_{D;p}) | v_{D;p} \in V_{D;p}\}$ respectively. Do the same computation for $G_{D;q}$.

2: Sort the labels of neighbourhoods for each vertex.

- For each vertex $v_{D;p}$ of $G_{D;p}$, sort the labels of its in-neighbourhood $\mathcal{N}^{in}_{p}(v_{D;p})$ and out-neighbourhood $\mathcal{N}^{out}_{p}(v_{D;p})$ in ascending order and concatenate the labels into stuples as $\mathcal{L}^{in;h}_{\mathcal{N};p}(v_{D;p})$ and $\mathcal{L}^{out;h}_{\mathcal{N};p}(v_{D;p})$ respectively. Do the same computation for $G_{D;q}$.

3: Update the label for each vertex.

- Set ***h=h+1***. For each vertex $v_{D;p}$ of $G_{D;p}$, assign a new in-label and a new out-label as tuples $\mathcal{L}^{in}_{p;h}(v_{D;p}) = [\mathcal{L}^{in}_{p;h-1}(v_{D;p}), \mathcal{L}^{in;h-1}_{\mathcal{N};p}(v_{D;p})]^{\top}$ and $\mathcal{L}^{out}_{p;h}(v_{D;p}) = [\mathcal{L}^{out}_{p;h-1}(v_{D;p}), \mathcal{L}^{out;h-1}_{\mathcal{N};p}(v_{D;p})]^{\top}$ respectively. Do the same computation for $G_{D;q}$.

4: Compress the vertex label into a new short label.

- Using the mentioned function $f : \mathcal{L} \rightarrow \Sigma$ (e.g., the Hash function), compress the in-label $\mathcal{L}^{in}_{p;h}(v_{D;p})$ and the out-label $\mathcal{L}^{out}_{p;h}(v_{D;p})$ into new short labels for each vertex $v_{D;p}$ of $G_{D;p}$ as

$$\mathcal{L}^{in}_{p;h}(v_{D;p}) = f(\mathcal{L}^{in}_{p;h}(v_{D;p})), \tag{6.8}$$

and

$$\mathcal{L}^{out}_{p;h}(v_{D;p}) = f(\mathcal{L}^{out}_{p;h}(v_{D;p})), \tag{6.9}$$

respectively. Do the same computation for $G_{D;p}$.

5: Check h and evaluate the isomorphism.

- Check h. Repeat step 2, 3 and 4 until the iteration h achieves an expected value. Or terminate the algorithm if the sets of newly generated in-labels in $G_{D;q}$ and $G_{D;q}$ and the sets of newly generated out-labels in $G_{D;q}$ and $G_{D;q}$ are both not identical (i.e., $G_{D;p}$ and $G_{D;q}$ are not isomorphic).

pergraphs $HG_p(V_{H;p}, E_{H;p})$ and $HG_q(V_{H;q}, E_{H;q})$, we commence by transforming HG_p and HG_q into directed line graphs as $G_{D;p}(V_{D;p}, \overrightarrow{E}_{D;p})$ and $G_{D;q}(V_{D;q}, \overrightarrow{E}_{D;q})$ respectively.

Based on step 1 of **Algorithm 3**, we compute the initialized in-label $\mathcal{L}^{in}_{p;0}(v_{D;p})$ and the initialized out-label $\mathcal{L}^{out}_{p;0}(v_{D;p})$ for each vertex $v_{D;p}$ of $G_{D;p}$, we also compute the initialized in-label $\mathcal{L}^{in}_{q;0}(v_{D;q})$ and the initialized out-label $\mathcal{L}^{out}_{q;0}(v_{D;q})$ for each vertex $v_{D;q}$ of $G_{D;q}$. Furthermore, based on Eq.(6.8) and Eq.(6.9), we compute the compressed in-label $\mathcal{L}^{in}_{p;h}(v_{D;p})$ and the compressed out-label $\mathcal{L}^{out}_{p;h}(v_{D;p})$ for each vertex $v_{D;p}$ of $G_{D;p}$, we also compute the compressed in-label $\mathcal{L}^{in}_{q;h}(v_{D;q})$ and the compressed out-label $\mathcal{L}^{out}_{q;h}(v_{D;q})$ for each vertex $v_{D;q}$ of $G_{D;q}$. The hypergraph kernel between HG_p and HG_q is defined as

$$\begin{aligned} k^{(h)}_{HD}(HG_p, HG_q) &= \textstyle\sum_{h=0}^{H} \sum_{v_p \in V_{D;p}} \sum_{v_q \in V_{D;q}} \\ &[\delta(\mathcal{L}^{in}_{p;h}(v_p), \mathcal{L}^{in}_{q;h}(v_q)) + \delta(\mathcal{L}^{out}_{p;h}(v_p), \mathcal{L}^{out}_{q;h}(v_q))], \end{aligned} \tag{6.10}$$

where H is the largest number of iterations h for the directed WL isomorphism test. Here δ is the Dirac kernel, that is, it is 1 if the arguments are equal and 0 otherwise (i.e., it is 1 if $\mathcal{L}^{in}_{p;h}(v_{D;p}) = \mathcal{L}^{in}_{q;h}(v_{D;q})$ or $\mathcal{L}^{out}_{p;h}(v_{D;p}) = \mathcal{L}^{out}_{q;h}(v_{D;q})$, and 0 otherwise). □

Lemma 6.1 *The kernel $k^{(h)}_{HD}$ is positive definite (**pd**).* □

Proof. Intuitively, the proposed kernel $k^{(h)}_{HD}$ is **pd**, because it counts the number of pairwise isomorphic in-subtrees and out-subtrees of up to height h in two directed line graphs. Furthermore, through **Definition 6.1** the proposed kernel can be seen as the sum of positive definite Dirac kernels.

More formally, for an iteration h of the directed WL algorithm, we define a mapping function $\phi^h(l, HG)$ which counts the number of a vertex label $l_x \in \mathcal{L}$ (identified from the directed WL algorithm) contained by the directed line graph $G_D(V_D, \overrightarrow{E}_D)$ of a hypergraph $HG(V_H, E_H)$. Here, $\mathcal{L} = \{l_1, \ldots, l_x, \ldots, l_{|\mathcal{L}|}\}$ is a label set which contains any possible vertex label for directed line graphs from hypergraphs. For the hypergraph HG, we thus define a label occurrence vector as

$$FV_{GH}^{h} = \{\phi^h(l_1, HG), \ldots, \phi^h(l_x, HG), \ldots, \phi^h(l_{|\mathcal{L}|}, HG)\}^\top. \tag{6.11}$$

At an iteration h for the directed WL algorithm, we define a counting function (i.e., a base counting kernel k_{HDB}^{h}) for a pair of hypergraphs $HG_p(V_{H;p}, E_{H;p})$ and $HG_q(V_{H;q}, E_{H;q})$. The base counting kernel counts the number of identical vertex label pairs (i.e., the number of isomorphic in and out subtree pairs) for the directed line graphs from the hypergraphs, and is defined as

$$\begin{aligned} k_{HDB}^{h}(HG_p, HG_q) &= \phi^h(l_1, HG_p)\phi^h(l_1, HG_q) + \ldots + \phi^h(l_x, HG_p)\phi^h(l_x, HG_q) \\ &\quad + \ldots + \phi^h(l_{|\mathcal{L}|}, HG_p)\phi^h(l_{|\mathcal{L}|}, HG_q) \\ &= \langle FV_{GH_p}^{h}, FV_{GH_q}^{h} \rangle, \end{aligned} \tag{6.12}$$

where

$$FV_{GH_p}^{h} = \{\phi^h(l_1, HG_p), \ldots, \phi^h(l_x, HG_p), \ldots, \phi^h(l_{|\mathcal{L}|}, HG_p)\}^\top,$$

and

$$FV_{GH_q}^{h} = \{\phi^h(l_1, HG_q), \ldots, \phi^h(l_x, HG_q), \ldots, \phi^h(l_{|\mathcal{L}|}, HG_q)\}^\top,$$

are the label occurrence vectors of HG_p and HG_q respectively. As a result, the base counting kernel k_{HDBase}^{h} is an inner product (i.e., a liner kernel) of $FV_{GH_p}^{h}$ and $FV_{GH_q}^{h}$, and is **pd**. Thus, the hypergraph kernel $k_{HD}^{(h)}$ can be re-written as

$$\begin{aligned} k_{HD}^{(h)}(HG_p, HG_q) &= k_{HDB}^{1}(HG_p, HG_q) + \ldots + k_{HDB}^{H}(HG_p, HG_q) \\ &= \sum_{h=0}^{H} k_{HDB}^{h}(HG_p, HG_q), \end{aligned} \tag{6.13}$$

which is the sum of **pd** base counting kernels and is also **pd**. ■

Time Complexity: For a pair of hypergraphs HG_p and HG_q, their directed line graphs are $G_{D;p}(V_{D;p}, \overrightarrow{E}_{D;p})$ and $G_{D;q}(V_{D;q}, \overrightarrow{E}_{D;q})$ ($|V_{D;p}| = |V_{D;q}| = n$) respectively. The proposed hypergraph kernel $k_{HD}^{(h)}$ on HG_p and HG_q requires time complexity $O(Hn^2)$.

This is because transforming HG_p and HG_q into $G_{D;p}$ and $G_{D;q}$ requires time complexity $O(n^2)$. Measuring the directed WL isomorphism test between $G_{D;p}$ and $G_{D;q}$ requires time complexity $O(Hn^2)$. Furthermore, the worst-case of counting the number of pairwise isomorphic in-subtrees and out-subtrees requires time complexity $O(Hn^2)$. Hence, the whole time complexity is $O(Hn^2)$.

In other words, the time complexity of the proposed hypergraph kernel relies on the number of the iteration h and the vertex number of the directed line graph for a hypergraph. Note that, since a hypergraph may have various order relationships among vertices, straightforwardly computing the number of vertices for its directed line graph based on the number of its vertices and hyperedges tends to be elusive. However, based on the definition in [4], if each hyperedge of a hypergraph $HG(V_H, E_H)$ is a 2-order relation between pairwise vertices (i.e., HG is an undirected graph), the number of vertices for $G_D(V_D, \overrightarrow{E}_D)$ is related to the number of hyperedges of $HG(V_H, E_H)$, i.e., $|V_D| = 2|E_H|$. As a result, for the pair of hypergraphs HG_p and HG_q both having m hyperedges, the kernel $k_{HD}^{(h)}$ on HG_p and HG_q requires time complexity $O(Hm^2)$.

Note that, the proposed kernel $k_{HD}^{(h)}$ limits tottering that arises in the rooted hypergraph kernel. This is due to the fact that the directed line graph of a hypergraph is a backtrackless structure which contains directed edges (i.e., uni-directional edges). As a result, many vertices cannot visit a vertex and then immediately return to themselves through the same edge.

6.2.3 Discussions and Related Work

Since a hypergraph is a generalization of an undirected graph and such a graph can also be transformed into a directed line graph, computing the hypergraph kernel $k_{HD}^{(h)}$ between a pair of graphs is just a special case of our kernel. On the other hand, the original undirected WL isomorphism test described in Section 6.2.1 can be directly measured between a pair of undirected graphs. It can be straightforward to establish a graph kernel (e.g. the fast subtree kernel defined in [7]) based on the undirected WL isomorphism test for undirected graphs. However, the proposed hypergraph kernel for undirected graphs through their directed line graphs can capture richer characteristics from the original graphs, because the Perron-Frobenius operator can extract (hyper)graph characteristics in a higher dimensional feature space than that of the original (hyper)graph. Moreover, the directed WL isomorphism test can not only distinguish the directed information residing on the directed edges but also identify more subtrees rooted at each vertex than the original undirected WL isomorphism test (i.e., for a vertex the directed WL method identifies an in-subtree and an out-subtree, by contrast the undirected WL method only identifies one undirected subtree). As a result, our hypergraph kernel can reflect the precise and rich isomorphism information of (hyper)graphs from their directed line graphs.

Furthermore, similar to the random walk based kernels, the subtree kernel from the original WL isomorphism test also suffers from tottering. This is because the subtrees identified by the WL isomorphism may also include several copies of the same pairwise vertices connected by the same edge. Aziz et al. [21] have shown that if each hyperedge of a hypergraph is a 2-order relation (i.e., the hypergraph is a graph), there is no bidirected edge in its directed line graph. In other words, such a directed line graph is a completely backtraceless structure (i.e., any vertex cannot visit other vertices and then immediately return to the vertex itself through the same edge). As a result, for graphs our hypergraph kernel tremendously limits the tottering problem that arises in the existing random walk and subtree based graph kernels [7, 22].

6.3 Experimental Results

We demonstrate the performance of our hypergraph kernel on several (hyper)graph datasets from bioinformatics and computer vision. These datasets include a) the HCOIL5 and HAMOS datasets (for hypergraphs), and b) the MUTAG, PTC(MR), PPIs, NCI1, NCI109, ENZYMES and GatorBait datasets (for graphs). The MUTAG, PTC(MR), PPIs, NCI1, NCI109 and ENZYMES datasets have been introduced in Chapter 3, 4 and 5. The detail information for the HCOIL5, HAMOS and GatorBait datasets is introduced as follows.

HCOIL5: The HCOIL5 dataset is a hypergraph dataset abstracted from the COIL image database. The COIL image database consists of 2D images of 100 3D objects. In our experiments, we select the first five objects. For each object we employ 72 images captured from different viewpoints. For each image we first extract corner points using the Harris detector [93], and then establish hypergraphs using the corner points as vertices. Each vertex is used as the seed of a Voronoi region, which expands radially with a constant speed. The linear collision fronts of the regions delineate the image plane into polygons, and a hyperedge encompassing a number of vertices is constructed using the high-level hypergraph feature method described by Ren et al. [12]. There are 360 graphs which are divided as 5 classes in the HCOIL5 dataset. The number of maximum, minimum and average vertices for the three datasets are all 549, 72 and 202.90 respectively.

HAMOS: The HAMOS dataset is a hypergraph dataset abstracted from the *Air Freight Image Sequences* database of Amos Storkey [101]. In the database, each frame is represented as an image. For each image, we again use the Harris method and the high-level hypergraph feature representation. There are 432 graphs which are divided as 21 classes in the HAMOS dataset. The number of maximum, minimum and average vertices for the three datasets are all 98, 4 and 55.60 respectively.

GatorBait: GatorBait has 100 shapes representing fishes from 30 different classes [103]. We have extracted Delaunay graphs from their shape quantization (Canny algorithm followed by contour decimation). Since the classes are associated to fish genus and not to species, we find high intraclass variability in many cases. Therefore, the database, though having only 100 samples, plays a challenging role in testing graph classification. The number of maximum, minimum and average vertices for the dataset are 545, 239 and 348.70.

Table 6.1: Accuracy comparisons on hyperraph datasets

Datasets	HK	HCIZF	TLS	TNLS
HCOIL5	**41.23%**	–	28.32%	23.21%
HAMOS	**43.67%**	40.26%	33.21%	30.12%

6.3.1 Experiments on Hypergraphs

Experimental setup: We illustrate the performance of our hypergraph kernel (HK) on hypergraph classification problems. The hypergraph datasets for testing are abstracted from two image databases. We also compare our kernel with several alternative state of the art hypergraph based learning methods. These methods include 1) the Ihara coefficients for hypergraphs (HCIZF) [12], 2) the truncated Laplacian spectra (TLS) and truncated normalized Laplacian spectra (TNLS) [63]. For each dataset, we compute the kernel matrix or feature vectors of test hypergraphs using our kernel and the alternative methods respectively. For our kernel, we perform 10-fold cross-validation using the C-Support Vector Machine (C-SVM) Classification associated with the supplied kernel matrix to compute the classification accuracies. We use nine folds for training and one for testing. We set the largest number of iterations h for our kernel as 10. All the C-SVMs were performed along with their parameters optimized on each dataset. For the alternative methods, we perform 10-fold cross-validation using the Support Vector Machine (SVM) Classifier with the Sequential Minimal Optimization (SMO) [97] and the Pearson VII universal kernel (PUK) [104] to compute the classification accuracies associated with the feature vectors. All the SMO-SVMs and their parameters were performed and optimized on a Weka workbench. We repeat the each experiment 10 times for each method. We report the average classification accuracies in Table.6.1.

Experimental Results and Evaluations: From Table.6.1 it is clear that our hypergraph kernel achieves the greatest accuracies over all image datasets. **1)** Our kernel outperforms TLS and TNLS which use spectral information for hypergraphs. The reason for this is that our kernel based on the line graph of a (hyper)graph captures richer (hyper)graph characteristics than the (hyper)graph spectral representations and also avoids the spectral truncation arising in TLS and TNLS. **2)** For the HAMOS dataset where the maximum number of vertices is 98, the accuracy of HCIZF is competitive with that of our kernel. Because HCIZF also relies on directed line graphs, and exploits richer hypergraph characteristics. However, for the HCOIL5 dataset where the maximum number of vertices is 549, HCIZF is intractable for characterizing the hypergraphs. Because the computation of the underlying Ihara cofficients tends to result in infinities even for hypergraphs of moderate sizes. In contrast, our kernel can easily scal to large hypergraphs.

6.3.2 Experiments on Graphs

Experimental Setup: We evaluate the performance of our hypergraph kernel (HK) on graph classification problems. The test graph datasets are abstracted from the bioinformatics databases. We compare our hypergraph kernel with several alternative state-of-the-art graph kernels. These graph kernels for comparison include 1) the WL subtree kernel (WLSK) [7], 2) the shortest path graph kernel (SPGK) [39], 3) the graphlet count graph kernels (GCGK) [82] and 4) the random walk graph kernel (RWGK) [22]. For our kernel, we set the largest number of iterations h for the new directed WL isomorphism as 10. For the WLSK subtree kernel, we set the largest number of iterations h for the original undirected WL isomorphism as 10. For the graphlet count graph kernels, we set the size of a graphlet as 3. Note that, the WLSK kernel is able to accommodate attributed graphs. In our experiments, we use the vertex degree as a vertex label for the WLSK kernel.

For our kernel and the alternative graph kernels, we compute the kernel matrix of each graph dataset. We perform 10-fold cross-validation using the C-SVM Classification, which has been introduced in Section 6.3.1, to compute the classification accuracies associated with the kernel matrices computed using different kernels. We report the average classification accuracies for each kernel method in Table.6.2. Furthermore, we also

Table 6.2: Accuracy (in $\%$ $\pm$ standard error) comparisons on graph datasets

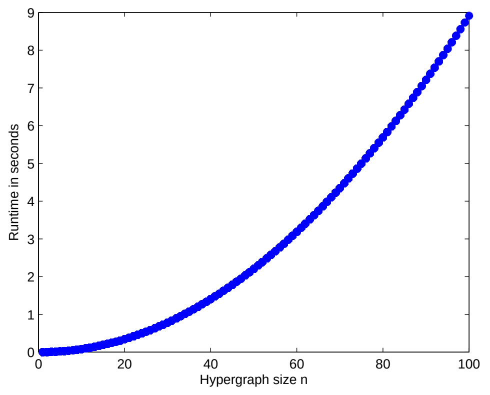
Datasets	MUTAG	PTC(MR)	PPIs	GatorBait	ENZYMES	NCI1	NCI109
HK	**83.33** $\pm$ .53	**56.89** $\pm$ .50	87.50 $\pm$ .83	**11.00** $\pm$ .71	37.58 $\pm$ .50	**80.95** $\pm$ .21	**81.15** $\pm$ .24
WLSK	82.94 $\pm$ .54	56.05 $\pm$ .51	**88.09** $\pm$ .41	10.30 $\pm$ .79	**38.41** $\pm$ .45	80.55 $\pm$ .20	80.79 $\pm$ .21
SPGK	83.16 $\pm$ .69	55.50 $\pm$ .68	61.62 $\pm$ 1.09	7.80 $\pm$ .69	28.55 $\pm$ .42	74.21 $\pm$ .30	73.89 $\pm$ .28
GCGK	81.33 $\pm$ .74	55.17 $\pm$ .36	49.00 $\pm$ 1.57	8.00 $\pm$ .39	24.87 $\pm$ .22	63.72 $\pm$ .12	62.33 $\pm$ .13
RWGK	77.87 $\pm$ .21	54.50 $\pm$ .67	69.70 $\pm$ .30	8.00 $\pm$ .73	22.37 $\pm$ .35	$>$ 1day	$>$ 1day

Table 6.3: Runtime for various kernels.

Datasets	MUTAG	PTC(MR)	PPIs	GatorBait	ENZYMES	NCI1	NCI109
HK	6”	14”	30′	25′47”	1′2”	6′10”	6′10”
WLSK	3”	9”	20”	–	20”	2′30”	2′30”
SPGK	1”	1”	22”	–	4”	16”	16”
GCGK	1”	1”	4”	–	2”	5”	5”
RWGK	14”	2′35”	4′26”	$>$ 1h35′	9′52”	$>$ 1day	$>$ 1day

report runtime of computing the kernel matrices of each kernel in Table.6.3, with the runtime measured under Matlab R2011a running on a 2.5GHz Intel 2-Core processor (i.e., i5-3210m).

Experimental Results and Evaluations: As a whole, our hypergraph kernel overcomes or is competitive to each of the alternative graph kernels in terms of the classification accuracies. Only the WL subtree kernel is competitive to our hypergraph kernel. The reason for this is that the WL subtree kernel relies on the original WL isomorphism test for undirected graphs, and like our kernel it can precisely capture all the isomorphic subtrees. However, our hypergraph kernel still outperforms the WL subtree kernel on most datasets. The directed line graphs used in our kernel can reflect richer characteristics than the original graphs, and our kernel also avoids the tottering problem that arises in the WL subtree kernel. Furthermore, our directed WL isomorphism test can precisely capture the directed information residing on the directed edges of the line graphs. In terms of the runtime, our hypergraph kernel is not the fastest kernel, but it can still finish the computation in a polynomial time. By contrast, some kernels cannot finish the computation on some datasets in on day.



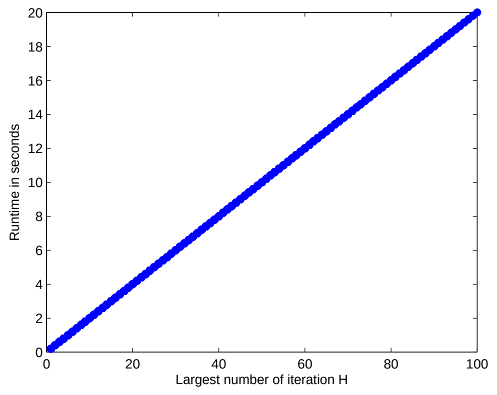


Figure 6.4: Runtime evaluations.

6.3.3 Computational Evaluation

In this subsection, we evaluate the computational efficiency (i.e., the CPU runtime) of our hypergraph kernel, and reveal the relationship between the computational overheads and the structural complexity of the associated hypergraphs.

Experimental setup: We evaluate the computational efficiency on randomly generated hypergraphs with respect two parameters: the hypergraph size n, and the largest number of iteration H for the directed WL isomorphism test. Separately, we vary $n = \{1, 2, \ldots, 100\}$ and $H = \{1, 2, \ldots, 100\}$. a) For the experiments with hypergraph size n, we generate 100 pairs of hypergraphs. The pairs of hypergraphs have an increasing number of vertices n connected by one hyperedge. In other words, here we investigate how the computational efficiency is effected by varying the relational order between vertices for hypergraphs. For each pair of hypergraphs, we set $H = 10$. We report the runtime for computing the kernel value for each pair of hypergraphs. b) For the experiments with the largest number of iteration H, we generate a pair of hypergraphs each of which has 50 vertices connected by one hyperedge. We report the runtime for computing the kernel values of the pair of hypergraphs based on different H. The runtime is reported in Fig.6.4, as operated in Matlab R2011a on a 2.5GHz Intel 2-Core processor (i.e., i5-3210m).

Experimental results: The left and right subfigures of Fig.6.4 present the experiments of the hypergraph kernel varying the parameters n and H, respectively. When varying the parameters n and H, we observe that the runtime scales quadratically with n and scales linearly with H. These computational evaluations verify that our hypergraph kernel can be computed in polynomial time.

6.4 Conclusions

In this paper, we have proposed a hypergraph kernel based on isomorphism tests. Our kernel is based on transforming a hypergraph into a directed line graph, which can not only accurately reflect the multiple relationships exhibited by the hypergraph but is also amenable to isomorphism tests. We develop a new directed WL isomorphism test for directed graphs. By performing the new isomorphism test on the directed line graphs of a pair of hypergraphs, the hypergraph kernel between the hypergraphs is computed in terms of the number of pairwise isomorphic in-subtrees and out-subtrees from the line graphs. Our kernel limits the tottering problem that arises in the existing random walk and subtree based (hyper)graph kernels. The experimental results demonstrate the effectiveness and efficiency of our kernel.

Chapter 7

Conclusions and Future Work

In this chapter, we first summarize the main contributions of this thesis. Moreover, we point out some of the weaknesses of this thesis. Finally, we give some suggestions for the future work.

7.1 Summary of Contributions

In this thesis, we have developed a new family of information-theoretic kernels, i.e., the Jensen-Shannon diffusion kernels, for either the unattributed or attributed graphs, using the Jensen-Shannon divergence measure. Furthermore, we have developed a novel framework of computing depth-based complexity traces for graphs. Based on the new framework, a novel depth-based matching kernel for graphs has also been developed. Finally, we have proposed a new hypergraph kernel based on isomorphism tests. Below, we provide a summary of these contributions for each chapter.

7.1.1 Information Theoretic Graph Kernels for Graphs

The first contribution of this thesis is to develop novel information theoretic kernels for graphs. To this end, in Chapter 3, we have defined a family of Jensen-Shannon diffusion kernels for (un)attributed graphs using the Jensen-Shannon divergence, that is an information theoretic measure. For the unattributed graphs, we compute the von Neumann or Shannon entropy for each graph in terms of the vertex degree. For the attributed graphs, we perform a tree-index (TI) label strengthening algorithm on attributed graphs. We compute a label Shannon entropy using the probability distribution associated with the strengthened labels. With the entropies for a pair of (un)attributed graphs to hand, we have shown how to compute the Jensen-Shannon diffusion kernel by measuring the entropy difference between the individual graph entropies and their composite entropy from the disjoint union graph (for unattributed graphs) or the composite probability distribution (for attributed graphs), using the Jensen-Shannon divergence.

We have shown that our new diffusion kernel for unattributed graphs overcomes the inefficiency arising in the R-convolution kernels. On the other hand, our new diffusion kernel for attributed graphs overcomes the shortcoming of discarding non-isomorphic substructures that arises in the R-convolution kernels. Moreover, this kernel also overcomes the shortcomings of restriction to attributed graphs, lacking correspondence information and reflecting limited interior topology information that arise in our diffusion kernel for unattributed graphs. The experimental results demonstrate the effectiveness and efficiency of our kernels.

7.1.2 Depth-Based Complexity Traces for Graphs

The second contribution of this thesis is to develop a novel framework of computing a depth graph complexity. To this end, in Chapter 4, we have shown how to construct a depth-based complexity trace for a graph, by combining the ideas of the entropy based graph complexity measures and the depth-based representations of graphs. For a graph, we have identified a centroid vertex by computing the minimum variance of its shortest path lengths, and obtained a family of dominant expansion subgraphs with increasing layer. The complexity trace of the graph is thus constructed by measuring how the graph entropies or the entropy differences vary with the subgraphs of increasing layer, as a function of depth.

We have shown that the depth-based complexity trace can not only be efficiently computed for a large graph (e.g., a graph having thousands of vertices) but also characterize a graph in a high dimensional complexity feature space. Experiments on graph datasets abstracted from bioinformatics and image data demonstrate the effectiveness and efficiency of our complexity traces in graph classification.

7.1.3 The Depth-Based Matching Kernel for Graphs

The third contribution of this thesis is to develop a novel matching kernel for graphs, based on the contribution in Chapter 4. To this end, in Chapter 5 we have described how to construct a depth-based graph kernel in terms of matching graphs based on the depth-based representation (i.e., the depth-based complexity trace) around each vertex, that reflects a high dimensional complexity characteristics of the graph around the vertex. Based on the obtained depth-based representations for two graphs we have defined a new matching strategy similar to that Scott et al. [19] previously used for point set matching. The resulted depth-based kernel is thus defined by counting the matched vertex pairs.

We have shown the relationship between the depth-based graph kernel and the all subgraph kernel and explained why our matching strategy incorporates structural correspondence into the kernel. We have empirically demonstrated the effectiveness and efficiency of our new kernel on synthetic graphs and real-world graphs abstracted from computer vision databases.

7.1.4 The Hypergraph Kernel Based on Isomorphism Tests

The novel methods developed in Chapters 3, 4 and 5 are only restricted on graphs, and thus cannot be performed for hypergraphs. To overcome the restriction, the fourth contribution of this thesis is to develop a novel hypergraph kernel. To this end, in Chapter 6 we have proposed a new Weisfeiler-Lehman hypergraph kernel based on isomorphism tests. Our kernel is based on transforming a hypergraph into a directed line graph, which not only accurately reflects the multiple relationships exhibited by the hypergraph but is also amenable to isomorphism tests. We have developed a new directed Weisfeiler-Lehman isomorphism test for directed graphs. By performing the new isomorphism test on the directed line graphs of a pair of hypergraphs, the hypergraph kernel between the hypergraphs is computed in terms of the number of pairwise isomorphic in-subtrees and out-subtrees from the line graphs.

We have shown that our new hypergraph kernel limits the tottering problem that arises in the existing random walk and subtree based (hyper)graph kernels. Experiments demonstrate the effectiveness and efficiency of our kernel.

7.2 Weaknesses

The novel methods proposed in this thesis outperform the state-of-the-art methods, however there are still a number of weaknesses to be noted. In this section, we discuss these weaknesses and analyze the reasons as follows. Specifically,

I) We have shown both theoretically and experimentally that the attributed Jensen-Shannon diffusion kernel can easily outperform the state-of-the-art graph kernels in terms of the classification accuracies. Unfortunately, this kernel still has several weaknesses. First, the attributed diffusion kernel can only capture the label information residing on the vertices. As a result, the label information residing on edges are discarded. This drawback limits the attributed diffusion kernel to reflect more detailed graph characteristics. Second, the attributed diffusion kernel requires expensive computation for computing the composite entropy for a pair of attributed graphs, since it needs to identify the correspondence between each pair of probability distributions in terms of the strengthened vertex labels. Third, the attributed diffusion kernel may also suffer from tottering problem. This is because each strengthened label from the required tree-index method corresponds to a subtree, and each subtree may include several copies of the same pairwise vertices connected by the same edge.

II) Though the proposed complexity trace can characterize a graph in a higher dimensional complexity feature space than the state-of-the-art graph complexity measures, there are still several weaknesses. First, the complexity trace method cannot accommodate attributed graphs, thus this method cannot reflect any label information residing on either the edges or the vertices. Second, the required von Neumann entropy or the Shannon entropy associated with the steady state random walk is computed using the vertex degree, which is structurally simple. Furthermore, for the Shannon entropy computed from the information functionals, the required local information graph rooted at a vertex is also structurally simple. As a result, the complexity trace using these entropies may discard some topology information.

III) Though the depth-based matching kernel overcomes the shortcoming of ignoring the location information between substructures arising in the state-of-the-art R-convolution kernels, there are still some weaknesses. First, the matching kernel is related to the depth-based complexity trace around each vertex, thus this kernel also suffers from the same weaknesses that arise in the depth-based complexity trace method. Furthermore, the matching kernel only identifies the correspondence between vertices, and thus ignores the correspondence information between edges. As a result, the matching kernel may discard potential similarity information for a pair of graphs.

IV) The proposed hypergraph kernel not only accommodates both graphs and hypergraphs but also limits the tottering problem arising in the state-of-the-art (hyper)graph kernels using the subtrees and walks. However, this kernel cannot completely avoid the tottering problem. This is because the proposed directed Weisfeiler-Lehman (WL) isomorphism test algorithm on a directed line graph cannot guarantee that any vertex is only visited one time. As a result, the in or out subtree identified by the directed WL isomorphism test algorithm may still contain several copies of the same vertices. Moreover, the proposed directed WL isomorphism test algorithm cannot be directly performed on a hypergraph, thus we need extra computation for transforming the hypergraph into a directed line graph. This may influence the computational efficiency for the hypergraph kernel.

7.3 Future Work

To address the weaknesses of this thesis, in this section we suggest some possible approaches to overcoming them for further research. Furthermore, we also provide a number of ways for extending the work in this thesis.

I) To overcome the problems of the attributed Jensen-Shannon diffusion kernel, we may consider the following strategies. First, we may develop a new label strengthening method that not only limits or avoids the tottering problem but also accommodates the label information residing on both the vertices and the edges. This may provide us an elegant way for overcoming the problems of tottering and discarding edge label information that arise in the attributed diffusion kernel. Second, we may also consider to define a new feature selection [105] method for the objective of selecting more discriminating labels. As a result, we may overcome the inefficiency of the attributed diffusion kernel, since we can only identify the correspondence information between a number of selected labels. Moreover, since some redundant labels may be not included in the selected labels, the attributed diffusion kernel may reflect more precise similarity measure between a pair of graphs.

Furthermore, we also consider to extend the Jensen-Shannon diffusion kernel in the following way. In prior work Hancock et al. have developed methods for characterising graphs using the commute time [106] and the heat kernel [107]. Both the commute time and heat kernel of an undirected graph encapsulate the path length information between pairs of vertices. It would be interesting to use the commute time or heat kernel as a means of computing a probability distribution for a graph. We thus can define a new graph kernel based on the Jensen-Shgannon divergence and the new probability distributions for graphs.

II) To overcome the problems arising in the complexity trace method for graphs, we may consider the following strategies. First, to overcome the shortcoming of discarding label information, we may perform a label strengthening algorithm (e.g., the TI method required for the attributed Jensen-Shannon diffusion kernel) and thus obtain a probability distribution in terms of the label frequency. A label Shannon entropy with the probability distribution can be computed. As a result, we may compute a depth-based complexity trace for an attributed graph using the label Shannon entropy. Second, to overcome the shortcoming of structurally simple problem, we may consider to perform the continuous time quantum walk (CTQW) [109, 110, 111] for a graph to assign each vertex a probability and thus obtain a probability distribution in terms of the CTQW. Since the CTQW is not dominated by the low frequency of the Laplacian spectrum, the CTQW is potentially able to discriminate better among different graph structures. As a result, we may compute a new depth-based complexity trace associated with the CTQW that reflects more complicated graph topology information. As we have stated, the depth-based matching kernel is related to the depth-based complexity trace, the two strategies may be also useful for overcoming the shortcoming of the kernel.

Furthermore, we also consider to extend the complexity trace method in a number of ways. First, the depth-based complexity trace not only provides a way for characterising complexity with depth in a graph, but also opens up directions for future research. Our method allows the inhomogeneity of complexity with depth to be used as a relatively compact yet potential detailed characterisation of graph structure. In this thesis we have concentrated on using the method for construct a vectorial signature for the purposes of classifying graphs. Of course, the characterisation could be used for a number of different tasks including the construction of graph kernels [112] and graph embedding [1]. Additionally, instead of using a feature-vector, we could also incorporate the complexity-depth characterisation into a tree or string representation of a graph. Finally, different entropy and divergence measures are available as measures of complexity [30, 110, 113], and these would also provide interesting alternatives for investigation. Suffice to say, studies along these lines are underway and will be reported in due course.

III) To completely avoid the tottering problem arising in our hypergraph kernel, we may consider some non-backtracking substructures for the directed line graph (e.g., the shortest paths or cycles). We may also consider to extend the hypergraph kernel in following ways. First, we may define a new high order Weisfeiler-Lehman isomorphism test method for hypergraphs. Thus, we can define a new hypergraph kernel by directly measuring the isomorphism between a pair of hypergraphs. Furthermore, in [114] we have explored the use of the discrete-time quantum walks [115, 116] on the directed line graph, which can be constructed by transforming a hypergraph. It would be interesting to extend this work, using the discrete-time quantum walks to compute the von Neumann entropy associated with the quantum state. This may provide a more principled means of computing a quantum kernel for hypergraphs.

Glossary of Notation

$G(V, E)$	Graph with vertex set V and edge set E
H_S	Shannon entropy
H_{VN}	von Neumann entropy
A	Adjacency matrix of a graph
D	Degree matrix of a graph
L	Laplacian matrix of a graph
$\hat{L}$	Normalized Laplacian matrix of a graph
$\hat{\Phi}$	Eigenvector of the normalized Laplacian matrix
$\hat{\Lambda}$	Eigenvalue of the normalized Laplacian matrix
S_G	Shortest path matrix of a graph
k	kernel function
δ	Dirac kernel
D_{JS}	Jensen-Shannon divergence
$HG(V_H, E_H)$	Hypergraph with vertex set V_H and hyperedge set E_H

References

[1] H. Bunke and K. Riesen. Improving vector space embedding of graphs through feature selection algorithms. In *Pattern Recognition*, vol. 44, pp. 1928-1940, 2010.

[2] E. Pekalska, R.P.W. Duin, and P. Paclík. Prototype selection for dissimilarity-based classifiers. In *Pattern Recognition*, vol. 39(2), pp. 189-208, 2006.

[3] R.C. Wilson, E.R. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1112-1124, 2005.

[4] P. Ren, R.C. Wilson, and E.R. Hancock. Graph characterization via Ihara coefficients. In *IEEE Transactions on Neural Networks*, vol. 22, pp. 233-245, 2011.

[5] D.J. Cook and L.B. Holder Eds. Mining graph data. Wiley-Interscience, New Jersey, 2006.

[6] T. Gärtner. A survey of kernels for structured data. In *ACM Special Interest Group on Knowledge Discovery and Data Mining*, vol. 5, pp. 49-58, 2003.

[7] N. Shervashidze, P. Schweitzer, E.J. Leeuwen, K. Mehlhorn, and K.M. Borgwardt. Weisfeiler-Lehman graph kernels. In *Journal of Machine Learning Research*, vol. 1, pp. 1-48, 2010.

[8] C. Berge. Hypergraphs: Combinatorics of finite sets. North-Holland, 1989.

[9] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1-8.

[10] M. Chertok and Y. Keller. Efficient high order matching. In *IEEE Transaction on Pattern Analysis Machine Intelligence*, vol. 10, pp. 2205-2215, 2010.

[11] A. Shashua and A. Levin. Linear image coding for regression and classification using the tensor-rank principle. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 623-630.

[12] P. Ren, T. Aleksic, R.C. Wilson, and E.R. Hancock. A polynomial characterization of hypergraphs using the Ihara zeta function. In *Pattern Recognition*, vol. 44, pp. 1941-1957, 2011.

[13] G. Wachman and R. Khardon. Learning from interpretations: a rooted kernel for ordered hypergraphs. In *Proceedings of International Conference on Machine Learning*, 2007, pp. 943-950.

[14] P. Lamberti, A. Majtey, A. Borras, M. Casas, and A. Plastino. Metric character of the quantum Jensen-Shannon divergence. In *Physical Review A*, vol. 77, 052311, 2008.

[15] L. Han, F. Escolano, and E.R. Hancock. Graph characterizations from von Neumann entropy. In *Pattern Recognition Letters*, vol. 33(15), pp. 1958-1967, 2012.

[16] N. Dahm, H. Bunke, T. Caelli, and Y. Gao. A unified framework for strengthening topological node features and its application to subgraph isomorphism detection. In *Proceedings of Graph-Based Representations in Pattern Recognition (GbRPR)*, 2013, pp. 11-20.

[17] F. Escolano, E.R. Hancock, and M.A. Lozano. Heat diffusion: Thermodynamic depth complexity of networks. In *Physical Review E*, vol. 85, pp. 206236, 2012.

[18] J.P. Crutchfield and C.R. Shalizi. Thermodynamic depth of causal states: Objective complexity via minimal representations. In *Physical Review E*, vol. 59, pp. 275283, 1999.

[19] G.L. Scott and H.C. Longuett-Higgins. An algorithm for associating the features of two images. In *Proceedings of the Royal Society of Londan B*, pp. 244:313-320, 1991.

[20] B. Weisfeiler and A.A. Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. In *Nauchno-Technicheskaya Informatsia*, vol. Ser2(9), 1968.

[21] F. Aziz, R.C. Wilson, and E.R. Hancock. Backtrackless walks on a graph. In *IEEE Transections on Neural Network and Learning System*, vol. 24, pp. 977-989, 2013.

[22] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of International Conference on Machine Learning (ICML)*, 2003, pp. 321-328.

[23] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, vol. 20(3), pp. 273.

[24] B. Schölkopf, A. Smola, K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. In *Neural Computation*, vol. 10(5), pp. 1299-1319, 1998.

[25] K. Riesen and H. Bunke. Classification and clustering based on vector space embedding. *World Scientific Publishing*, 2010.

[26] T. Gartner, P.A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of Conference on Learning Theory*, PP. 129-143, 2003.

[27] B. Schölkopf, K. Tsuda, and J.-P. Vert. Kernel methods in computational biology. MIT press, 2004.

[28] J. Shawe-Taylor and N. Cristianini. Kernel methods for pattern analysis. Cambridge University Press, 2004.

[29] C.M. Bishop. Pattern recognition and machine learning. Springer, 2006.

[30] A.F. Martins, N.A. Smith, E.P. Xing, P.M. Aguiar, and M.A. Figueiredo. "Nonextensive information theoretic kernels on measures. In *Journal of Machine Learning Research*, vol. 10, pp. 935-975, 2009.

[31] J. Lafferty and G. Lebanon. Diffusion statistical manifolds. In *Journal of Machine Learning Research*, vol. 6, pp. 129C163, 2005.

[32] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. (T.R.) Universität Dortmund, 1997.

[33] M. Cuturi, K. Fukumizu, and J.P. Vert. Semigroup kernels on measures. In *Journal of Machine Learning Research*, vol. 6, pp. 1169C1198, 2005.

[34] M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *Proceedings of 10th AISTATS*, 2005.

[35] M. Havrda and F. Charät. Quantification method of classification processes: Concept of structural α-entropy. Kybernetika, vol. 3, pp. 30C35, 1967.

[36] M. Gell-Mann and M. Tsallis. Nonextensive entropy: Interdisciplinary application. Oxford University Press, 2004.

[37] M. Cuturi and J.P. Vert. Semigroup kernels on finite sets. In *Advances in Neural Information Processing Systems*, vol. 17, pp. 329C336, 2005.

[38] D. Haussler. Convolution kernels on discrete structures. In *Technical Report UCS-CRL-99-10 of University of California at Santa Cruz*, 1999.

[39] K.M. Borgwardt and H.P. Kriegel. Shortest-path kernels on graphs. In *Proceedings of IEEE International Conference on Data Mining*, 2005, pp. 74-81.

[40] F. Costa and K.D. Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of International Conference on Machine Learning (ICML)*, 2010, pp. 255-262.

[41] Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[42] F.R. Bach. Graph kernels between point clouds. In *Proceedings of International Conference on Machine Learing (ICML)*, 2008, 25-32.

[43] N. Kriege and P. Mutzel. Subgraph matching kernels for attributed graphs. In *Proceedings of International Conference on Machine Learing (ICML)*, 2012.

[44] U. Brandes and T. Erlebach Eds. Network analysis. Springer, Berlin Heidelber, 2005.

[45] A.N. Kolmogorov. Three approaches to the definition of infornmation (in Russian). In *Problemy Peredachi Informatsii*, vol. 1, pp. 3-11, 1965.

[46] A. Mowshowitz and M. Dehmer. Entropy and the complexity of graph revisited. In *Entropy*, vol. 14, pp. 559-570, 2012.

[47] D. Bonchev and D.H. Rouvray Eds. Complexity in chemistry, biology, and ecology, mathematical and computational chemistry. Springer, New York, 2005.

[48] D. Bonchev. Complexity analysis of yeast proteome network. In *Chemistry & Biodiversity*, vol. 1, pp. 312-326, 2004.

[49] P. Pudlák, V. Rödl, and P. Savickiý. Graph complexity. In *Acta Informatica*, vol. 25, pp. 515-535, 1988.

[50] D.P. Feldman and J.P. Crutchfield. Measures of statistical complexity: Why?. In *Physics Letters A*, vol. 238, pp. 244252, 1998.

[51] J. Korner. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *Proceedings of Transactions of of the 6th Prague Conference on Information Theory*, 1973, pp. 411-425.

[52] M. Dehmer and A. Mowshowitz. A history of graph entropy measures. In *Information Sciences*, vol. 181, pp. 57-78, 2011.

[53] M. Dehmer. Information processing in complex network: Graph entropy and information functionals. In *Applied Mathematics and Computing*, vol. 201, pp. 82-94, 2008.

[54] F. Passerini and S. Severini. Quantifying complexity in networks: The von Neumann entropy. In *International Journal of Agent Technologies and Systems*, vol. 1, pp. 58-67, 2009.

[55] K. Anand, G. Bianconi, and S. Severini. Shannon and von Neumann entropy of random networks with heterogeneous expected degree. In *Physical Review E*, vol. 83, pp. 036109, 2011.

[56] S.L. Braunstein, S. Ghosh, and S. Severini. The Laplacian of a gaph as a density matrix: A basic combinatorial approach to separability of mixed states. In *Annals of Combinatorics*, vol. 10, pp. 291-317, 2006.

[57] C.D. Godsil, G. Royle, and C.D. Godsil. Algebraic graph theory. *Springer*, New York, 2001.

[58] A. Robles-Kelly and E.R. Hancock. Steady state random walks for path estimation. In *Proceedings of Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pp. 143-152, 2004.

[59] C. Ye, R.C. Wilson, L.F. Costa, and E.R. Hancock. Entropy and heterogeneity for directed graphs. In *Proceedings International Workshop on Similarity-Based Pattern Recognition* , 2013, pp. 219-234.

[60] F. Chung. Laplacians and the Cheeger Inequailty for Directed Graphs. In *Annals of Combinatorics*, vol. 9, pp. 1-19, 2005.

[61] R. Nock and F. Nielsen. Fitting the smallest enclosing Bregman ball. In *Proceedings of International Conference of Machine Learning*, 2005, pp. 649-656.

[62] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie. Beyond pairwise clustering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 838-845.

[63] D. Zhou, J. Huang, and B. Scölkopf. Learning with hypergraphs: clustering, classification, and embedding. In *Proceedings of Advanced Neural Information Processing System*, 2007, pp. 1601-1608.

[64] O. Duchenne, F.R. Bach, I.S. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1980-1987.

[65] A. Shashua, R. Zass, and T. Hazan. Multi-way clustering using super-symmetric non-negative tensor factorization. In *Proceedings of European Conference on Computer Vision*, 2006, pp. 595-608.

[66] Z. He, A. Cichocki, S. Xie, and K. Choi. Detecting the number of clusters in n-

way probabilistic clustering. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 2006-2012, 2010.

[67] V.M. Govindu. A tensor decomposition for geometric grouping and segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 1150-1157.

[68] G. Chen and G. Lerman. Spectral curvature clustering. In *International Journal of Computer Vision*, vol. 81, pp. 317-330, 2009.

[69] G. Chen and G. Lerman. Foundations of a multi-way spectral clustering framework for hybrid linear modeling. In *Journal of Foundations of Computational Mathematics*, vol. 9, pp. 517-558, 2009.

[70] G. Chen, S. Atev, and G. Lerman. Kernel spectral curvature clustering. In *Proceedings of ICCV Workshop on Dynamical Vision*, 2009, pp. 317-330.

[71] S.R. Bulo and M. Pelillo. A game-theoretic approach to hypergraph clustering. In *Proceedings of Advanced Neural Information Processing System*, 2009, pp. 1571-1579.

[72] O. Carroll and M. Krotoski. Using 'Digital Fingerprints' (or Hash Values) for Investigations and Cases Involving Electronic Evidence. In *United States Attorneys Bulletin*, vol. 62, 44-82, 2014.

[73] L. Bai and E.R. Hancock. Graph kernels from the Jensen-Shannon divergence. In *Journal of Mathematical Imaging and Vision*, vol. 47, pp. 60-69, 2013.

[74] R. Boulet. Disjoint unions of complete graphs characterized by their Laplacian spectrum. In *Electr. J. Lin. Alg*, vol. 18, pp. 773-783, 2009.

[75] R.I. Kondor and J.D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of Internitional Conference on Machine Learning*, 2002, pp. 315-322.

[76] J.D. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. In *Journal of Machine Learning Research*, vol. 6, pp. 129-163, 2005.

[77] A.K. Debnath, R.L. Lopez de Compadre, G. Debnath, A.J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds, Correlation with molecular orbital energies and hydrophobicity. In *J. of Med. Chem.*, vol. 34, pp. 786-797, 1991.

[78] P.D. Dobson and A.J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. In *J. Mol. Biol.*, vol. 330, pp. 771-783, 2003.

[79] N. Wale and G. Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. In *Proceedings of IEEE International Conference on Data Mining*, 2006, pp. 678-689.

[80] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg. The enzyme database: Updates and major new developments. In *Nucleic Acids Research*, vol. 32, pp. 431-433, 2004.

[81] L. Bai, E.R. Hancock, A. Torsello, and L. Rossi. A quantum Jensen-Shannon graph kernel using the continuous-time quantum walk. In *Proceedings of Graph-Based Representations in Pattern Recognition (GbRPR)*, 2013, pp. 121-131.

[82] N. Shervashidze, S.V.N. Vishwanathan, T. Petri, K. Mehlhorn, and K.M. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Journal of Machine Learning Research*, vol. 5, pp. 488-495, 2009.

[83] C.-C Chang and C.-J. Lin. LIBSVM: A library for support vector machines, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[84] A. Mowshowitz. Entropy and the complexity of the graphs I: An index of the relative complexity of a graph. In *Bulletin Mathematical Biophys*, vol. 30, pp. 175-204, 1968.

[85] N. Rashevsky. Life, information theory, and topology. In *Bulletin Mathematical Biophys*, vol. 17, pp. 229-235, 1955.

[86] E. Trucco. A note on the information content of graphs. In *Bulletin Mathematical Biophys*, vol. 18, pp. 129-135, 1956.

[87] C. Jordan. Sur les assemblages des lignes. In *J. Reine Angew. Math*, vol. 70, pp. 185-190, 1969.

[88] P.J. Slater. Centers to centroids in graphs. In *Journal of Graph Theory*, vol. 2, pp. 209-222, 1978.

[89] A. Dutot, D. Olivier, and G. Savin. Centroids: A decentralized approach. In *Proceedings of Emergent Properties in Natural and Artificial Complex Systems within ECCS*, 2011, pp.23-28.

[90] A.P. Santhakumaran. Centroid of a graph with respect to edges. In *Mathemitical Science*, vol. 19, pp. 13-23, 2010.

[91] D.B. Johnson. Efficient algorithms for shortest paths in sparse networks. In *Journal of the ACM*, vol. 24, pp. 1-13, 1977.

[92] S.A. Nene, S.K. Nayar, and H. Murase. Columbia object image libary (COIL-20), Dept. Comput. Sci., Columbia Univ., New York, Tech. Rep. CUCS-005-96, Feb. 1996.

[93] C.G. Harris and M.J. Stephens. A combined corner and edge detector. In *Proceedings of Alvey Vision Conference*, 1988, pp. 147-151.

[94] N.I. Bersano-Méndez, S.E. Schaeffer, and J. Bustos-Jiménez. Metrics and models for social networks. In *Ajith Abraham, Aboul-Ella Hassanien, (Eds.) Computational Social Networks, Springer*, pp. 115-142, 2012.

[95] H. Bunke and K. Riesen. Improving vector space embedding of graphs through feature selection algorithms. In *Pattern Recognition*, vol. 44, pp. 1928-1940, 2011.

[96] K.M. Borgwardt, C.S. Ong, S. Schoenauer, S.V.N. Vishwanathan, A.J. Smola, and H.P. Kriegel. Protein function prediction via graph kernels. In *Bioinformatics*, vol. 21 (Suppl 1), pp. 47-56, 2005.

[97] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Schölkopf, B., Burges, C.J.C., and Smola, A.J. (Eds.) Advances in Kernel Methods*, MIT Press, pp. 185-208, 1999.

[98] W.S. Sanders, C.I. Johnston, S.M. Bridges, S.C. Burgess, and K.O. Willeford. Prediction of cell penetrating peptides by support vector machines. In *PLoS Computational Biology*, vol. 7, e1002101, 2011.

[99] J. Munkres. Algorithms for the assignment and transportation Problems. In *Journal of the Society for Industrial and Applied Mathematics*, vol. 5(1), pp. 32-58, 1957.

[100] V. Barra, and S. Biasotti. 3D shape retrieval using Kernels on Extended Reeb Graphs. In *Pattern Recognition*, vol. 46, pp. 2985-2999, 2013.

[101] G. McCarter and A. Storkey. Air Freight Image Segmentation Database, 2007. Available at http://homepages.inf.ed.ac.uk/amos/afreightdata.html.

[102] F. Escolano, E.R. Hancock and M.A. Lozano. Graph matching through entropic manifold alignment. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, PP. 2417-2424.

[103] S. Biasotti, S. Marini, M. Mortara, G. Patanè, M. Spagnuolo, and B. Falcidieno. 3D shape matching through topological structures. In *Proc. DGCI* 2003, pp. 194-203.

[104] I.H. Witten, E. Frank, and M.A. Hall. Data Mining: Practical machine learning tools and techniques. *Morgan Kaufmann*, 2011.

[105] K.M. Borgwardts. Graph Kernels. PhD thesis, Munchen, 2007.

[106] H. Qiu, and Edwin R. Hancock, "Clustering and embedding using commute times," In *IEEE Transactions on Pattern Analysis Machine Intellgence*, vol. 29, pp. 1873-1890, 2007.

[107] B. Xiao, Edwin R. Hancock, and Richard C. Wilson. Graph characteristics from the heat kernel trace. In *Pattern Recognition*, vol. 42, pp. 2589-2606, 2009.

[108] T. Gärtner, P.A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of Conference on Learning Theory*, 2003, pp. 129-143.

[109] L. Bai, L. Rossi, A. Torsello, and E.R. Hancock. "A quantum Jensen-Shannon kernel for unattributed graphs," To appear in *Pattern Recognition*, 2014.

[110] L. Bai, E.R. Hancock, A. Torsello and L. Rossi. A quantum Jensen-Shannon graph kernel using the continuous-time quantum walk. In *Proceedings of International Workshop on Graph-Based Representations in Pattern Recognition (GbRPR)*, 2013, pp. 121-131.

[111] L. Bai, L. Rossi, H. Bunke and E.R. Hancock. Attributed graph kernels using the Jensen-Tsallis q-differences. To appear in *Proceedings of European Conference on*

Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2014.

[112] M. Neuhaus, and H. Bunke. Bridging the gap between graph edit distance and kernel machines, *World Scientific*, 2007.

[113] A. Majtey, P. Lamberti, and D. Prato. Jensen-shannon divergence as a measure of distinguishability between mixed quantum states. In *Physical Review A*, vol. 72, 052310, 2005.

[114] P. Ren, T. Aleksic, D. Emms, R.C. Wilson, and E.R. Hancock. Quantum walks, ihara zeta functions and cospectrality in regular graphs. In *Quantum Information Processing*, vol. 10, pp. 405-417, 2011.

[115] A.M. Childs. Universal computation by quantum walk. In *Physical Review Letters*, vol. 102(18), 180501, 2009.

[116] P. Dirac, The Principles of Quantum Mechanics, 4th edn, *Oxford Science Publications*, 1958.