Generative Models for Graphs

David H. White

Submitted for the degree of Doctor of Philosophy

Department of Computer Science
THE UNIVERSITY of York

October 2009

Abstract

In this thesis we present three different approaches for constructing generative models of relational graphs. The central problem to be solved is how to capture structural variations in a sample set of graphs in such a way that allows new graphs to be generated from the distribution.

While methods for constructing generative models are well-known when the sample data are in vectorial form, by comparison little work has been done when the sample data are relational graphs. This is because working with graphs presents two problems; firstly, the order over the vertices of a graph is arbitrary and secondly, the number of vertices in each sample graph may vary. Both these problems mean that defining statistical quantities such as the mean and variance on a graph set is difficult.

One solution is to perform an alignment step on the vertices of each sample graph thus placing them in a canonical order. From this canonical representation we can construct a vectorial description of each graph and use standard techniques to construct a parameterized distribution over this vector space. This is the approach taken in our first generative model, *Vectorial Generative Models for Graphs*. However, while it is possible to sample vectors describing new graphs from the defined distribution, the recovery of a graph from its vectorial representation is a challenging problem. We investigate a number of different methods of constructing the vector spaces to aid in this recovery process.

One vector space that is of special interest is that resulting from the spectral representation of a graph. By using this representation we gain access to the different levels of structure present in the graph. We have shown in our work on *Mixing Spectral Representations of Graphs* that we may combine different levels of structure from two graphs by mixing the eigenmodes of both graphs in different proportions. It is this mixing process that results in the spectral representation producing such a successful generative model.

In our second generative model, *Parts Based Generative Models for Graphs*, we adopt a different approach. By segmenting each sample graph into a set of subgraphs, we can model structural variations in the sample set in terms of the

subgraphs and the connections between subgraphs. Again we insist that our model is truly generative, in the sense that we may draw new graphs from the sample distribution.

Our final contribution is to provide a *Generative Model for Chemical Structure*. This presents a difficult challenge since chemical structures are constrained by the laws of chemistry. In our approach we avoid the possibility of generating invalid chemical structures by defining a method of projecting an invalid chemical structure onto the nearest valid structure. Therefore, we need not concern ourselves with generating valid structures directly; we sample from the distribution and correct samples representing invalid chemical structures using our projection method.

We apply this approach to the domain of drug discovery. First we note that the function of a drug is largely due to its structural configuration. Therefore, by populating the sample set with drugs that are known to be effective against a specific target, we generate new structurally similar molecules which should also be effective against the target. We confirm this by using a docking program to simulate the molecular interactions between the generated structures and the target.

Contents

List of Figures

Acknowledgements

Firstly, I would like to express my sincere thanks to my supervisor, Dr Richard C. Wilson. I would not have been able to complete this research without his excellent guidance and support. He has always been available when I have needed help and for this I am very grateful.

I would also like to thank Professor Edwin R. Hancock for being my internal assessor and providing me with useful feedback and discussion at the different milestones of my PhD.

I would like to thank the EPSRC for providing the funding that allowed me to undertake this research and the Department of Computer Science as a whole for being my home for the last 8 years throughout both my PhD and undergraduate degree.

I would like to thank all my friends that have supported me and made life so enjoyable.

Finally, I would like to dedicate this work to my parents and my sister who have given me so much support over the years.

Declaration

I declare that all the work in this thesis is solely my own except where attributed and cited to another author. Some of the material in the following chapters has been previously published in the cited places; for Chapter 3, the work has been published in a paper titled *Mixing Spectral Representations of Graphs* [115]; for Chapter 4, the paper *Spectral Generative Models for Graphs* [116] describes the dual-spectral generative model and for Chapter 5, the paper *Parts Based Generative Models for Graphs* [117] provides an overview of the proposed generative model.

Symbols & Operators

The following conventions are used for symbols throughout. In addition, for each chapter a table of symbols detailing their meaning is given at the end of that chapter.

Uppercase calligraphy letters	Sets ($\mathcal{S}, \mathcal{P}, \mathcal{G}$)
Uppercase letters	Single Objects (S, P, G)
Uppercase boldface letters	Matrices ($\mathbf{X}$, $\mathbf{\Phi}$, $\mathbf{\Lambda}$)
Lowercase boldface letters	Vectors ($\mathbf{x}$, $\mathbf{a}$, $\boldsymbol{\mu}$)
Lowercase letters	Functions / Scaler Values (w, ρ, ξ, ζ)

We make use of the following operators:

$\mathrm{vec}(\mathbf{X})$	Converts a matrix to a vector.
$\mathrm{devec}(\mathbf{x})$	Converts a vector back into a matrix.
$\mathcal{N}(Mean, Variance)$	Returns a sample from the normal distribution.

The spectral decomposition is used in two different ways in this work. When we consider the spectral decomposition of a matrix representation of a graph then we use the symbols $\mathbf{E}$ and $\mathbf{V}$ to represent the eigenvalues and eigenvectors respectively. When the spectral decomposition is used to perform PCA we use the symbols $\mathbf{\Lambda}$ for the eigenvalues and $\mathbf{\Phi}$ for the eigenvectors.

Chapter 1

Introduction

1.1 Overview

In 1736 Leonhard Euler published a paper on the *Seven Bridges of Königsberg* which is generally considered to be the first paper on graph theory. The paper details the now famous problem of how a circular walk may be taken through the city of Königsberg such that each bridge is crossed exactly once. Euler showed that no such walk was possible since at least one landmass had an odd number of bridges. While the solution is elegant, it is the method of arriving at the solution that is of interest to us. By removing all unnecessary information Euler produced a *relational graph* describing the layout of the bridges and landmasses. Relational graphs are now used as tools for expressing abstract relationships between sets of objects in almost every scientific discipline.

Put formally, a relational graph is a set of *vertices* which describe the objects in question, and a set of *edges* which connect two vertices and therefore capture relationships between the objects. As such, relational graphs provide a very general framework for expressing relational data. One interesting application of relational graphs is their ability to express chemical structure. By mapping the atoms to vertices and the bonds between atoms to edges a graph may be constructed for a particular chemical structure. This mapping can be extended if we allow vertices and edges to take on *attributes*. For example, we could attribute each vertex with the atomic label of the atom it corresponds to and each edge could be attributed with the type of the molecular bond.

In the field of *pattern recognition* graphs are commonly used to represent examples if the source data that the processes are being applied to is relational in nature. However, if pattern recognition algorithms are to be applied to relational graphs then we must have a method of representing graphs. Constructing a *graph representation* is not a simple operation due to the abstract nature and powerful representational ability of graphs.

The most problematic property of relational graphs is that the ordering in the vertex set is arbitrary. For example, if we want to determine whether two graphs are isomorphic then we must either establish a set of correspondences between the vertices of both graphs or construct a representation that is invariant to vertex order. A second issue arises when constructing representations for graphs that vary in the number of vertices. This issue is most apparent when we have one graph that is a subgraph of another. If we wish to establish that the subgraph really is part of a larger graph then our representation must be capable of handling graphs with varying numbers of vertices.

One representation that has attracted considerable interest recently is the *spectral representation* of a graph. In addition to partially solving the vertex correspondence problem, the representation allows the analysis of different levels of relational structure present in a graph. This can range from a view favouring local topology to one considering global graph structure.

A common task in pattern recognition is to construct a *model* describing the *distribution of sample data*. If the data in question are relational graphs then such a model will explain structural variations in the graph set. For example, if the sample data were graphs of a certain individual's face then the model would describe the variations in facial poses present in the sample data. Such a model could then be used to classify previously unseen images of people's faces to determine whether an image was of the individual that the system was trained with.

The problem of constructing *generative models* for graph structure can be viewed as the reverse of the classification operation described above. In other words, given a set of sample data, how may we draw additional examples from the distribution of sample data? This process is well known when the data is in vectorial form but, due to the problems outlined above, it is difficult to construct a vectorial representation of a relational graph. Furthermore, if we were to use a vectorial representation then

we would require a reconstruction step that allowed us to recover a graph from the representation. However, if these problems can be overcome then standard statistical techniques can be applied to the graphs in vectorial form.

In a paper by Luo et al [75] a method is detailed for constructing a generative model over a set of graphs by representing each graph as a vector. The variations in the resulting vector space are then analyzed using principle component analysis (PCA). However, while it is possible to sample new graphs using their model, a reconstruction process is required and the authors do not use their model in such a way. A similar model is suggested by Xiao & Hancock [126] which uses the spectral representation of a graph to construct a vectorial representation. However, again the authors do not describe how their model may be used in the generative sense.

A different approach to constructing generative models for graphs has been explored by Torsello and Dowe [103]. By making the simplifying assumption that the observation of each vertex or edge is independent from all others, the existence of each vertex and each edge is modeled as a Bernoulli trial. To learn the model, an approach similar to expectation-maximization is used which estimates the correspondences between each sample graph and the model graph on one step and then updates the model parameters on the next step. While it is possible to sample from this model, the assumption of vertex and edge independence results in no co-occurrences of edges or vertices being considered. These co-occurrences are the ingredients of describing structural variation successfully and therefore this model would be unlikely to generate examples drawn from the input distribution.

1.2 Goals

In this section we outline the goals of this research, which are two-fold.

Firstly, we wish to investigate methods of constructing generative models for graphs. In all cases we will require that the models are fully generative in the sense that we can draw new graphs from the distributions. For each method developed we want to experimentally evaluate the quality of the solution. This involves the assessment of the applicability of the distribution we fit to the sample data and the quality of the generated graphs to name but a few points. Furthermore, we want to

evaluate under which graph types each approach performs best.

The second goal is to take what we have developed for constructing generative models for graphs and extend this to the domain of chemical structure. Chemical structures are an interesting domain because the data is restricted by a high-level set of constraints resulting from the laws of chemistry. Any methods developed here would be of interest in the field of drug discovery since the exploration of related chemical structures frequently occurs in the search for new drugs.

1.3 Thesis Outline

We will now give an outline of the remainder of this thesis.

In chapter 2 we describe the literature relevant to the research detailed in this thesis. In section 2.1 we begin by considering the construction of graph features. We then discuss graph similarity which leads to graph matching. Next, we turn to the field of graph segmentation which is followed by literature on constructing graph means and medians. Finally, we survey work on generative models for graphs.

In section 2.2 we discuss the field of chemoinformatics which is the application of computational methods to solve problems in chemistry. This is relevant to our work on applying generative models to chemical structure. We begin by looking at methods of representing chemical structure and then proceed to discuss measuring chemical structure similarity. Next, we survey work on the generation of chemical structures. Finally, we describe docking which we can use as an evaluation domain in which the interaction between a molecule and a protein can be simulated.

We begin the description of our work in chapter 3 where we discuss mixing spectral representations of graphs. This work is a precursor to an actual generative model based on spectral representations which is described in chapter 4. In chapter 3 we show how the spectral representation allows the mixing of graph structures on different levels by combining, for example, the global structure of one graph with the local structure of another. This approach can also be applied to computing an approximate median spectral graph which we compare with the true generalized median graph for small sets of graphs.

In chapter 4 we present our first generative models. These models are based on constructing vectorial representations for each graph in the sample set and then

fitting a parameterized distribution over the points in the resulting vector space. A reconstruction step is described that allows a graph to be recovered from a vectorial representation. In some cases the generated vectors do not display all the properties that we require for our graph representations, which hinders graph recovery. To solve this we consider constructing vector spaces that are produced from manifolds to enforce the properties we require in the generated vectors. We perform an extensive evaluation of the different models by considering the normality of the data in each vector space, the classification accuracy, the reconstruction of graphs using limited components of the model and the quality of the generated graphs. Results are given for three different data sets; random graphs, Delaunay graphs and graphs from the COIL data set.

In chapter 5 we present a parts based generative model for graph structure. By decomposing each sample graph into a set of subgraphs, each sample graph can be represented by a set of subgraphs and a set of connections between subgraphs. We then form models on the distribution of subgraphs and the distribution of connections between subgraphs. By sampling from these models we may generate new graphs. The approach is especially effective on graph types that lend themselves to segmentation such as those constructed from images of scenes or articulated objects. The method is evaluated on both synthetic and real-world data and since the graphs in question have quite a rigid structure we can actually visualize the generated graphs.

In chapter 6 we apply the methods previously developed in constructing generative models to the domain of chemical structures. By developing a method of projecting invalid chemical structures onto the nearest valid chemical structure, we can approximate sampling the input distribution while retaining the ability to generate only valid chemical structure. Furthermore, by populating the sample set with drugs that are known to be effective for a particular target, we hope to generate chemical structures that might also be effective for the target. This is possible because the function of a drug is largely due to its structural properties and our method will generate structurally similar molecules to those in the sample set. The affinity of the generated molecules for a specific target can be assessed by performing a virtual docking between the generated molecule and the 3D structure of the target protein.

Finally, in chapter 7 we give our conclusions. This involves a summary of the contributions we have made along with our most significant results. This is followed by a selection of future work that we feel would be most valuable to pursue.

Chapter 2

Literature Review

In this chapter we will review literature relevant to the work that is described in this thesis, that is, generative models for graphs and their application to chemical structure. The first section discusses literature related to graph theory, while in the second section we will introduce literature from the field of chemoinformatics.

The section on graph theory begins by considering work on the construction of graph features; these allow graphs to be represented in a more compact and expressive form. We then discuss graph similarity and this leads naturally to the discussion of graph matching algorithms which may also be used to assess similarity. The field of graph segmentation is then introduced which considers how a graph may be decomposed into a set of subgraphs in the most logical manner. Finally, we discuss graph means and medians and then detail approaches to building generative models of graphs. Since the literature on generative models for graphs is so sparse, we also consider generative models on trees.

In the second section of this chapter we will begin our review of chemoinformatics by looking at ways of representing chemical structure. We then discuss methods of assessing chemical structure similarity. Related work on the virtual construction of chemical structures and chemical structure libraries is discussed next. Finally, we describe an important evaluation domain that will be employed later to test the effectiveness of generated chemical structures. This evaluation domain is known as docking and allows the simulation of molecule-protein interactions.

2.1 Graphs

2.1.1 Graph Features

The methods discussed in this section are concerned with how the best set of features can be extracted from a graph representation. These techniques are often necessary due to a phenomenon known as the *curse of dimensionality* [105]. It has been shown [15] that for a simple look up classifier (that associates a cell of the feature space with its classification) the number of training examples required to accurately train the system is exponential in the number of features. Therefore, the construction of a good set of features accurately describing the structure of a graph can be a useful tool.

The eigendecomposition of a matrix can be applied to a matrix representation of a graph to produce a *spectral representation*. This has produced the field of spectral graph theory[27] which has led to many different applications and approaches. If $\mathbf{X}$ is a matrix representation of a graph, then the *spectral decomposition* $\mathbf{X} = \mathbf{VEV}^T$ produces a set of eigenvalues $\mathbf{E}$ and eigenvectors $\mathbf{V}$. If the eigenvalues are ordered by magnitude they can be used as a feature set with the following two properties: firstly, they are invariant to vertex order and secondly, they express some of the structural information present in the graph. The set of eigenvalues is known as the *spectrum* of a graph.

Haemers and Spence [50] have enumerated all graphs on 11 vertices and less to determine the uniqueness of the spectrum associated with the adjacency, Laplacian and signless Laplacian matrices. If two graphs are structurally different but share the same spectrum then one is termed the *cospectral mate* of the other. The main result of the paper is to give the fraction of all graphs on n vertices that have cospectral mates. The results show that the signless Laplacian representation has the lowest fraction of graphs with cospectral mates, followed by the Laplacian and then the adjacency matrix.

Zhu and Wilson [128] have studied the stability and representational power of the eigenvalues obtained from various matrix representations of graphs. They use two methods to gauge the performance of each representation. The first experiment attempts to find if the relationship between the graph edit distance (section 2.1.2) and the distance between graph spectra is linear. The second test considers the clas-

sification accuracy of each representation method when some of the graph structure is perturbed.

Their results show that all the representations give spectral distances that follow the graph edit distances closely. However, the heat kernel and path length distribution representations are clearly the best. In the classification experiments, again the heat kernel and path length distribution methods perform the best. Next best are the various Laplacian representations and finally the adjacency matrix.

Luo et al [74] have described a number of methods of constructing feature vectors using the spectral decomposition of a graph's adjacency matrix. Of the unary features on eigenmodes they consider eigenvalues, eigenmode volume, eigenmode perimeter and spectral features based on the Cheeger constant. They also describe features based on pairwise measures between the eigenmodes such as the inter–mode adjacency matrix and the inter–mode distances. By ordering the features by eigenmode magnitude, invariance in vertex ordering can be achieved.

Wilson et al [123] suggest a method for creating graph features that are invariant to vertex order and can capture attributes that appear on the vertices and edges. The algorithm commences from the Laplacian matrix of a graph. An eigendecomposition is then performed on the Laplacian and the spectral matrix combining the eigenvalues and eigenvectors is produced (see equation 3.7).

The spectral matrix is only vertex invariant in the columns, providing the eigenvalues, and associated eigenvectors, are sorted by magnitude. To obtain invariance in the rows, symmetric polynomials are created. The use of symmetric polynomials is due to the fact that they are invariant in the order of their inputs. If each element of a column in the spectral matrix is used as the input for a symmetric polynomial, then the value of the symmetric polynomial will be the same regardless of vertex order. If n is the number of vertices then n^2 features are generated.

The values of the elements of each column can be reconstructed by finding the roots of all n^2 symmetric polynomials. Since the root order is undetermined, the elements of each column can only be recovered up to a permutation per column. Since the permutation of each column could be different, the graph cannot be easily reconstructed from the features.

Riesen et al [90] describe a method of embedding graphs in a vector space by using a dissimilarity measure. The algorithm proceeds from a set of graphs $\mathcal{S}$ from

which a subset of the graphs $\mathcal{P} \subset \mathcal{S}$ are selected as a graph prototype set. The graphs in $\mathcal{P}$ are designed to be representatives of the population of $\mathcal{S}$. By computing the graph edit distance between a graph and all prototypes, a dissimilarity vector can be constructed for a graph. Hence the vector is of dimension $|\mathcal{P}|$. Clearly the quality of the prototype set $\mathcal{P}$ is of great importance if a meaningful feature vector is to be produced. The authors suggest five methods of producing the prototype set ranging from taking a random selection to using a method similar to k-means clustering.

Caelli and Kosinov [22] suggest an approach that differs to those described above in that it embeds each vertex in a vector space rather than the graph as a whole. The reason behind this is to allow inexact graph matching which is discussed in section 2.1.3. Commencing from the adjacency matrix they compute the spectral decomposition and then project each column of the adjacency matrix into the eigenspace resulting from the decomposition. However, the features constructed for two different graphs result from two different eigendecompositions and therefore reside in different spaces. To solve this a renormalization step is applied. To handle graphs of different sizes the spectral information is truncated so it is the same dimensionality for both graphs. While the individual features are not so useful, the relative distances between the features and the trajectory of the features as a whole provide much information about the graph.

2.1.2 Graph Similarity Methods

Measuring the similarity of two graphs is a common problem in structural pattern recognition. For example, in a data retrieval problem where the data is structural in nature, we would like to be able to query a large database by computing the similarity between a query structure and the structures contained in the database. Later in this chapter we will discuss graph matching methods which provide a set of explicit correspondences between the vertices of two graphs. This correspondence information can be used to compute the similarity between two graphs by, for example, taking the *Frobenius norm* of the matrix representations of the aligned graphs. First we will introduce a simpler, more intuitive measure of graph similarity termed the *graph edit distance* and then describe similarity methods resulting from the feature descriptions discussed earlier in the chapter.

Graph edit distance[95, 19] is an approach that seeks to measure the similarity

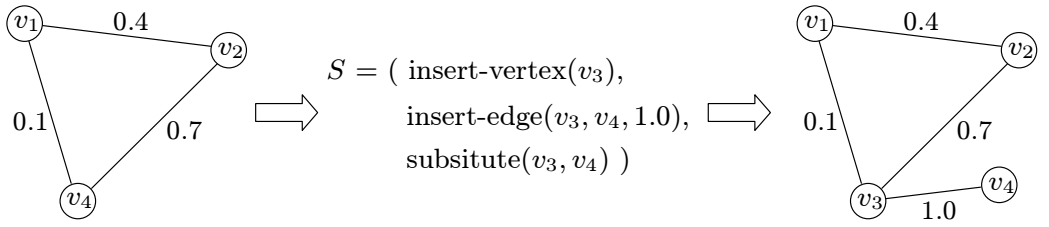


Figure 2.1: Transforming one graph into another by a sequence S of edit operations.

between two graphs by transforming one graph into the other. This transformation can be expressed by an *edit path* that describes a sequence of operations that may be applied to a graph to alter its structure. Such operations include the insertion and deletion of vertices and edges and the substitution of one vertex or edge for another. Each operation is assigned a cost and it is the sum of the costs of the operations in the edit path that determine the total cost for applying that edit path to a graph. The edit distance between two graphs is formally defined as the edit path that transforms one graph into the other such that the cost for the edit path is the minimum over all possible edit paths. However, the problem of finding the minimum cost edit path is NP-complete. Figure 2.1 shows an example of transforming one graph into another using an edit path.

The idea of using graph edit distance as a similarity measure appears frequently in the literature and methods of improving its usability have been developed. For example, Bunke [17] has established a relationship between the graph edit distance and the maximum common subgraph under a certain cost function. The *maximum common subgraph*, termed $mcs(G_1, G_2)$, is a subgraph of both G_1 and G_2 such that there is no subgraph possible with more vertices. The cost function in question is as follows: vertex insertions and deletions have cost 1, identical vertex and edge substitutions have cost 0 but different substitutions have cost ∞ and finally edge insertions or deletions have no cost.

While this cost function appears simple, the authors only consider complete graphs. This simplifies the proofs and results in no loss of generality. A standard graph is mapped onto a complete graph by setting the weights of edges that do not appear in the standard graph to zero in the complete graph. In this way the vertex insertions and deletions actually encode the edge insertions and deletions.

The graph edit distance $d(G_1, G_2)$ and the maximum common subgraph are

related as follows:

$$d(G_1, G_2) = |G_1| + |G_2| - 2|mcs(G_1, G_2)| \tag{2.1}$$

Bunke [18] has conducted an extensive analysis of cost functions and related the concept of graph edit distance to graph isomorphism and subgraph isomorphism under certain cost functions. In the tree domain, Torsello and Hancock [104] have shown how the tree edit distance may be estimated from the minimum description length of a tree set that they use in their generative model of tree structure (section 2.1.6).

Most methods that construct graph features produce feature vectors suitable for measuring graph similarity. For example, the feature vectors constructed using symmetric polynomials by Wilson et al [123] are suitable for measuring graph similarity. The feature vectors may be compared directly or projected into a low dimensional space using PCA to help visualize the similarities. The authors describe experiments involving graphs from the COIL data set which consists of sequences of images of various different objects. When graphs of different objects are considered, the resulting feature vectors do a good job of separating the objects in the vector space. In other words, similar graphs have similar feature vectors which in turn give high similarity. Furthermore, when a sequence of images are considered (i.e. when the camera moves very little between images) the resulting points in the vector space show a smooth trajectory.

2.1.3 Graph Matching Methods

In this section various methods to perform graph matching will be discussed. The general task to be solved when matching two graphs is to establish correspondences between the vertices of the two graphs. Of course, this is a hard task due to the arbitrary ordering of the vertices. Depending on the algorithm, a many-to-many vertex mapping may be found. However, most approaches solve the problem of finding a one to one mapping between the vertex sets ($\mathcal{V}_1$ and $\mathcal{V}_2$) of two graphs G_1 and G_2. The inclusion of the vertex $\emptyset$ in the vertex sets of both graphs ($\mathcal{V}_1 \cup \emptyset$ and $\mathcal{V}_2 \cup \emptyset$) allows vertices from one graph to be in correspondence with no vertices in the other graph, or alternatively, mapped to a dummy vertex. The quality of

the mapping is indicated by some criterion, on which the mapping function is then optimized.

The two main approaches to solve this problem are search based methods and nonlinear optimization methods. Search based methods construct a state space representing all the possible mappings between the two graphs, this space is then searched to find the mapping that minimizes some criterion. For example, Wang et al [113] describe the use of a genetic algorithm to search this space of mappings. However, due to the fact that the state space grows exponentially in the number of vertices of the graphs being matched, even advanced search techniques such as genetic algorithms are unable to provide mappings for large graphs in a reasonable time. For this reason nonlinear optimization approaches have attracted great interest. Some methods that fall under this field are those involving probabilistic relaxation[122], spectral methods [106, 22] and methods enforcing two-way assignment constraints [48, 107]. We now study these approaches in more detail below.

Umeyama [106] discuses an approach to solve the weighted graph matching problem in both the directed and undirected cases for graphs of the same size. The method uses an eigendecomposition on either the adjacency matrix, $\mathbf{A}$, (in the undirected case) or a Hermitian matrix computed from the adjacency matrix (in the directed case). If two graphs G_1 and G_2 are isomorphic then the permutation matrix $\mathbf{P}$ that solves $\mathbf{P}\mathbf{A}_1\mathbf{P}^T = \mathbf{A}_2$ provides the mapping. However, this equation is very difficult to solve exactly so Umeyama details a method that can provide a permutation matrix $\mathbf{P}'$ that nearly solves the problem such that error associated with the mapping $\mathbf{P}'$ is as close to zero as possible.

Umeyama shows that the permutation matrix $\mathbf{P}'$ that maximizes $tr(\mathbf{P}'^T\bar{\mathbf{V}}_1\bar{\mathbf{V}}_2^T)$ (where $\bar{\mathbf{V}}_1$ and $\bar{\mathbf{V}}_2$ refer to the absolute eigenvectors of graph G_1 and G_2 respectively) will be very close to the optimum permutation matrix when G_1 and G_2 are nearly isomorphic. Finding $\mathbf{P}'$ such that it maximizes the trace can be accomplished by the Hungarian method which runs in $O(n^3)$ time.

The permutation matrix produced by the method becomes progressively worse as the two graphs in question become more distant to each other. Performing hill-climbing or other optimization methods on the nearly optimal permutation matrix can improve the solution but the bottleneck on the quality of the solution is the

distance between the two graphs in question.

Scott and Longuet-Higgins [99] describe an algorithm that works on the distance between features in two related images and gives the correspondences between them. The correspondences are given by a *pairing* or *proximity* matrix that indicates the likelihood of two features being in correspondence. The algorithm commences by determining the Gaussian proximities of the distance between two features i and j, r_{ij}:

$$G_{ij} = \exp \frac{-r_{ij}^2}{2\sigma^2} \tag{2.2}$$

The σ variable provides a convenient method to study the difference between large and small displacements, in other words, local or global structure. This is similar to the role of the time variable in the heat kernel (section 3.2.4). The method continues by performing singular-value decomposition (SVD) on the matrix $\mathbf{G} = \mathbf{TDU}$. Next, $\mathbf{D}$ is conditioned by replacing all diagonal elements by 1, so the pairing matrix becomes $\mathbf{P} = \mathbf{TIU}$.

The element P_{ij} indicates how much the feature associated with row i is in correspondence with the feature associated with column j. If P_{ij} is the largest value in row i and column j then the two features represented by the row and column are in one-to-one correspondence.

Their results show that with a sufficiently large value of σ the algorithm can recover one-to-one correspondences between images affected by translation, sheer deformation, expansion or any combination of the three. However, the algorithm is not very successful when one image is a rotation of the other.

Wilson and Hancock [122] describe a method of graph matching based on a discrete relaxation labeling process. The structural correspondences between the two graphs are posed in terms of correspondences between substructures termed super–cliques. A super–clique consists of a central vertex and all other vertices at a distance of one edge. Associated with each super–clique to super–clique mapping is a probability and using a maximum a posteriori (MAP) criterion they aim to find the set of mappings such that the product of probabilities is maximized.

However, it is not always the case that one super–clique will map exactly onto

another due to vertices having different edge degrees. Therefore, a super–clique may be padded with extra vertices to accommodate size differences and this incurs a penalty in the probability of the mapping. Furthermore, all permutations of vertices in the perimeter of the super–clique must be considered and as such a probability is associated with mapping to each permutation. The algorithm proceeds by greedily assigning the mapping with highest probability to each vertex in order. Initially these mappings have high error associated with them and this is represented by an error term in the computation of the probabilities. As the algorithm iterates, this error term is reduced in a manner similar to simulated annealing until convergence occurs. The authors show the utility of their approach by matching aerial imagery of road networks and hedge structures.

Gold and Rangarajan [48] describe a graph matching method that uses two key ideas. These are iterative projective scaling and graduated non-convexity. Iterative projective scaling is used to enforce two-way assignment constraints, i.e. the requirement that the vertices in both graphs are equally constrained. The second technique used is graduated non-convexity which turns a matrix describing the (discrete) mappings into one containing continuous values. This provides protection against getting stuck in local minima by providing a control parameter that can move the solution from continuous (at the beginning of execution) to almost discrete (near the end of execution).

Since the graduated non-convexity technique only provides an almost discrete matrix at high values of the control parameter, it is necessary to perform some additional computation on the non-discrete match matrix to transform it into a discrete version. The authors use a simple heuristic that sets the maximum element in each column to 1 and all other values to 0. However, using this heuristic can result in a sub-optimal mapping since assignments are determined in a sequence rather than simultaneously. One algorithm that considers all the mappings simultaneously is the Hungarian algorithm [63].

Van Wyk and Van Wyk [107] describe an algorithm similar to that of Gold and Rangarajan. Instead of using iterative projective scaling to enforce two-way assignment constraints, they use a POCS (projections onto convex sets) method. The two convex sets used in this case are the set of matrices that satisfy assignment constraints on the columns and the set of matrices that satisfy assignment constraints on

the rows. Each of these corresponds to the assignment constraints in one direction and used together they can enforce two way constraints. The algorithm proceeds as that of Gold and Rangarajan's with the row and column normalizations being performed using the POCS approach.

Solving the one-to-one vertex correspondence problem is generally a good solution to matching graphs, however sometimes it can be more appropriate to produce a many-to-many matching. This gives far more flexibility and depending on graph structure allows for better matching. The graph features described in 2.1.1 by Caelli and Kosinov [22] can be used for such an approach. As described earlier the vertices of two graphs can be projected into a common subspace. With these two sets of points describing the vertices of each graph the correspondences can be established using a clustering step. This clustering, combined with the spectral representation's graph size invariant information, allows many to many correspondences to be established. The power of the approach comes from its ability to perform matching between substructures of different sizes in some parts of the graph, and in others to retain the possibility of one to one matching.

2.1.4 Segmenting Graphs

Perceptual grouping methods seek to split up a graph into k subgraphs such that by some measure of similarity, the similarity between vertices in different subgraphs is low but within the subgraph the similarity is high. This is usually accomplished by performing a *cut* on the graph. If a graph G is split into two subgraphs G_1 and G_2 with vertex sets $\mathcal{V}_1$ and $\mathcal{V}_2$ respectively, then the value of the cut is the sum of all edges that begin in one subgraph and end in the other.

$$\text{cut}(G_1, G_2) = \sum_{u \in \mathcal{V}_1, v \in \mathcal{V}_2} w_{\mathcal{E}}(u, v) \tag{2.3}$$

The weight function on the edges $w_{\mathcal{E}}$ is described in section 3.2. Various different cut criteria have been suggested that try to split the graph up in the most logical way possible. One of the first criteria proposed was Wu & Leahy's *Minimum Cut* [124]. This method aims to minimize the maximum cut across all subgraphs. However, in practice this approach does not perform particularly well, tending to cut

very small subgraphs without taking into account the global structure.

Shi & Malik's *Normalized cut* [100] solves this problem by computing cut cost as a fraction of the total edge connections to all vertices in the graph. The normalized cut is defined as follows:

$$\text{Ncut}(G_1, G_2) = \frac{\text{cut}(G_1, G_2)}{\text{assoc}(G_1, G)} + \frac{\text{cut}(G_1, G_2)}{\text{assoc}(G_2, G)} \tag{2.4}$$

The total connection weight of edges between vertices in the subgraph G_1 and the original graph G is given by $\text{assoc}(G, G_1)$, which has the same mathematical definition as $\text{cut}(G, G_1)$. Using the assoc value as normalization prevents cuts taking place where the number of vertices in the two sets is vastly different. The algorithm commences from a matrix describing the similarities between the vertices of the graph. Next an eigendecomposition is performed on the similarity matrix. The authors show that the eigenvector associated with the second smallest eigenvalue, also known as the Fiedler vector, will partition the graph according to the Ncut definition above. Finally, some measure is made to determine if the graph has been segmented enough (or if some fixed number of partitions is reached) and if not, then the process recursively repeats. The authors evaluate their method on both still images and motion sequences. The algorithm is shown to give good performance in both cases.

2.1.5 Median and Mean Graphs

The median graph of a graph set can serve as a useful tool and as such has received much attention in the literature. One use is as a model for representing a set of graphs in a compact notation. It can also serve as the starting point for generating new graphs or expressing the variation in a graph set. Given the mean graph, each graph in the set can be produced from the mean by applying some transformation. If the distribution of these transformations can be found, then it can be sampled from and used to generate new graphs.

Jiang et al [58] provide two different definitions for median graphs. The difference is in the space of graphs in which the median may be drawn from. The first is termed the generalized median graph and may be drawn from the space of all graphs. If we let $\mathcal{U}$ be the set of all graphs and $\mathcal{S} = \{S_1, S_2, ..., S_n\}$ be the set of

graphs we are attempting to find the median for, then the generalized median graph G_M is defined as follows:

$$G_M = \underset{S \in \mathcal{U}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{S}|} d(S, S_i) \tag{2.5}$$

where $d(S_1, S_2)$ is a distance measure. In the second definition, termed the set median, the median graph may only be drawn from the set of graphs in question. It is defined as such:

$$G_{M*} = \underset{S \in \mathcal{S}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{S}|} d(S, S_i) \tag{2.6}$$

Both definitions minimize the sum of distances to the graphs in the set $\mathcal{S}$. The generalized median is drawn from a larger set of potential medians and is therefore a better representation than the set median. However, the complexity of finding the generalized median is far greater than finding the set median.

We begin by discussing the simpler problem of computing the mean of a pair of graphs. Bunke and Günter [20] solve this problem by first finding the set of edit operations that transform one graph into the other. A cost is then assigned to each edit operation with the sequence of edit operations having total cost c. The mean graph may then be produced by applying the sub-sequence of edit operations with cost $\frac{c}{2}$. A weighted mean can be produced by applying the appropriate proportion of the edit path. Note that this weighted median may not be unique. The authors apply their approach to line drawing analysis and produce sequences of weighted means between the line drawings of two different letters.

Bunke et al [21] describe two methods for computing the generalized median graph of a set of graphs. The two methods they evaluate are a complete search using A* and a genetic algorithm. The two algorithms perform very differently, firstly, the complete search using A* has exponential worst case execution time and very high memory requirements. In practice both these properties mean that the algorithm is infeasible for anything other than very small graph sets. The genetic algorithm's performance varies according to the population initialization chosen, which can either be a random initialization or one based on the input graphs. However, in both cases the time complexity is far better than the A* method.

In an effort to reduce the size of the search space in which the generalized median graph can reside, Ferrer et al [39] describe a method to place an upper bound on the sum of distances (SOD) from the generalized median graph to all graphs in the set. This is done by establishing a relationship between the generalized median and the maximum common subgraph of the graph set. We first denote G_e as the empty graph and S_u as the union graph of the set. Then for any pairwise partitioning p of the set $\mathcal{S}$, where $SOD(p)$ is the SOD between the elements of the partition, then the bound on the SOD from the median G_M to all graphs in $\mathcal{S}$ must be as follows:

$$\max(SOD(p)) \leq SOD(G_M) \leq \min(\{SOD(G_e), SOD(S_u)\}) \tag{2.7}$$

They prove that the sum of distances to the maximum common subgraph of $\mathcal{S}$, $mcs(\mathcal{S})$, must be the upper bound on the sum of distances to the generalized median G_M. Thus the following inequality must hold true:

$$\max(SOD(p)) \leq SOD(G_M) \leq SOD(mcs(\mathcal{S})) \leq \min(\{SOD(G_e), SOD(S_u)\}) \tag{2.8}$$

Another result by Ferrer et al [40] places a bound on the number of vertices in the generalized median graph. This improves the bounds introduced by Jiang et al [58] which states, very generally, that the generalized median graph must have more than zero vertices but less than the sum of the number of vertices of all graphs in $\mathcal{S}$. The new bound relates the size of G_M to the size of the maximum common subgraph $mcs(\mathcal{S})$ and the *minimum common supergraph* $MCS(\mathcal{S})$ as such:

$$0 \leq |mcs(\mathcal{S})| \leq |G_M| \leq |MCS(\mathcal{S})| \leq \sum_{i=1}^{|\mathcal{S}|} |S_i| \tag{2.9}$$

The result of this is that more pruning in the A* search can be performed and more chromosomes in the generic algorithm can be disregarded, resulting in a lower search time. Recently, Ferrer et al [41] have used the above two results [39, 40] to detail a new genetic algorithm for finding the approximate generalized median graph. The new bounds are used to reject chromosomes representing median graphs that fall outside the space of where the median graph may lie. They present very encouraging results when their approach is applied to a data set representing web

pages as graphs.

Using the graph embedding method of Riesen et al [90] that was described in section 2.1.1, another two methods for generalized median computation have been developed. The first searches for the generalized median graph in a reduced space[42] while the second directly computes an approximation of the generalized median graph[43]. We will discuss the search based approach first.

The algorithm of Riesen et al is used to produce a vector describing each graph in the set $\mathcal{S}$. However, instead of selecting a subset of the graphs in $\mathcal{S}$ to form the prototype set, all graphs in $\mathcal{S}$ are used as prototypes. This means that the dissimilarity matrix is $|\mathcal{S}|$ by $|\mathcal{S}|$ in size and the feature vector for each graph records the dissimilarity to all graphs in the set $\mathcal{S}$.

The dissimilarities are calculated using a restricted graph edit distance as discussed in section 2.1.2 where deletions and insertions of vertices have a cost of 1, deletions and insertions of edges have a cost of 0 and vertex and edge substitutions have a cost of 0 of ∞ depending on whether the substitution is identical or not. This means that the Manhattan distance can be used to compute the mean of the feature vectors. However, although this mean feature vector represents the median graph, it can not be directly translated back into a graph and must instead be used to limit the search space.

Although the search space should be exponential in the sum of the vertices of the graphs in $\mathcal{S}$, it has been shown (Section 5.3.2 [36]) that under the cost function mentioned above, the search space can be reduced and is composed of the vertex induced subgraphs of the minimum common supergraph of $\mathcal{S}$. The search space of the generalized median graph can be visualized as a rhombus with the minimum common supergraph at one end and the empty graph at the other. The graphs from the set $\mathcal{S}$ must all fall within this rhombus.

The mean feature vector describes the dissimilarly between the median graph and every graph in $\mathcal{S}$. Hence, if the dissimilarity between the generalized median graph and a graph $S_1 \in \mathcal{S}$ is a then only graphs with an edit distance of a from S_1 must be searched. The graph in $\mathcal{S}$ that places the maximum restriction on the search space (smallest dissimilarity) is selected to restrict the rhombus search space. Thus the graph that has the minimum SOD to the graphs in $\mathcal{S}$ which is selected from the reduced search space must be the generalized median. The authors describe good

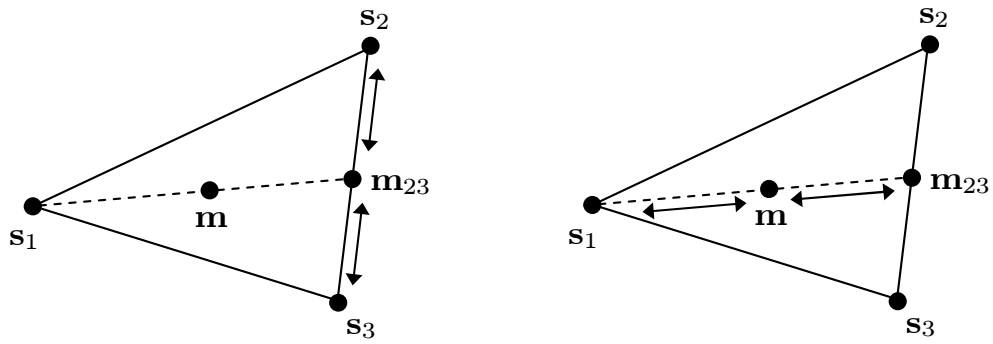


Figure 2.2: The triangulation sequence from the approximate generalized median algorithm of Ferrer et al [43].

results, in terms of time complexity, when the approach is applied to line drawings of letters.

The algorithm that computes the approximate generalized median graph proceeds as described above until the search stage. Instead of a search, the approximate generalized median is recovered from the vector space using the approach described earlier[20] to compute the weighted mean of a pair of graphs. The algorithm proceeds by finding the three graphs $\{S_1, S_2, S_3\} \in \mathcal{S}$ that are closest in the vector space to the mean vector[1]. A triangulation procedure is then performed on the points representing these graphs which we will denote $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$. First, the mean point $\mathbf{m}$ of $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$ is found. A line is projected, from $\mathbf{s}_1$ through $\mathbf{m}$ intersecting the line between $\mathbf{s}_2$ and $\mathbf{s}_3$. Let us call the point of intersection $\mathbf{m}_{23}$. Using the distances between the intersection point $\mathbf{m}_{23}$ and the graph points $\mathbf{s}_2$ and $\mathbf{s}_3$, a weighted mean graph M_{23} representing the graph at $\mathbf{m}_{23}$ can be computed (see figure 2.2 left pane). Finally, the distances between $\mathbf{m}_{23}$, $\mathbf{m}$ and $\mathbf{s}_1$ are used to compute the weighted mean graph between M_{23} and S_1 (see figure 2.2 right pane). This graph will be used as the approximate generalized median.

The authors evaluate the quality of the approximate generalized median graphs by comparing their SOD to all graphs in the set with the SOD from the set median to all graphs in the set. It is not possible to compare the true generalized median graph with the computed approximation due to the size of the data sets, which make

[1]Since it is not necessary for the mean vector to represent the dissimilarities to graphs in the set as integers (as in the search based approach), the Euclidean mean is used instead of the Manhattan mean.

computation of the true generalized median graph impossible. They show that in all cases the quality of the approximate generalized median is better than the set median.

Ferrer et al [37] extend the definitions of the generalized median and set median to the spectral domain. As with the non-spectral definition, the set spectral median must be a graph chosen from the set while the generalized spectral median may be taken from the set of all graphs. The spectral matrix chosen is the one that maximizes the sum of the correlations between it and the spectral matrices of all other graphs in the set.

Due to the restrictive computational complexity of computing the spectral generalized median the authors suggest two approximate methods, an incremental method and a hierarchical method. They provide details for the incremental method which we describe briefly here. Given a sequence of graphs $(S_1, S_2, ..., S_n)$, the first graph S_1 is selected and its spectral representation is computed. This results in the pair $(S_1, \mathbf{V}_1) = (G_M, \mathbf{V}_M)$ which represents the initial generalized median graph. Next, they select another graph S_2 and compute $\mathbf{V}_2$. Using Umeyama's algorithm [106] they find a labeling between S_2 and G_M (using spectral matrices $\mathbf{V}_2$ and $\mathbf{V}_M$) and use this to perform an update to the median graph G_M. An eigendecomposition of the resulting median graph is performed which produces a new spectral matrix $\mathbf{V}_M$. The algorithm iterates updating the median with the remainder of the graph sequence. Since a sequence is used it is possible to produce different medians depending on the order of the graphs in the sequence.

Ferrer et al [38] compare their approach described in [37] of computing the median spectral graph using incremental updates to the work we describe in this thesis in chapter 3. We propose the computation of the median spectral matrix directly through a single averaging operation rather than a series of incremental adjustments. Ferrer et al show that both methods perform approximately the same except under severe noise in which the incremental method outperforms the direct averaging method.

Traditionally, using the generalized median of a graph set for pattern recognition tasks has been limited to simple examples due to the high computational complexity of constructing it. However, with the recent advances detailed in [42, 43, 41], its usage for real world data sets is becoming feasible.

2.1.6 Generative Models

We now turn to discuss work on constructing generative models for graphs. While there is much literature on generative models in the vectorial domain there is little work in the graph based domain. Nevertheless, we survey the few pieces of such work.

Luo et al [75] detail a method for constructing a generative model over a set of graphs. By vectorizing the adjacency matrix of each sample graph they construct a vectorial description such that statistical techniques applicable to vectorial data may be used. Before this can occur however, the graphs in the sample set are aligned using the algorithm of Luo and Hancock [73]. Differences in graph sizes are dealt with by padding the smaller graphs with dummy vertices.

With the aligned padded adjacency matrix $\mathbf{A}_k$ of each sample graph S_k to hand they proceed to vectorize each matrix.

$$\mathbf{a}_k = \mathrm{vec}(\mathbf{A}_k) \tag{2.10}$$

They construct a linear deformable model over the resulting vectors which can be used to express the structural variations present in the sample set of graphs. In practice this means computing the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of the vector set and then performing an eigendecomposition on the covariance matrix. This results in the diagonal matrix of ordered eigenvalues $\boldsymbol{\Lambda}$ and a matrix of ordered eigenvectors $\boldsymbol{\Phi}$. The ordered eigenvector matrix $\boldsymbol{\Phi}$ gives the principle components of variance in the data and the set of eigenvalues determines the degree of variance associated with each component. This allows a vectorial description of a sample graph to be represented as a mean summed with a vector determining the specific way that graph structurally varies. This vector is projected into the principle components of the distribution. This is illustrated by the following equation:

$$\hat{\mathbf{a}}_k = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{b} \tag{2.11}$$

where $\mathbf{b}$ is a parameter vector describing the degree of variation experienced by a graph S_k in the direction of each principle component. The authors proceed to use their model to explore the structural variation present in a set of graphs representing faces.

A similar generative model is described by Xiao & Hancock [126]. The method constructs a point distribution model representing the structural variation in a set of graphs. The process commences from the heat-kernel representation of each graph. To obtain a coordinate matrix $\mathbf{Y}_k$ for each graph S_k, the Young-Householder decomposition $\mathbf{H} = \mathbf{Y}_k^T\mathbf{Y}_k$ is performed on the heat-kernel. Given that $\mathbf{E}_k$ and $\mathbf{V}_k$ result from the eigendecomposition of the normalized Laplacian of a graph S_k, the coordinate matrix is computed as follows.

$$\mathbf{Y}_k = \exp\left[-\frac{1}{2}\mathbf{E}_k t\right]\mathbf{V}_k^T \tag{2.12}$$

To handle graphs of different sizes the coordinate matrices are truncated. The truncated coordinate matrices are aligned using the method of Scott and Longuet-Higgins [99] where the largest graph is selected as the reference. As in the previous work, PCA is performed on the aligned truncated coordinate matrices to construct the model.

Experiments are performed on object sets from the COIL database. The trajectory of the coordinates for an object sequence appear to vary smoothly throughout the vector space. In addition, they assess the compactness of their model by measuring the error involved in recovering a graph using a reduced set of components of the model. For small values of the time parameter in the heat kernel they achieve low error rates with limited components.

Torsello [102] describes a generative model for graph structure based on importance sampling. By making the simplifying assumption that the observation of each vertex and each edge is independent from all other vertices and edges, the existence of each vertex is modeled as a Bernoulli trial. However, examples from the model can only be generated by removing a vertex from the model graph. This means that a vertex is required in the model for every vertex in the sample graphs, whether that vertex is key to the structure represented in the set or just noise.

Nevertheless, this provides a generative model of the graph set but the correspondences between each sample graph and the model graph are unknown. To solve this an importance sampling method is introduced to establish the correspondences. Using the derivative of the likelihood of the model, a maximum likelihood estimation of the model parameters is performed. The success of the model is gauged using Delaunay graphs constructed from images of scenes and shock graphs constructed

from objects.

Torsello and Dowe [103] extend the model and correspondence step using an EM like approach. As above they model the existence of each vertex as a Bernoulli trial but to relax the requirement that every vertex must be represented in the model graph, they introduce a two part model. This consists of a structural part of the model, that represents the core structural variations in the set, and a noise model that allows the generation of vertices that do not correspond to any vertex in the core model graph. They also improve on the previous approach by allowing the inclusion of attributes on the vertices and edges.

To learn the model, vertex correspondences between a sample graph and the model are separated from the model parameters which facilitates a two step, EM like, estimation approach. On each step the correspondences between each sample graph and the model are updated and then the model parameters are re-estimated given the updated correspondences. The model parameters are determined using a minimum message length criterion which states that the best model describing the data is the one resulting in the shortest length description of both the model and the encodings of the data given the model. The authors apply their approach to classifying shock graphs of shapes and show that using their model with attributes on the sample graphs results in good performance.

While it is possible to sample from the models proposed by Torsello [102] and Torsello and Dowe [103], the assumption of vertex and edge independence results in no co-occurrences of edges or vertices being considered. These co-occurrences are the ingredients of describing structural variation successfully and therefore these models would be unlikely to generate examples drawn from the input distribution.

Torsello and Hancock [104] propose a generative model for tree structure that makes use of the method similar to the minimum message length criterion detailed above. By constructing a tree-union describing all trees that belong to a certain class, a mixture model of tree-unions can be constructed that can be used for classification and clustering. They associate a probability distribution with each tree-union that explains the structural variations present in that class. This distribution, along with the tree-union and correspondences from samples to their class union, are all learnt in an optimization framework. This optimization process works by merging tree-unions such that the length of the description of the trees from the (now com-

bined) classes is minimized. By starting with a separate model for every sample tree, similar models can be merged hierarchically until there are no further merges possible that reduce the description length. The authors evaluate their method on shock graphs and show that it is capable of handling structural noise effectively.

2.2 Chemoinformatics

We will now describe literature in the field of *chemoinformatics*[69, 51] which we will make use of when we apply our generative models to chemical structure. Chemoinformatics is a relatively new field which studies the application of computational techniques to chemistry. The main topics discussed in this section are the representation of molecules, assessing molecule similarities, generation of chemical structure and docking of molecule-protein complexes.

2.2.1 Molecule Representations

It is difficult to obtain a complete representation of a molecule by using anything less than a full 3D description of the location of the atoms, the type of bonds between atoms and the molecular surface resulting from the atoms and bonds. This is due to many small molecules being steroisomers[2] that, when described in only 2D, lead to a number of possible 3D configurations.

Nevertheless, 2D representations of molecules are used extensively, with the line notations we are used to seeing being the most common. It is clear that 2D line notations of a molecule can be easily mapped onto a graph by expressing atoms as vertices and bonds as edges. This mapping, along with other applications of graph theory to chemistry, are discussed by Balaban [11]. Basak et al [14] describe calculating 90 graph theoretic properties for a set of molecules. By performing PCA on the properties they clustered the molecules and showed that the groupings reflected intuitive ideas of chemical similarity.

With a mapping from chemical structure to graphs to hand, standard representations such as the adjacency matrix can be used to represent molecules. Dugundji and Ugi [34] describe a matrix representation known as the bond-electron matrix

[2]Molecules which are steroisomers may exist in a number of different 3D structures, and most small, drug-like molecules have this property.

that has bond values in the off-diagonal elements and the number of free valance electrons of an atom in the associated diagonal element. This matrix has a number of useful properties such as the easy computation of the number of valence electrons present in a molecule. This, combined with knowledge of the standard number of valance electrons an atom possess, can be used to determine if an atom carries a charge. More recently, Wilson [121] has used Hermitian matrices to represent properties on graphs such as atomic number and bond strength.

Despite the elegance of matrix representations, connection tables are the preferred method of molecular representation since any number of additional attributes on an atom, a bond or the molecule can be easily expressed. Connection tables contains two lists, one detailing the atoms and a second describing the bonds between them. Most chemistry toolkits (such as *Open Babel* [4] and the *Open Eye Chemistry Toolkit* [5]) support molecules entered in connection table format, with the most wide spread of these being the SDF and MOL2 data formats. Connection table notations can also express 3D structure by associating a coordinate attribute with each atom.

Molecules may also be described by linear notations, the most popular of these being SMILES (Simplified Molecular Input Line Entry System). Devised by Weininger [114], the notation allows the description of chemical structures by character strings. In the notation: atoms are represented by their atomic label, neighboring atoms are placed adjacent in the string, branches are indicated by parentheses and ring systems are described by labeling the two atoms completing the ring with digits. This is the notation of choice in the *Daylight Toolkit* [7]. In some cases there are multiple SMILES strings that describe the same molecule. Therefore, it can be useful to have a method of computing a canonical SMILES string. However, there is no well documented way to do this in the literature and various approaches have been developed independently. A newer form of SMILES called *InChI* [8] provides a unique description of a molecule but is far less human-readable.

Markush structures [12] are another type of molecular representation and were designed for identifying chemical structures which are under patent. Often a patent covers a wide range of different molecules and this is indicated by identifying a core part of the molecule (which stays constant) with a number of attachment points and a set of rules governing which chemical structures may be attached to each

attachment point. Markush structures have many uses beyond patent identification and Barnard et al [13] have used them for clustering large libraries of molecules.

The *fingerprint*[1] of a molecule, made popular by the Daylight Toolkit, is a method of describing a molecule through a set of structural keys. Each key indicates whether a particular substructure is present in the molecule. Fingerprints are normally represented as bit vectors where each bit represents the presence of the corresponding substructure. This representation is very popular for computing similarities and performing substructure searching as discussed in the next section. However, due to a limited number of structural keys being available, it is possible that two molecules will map to the same fingerprint. Therefore, any approach using fingerprints must often be combined with a more exhaustive approach.

Finally, we consider very simple molecular descriptors that apply to the molecule as a whole. For example, molecular weight can be used as a very simple molecular descriptor. Another is hydrophobicity which determines the degree that a molecule is repelled from water. This has an important affect on the transport and activity of drugs in humans and as such makes an interesting descriptor of molecules to be used as drugs.

One would think that such simple descriptors would be of little use, but sets of rules for determining the applicability of a molecule as a drug often make use of these descriptors. For example, Lipinski's rule of five [72] is a rule of thumb to evaluate druglikeness and is described using simple descriptors. It has been shown by Xue et al [127] that combining short fingerprints with whole molecule descriptors results in a better description of a molecule than a long fingerprint used by itself.

2.2.2 Computing Molecule Similarities

Efficient methods of computing molecule similarities are of great interest in chemoinformatics. This is due to the fact that structurally similar molecules often have similar properties. This has been observed by Johnson and Maggiora [59] as the *similar property principle* and by Patterson et al [85] as the *neighborhood principle*. Therefore, if a molecule is known to have certain therapeutic effects, a good place to start the search for other molecules with the same effect would be by finding molecules that are structurally similar to the original. This idea of similarity searching in databases of molecules was introduced by Carhart et al [25] and Willet

et al [120].

The search can proceed in one of two ways depending on what information is available beforehand. As described previously, if a whole molecule is known that produces the desired effect, then a database can be searched for similar molecules. This approach is known as *full structure search*. Although graph matching algorithms could be used to establish a set of correspondences between the two molecules, in practice this is rarely required and approximate methods suffice.

On the other hand, if the mechanism of action is known, i.e. the fragment from the molecule that gives rise to the effect, then a *substructure search* (Chapter 5 [69]) can be performed to find molecules which also contain that specific fragment. This type of search often consists of two phases. The first phase, known as *virtual screening*, uses a fast substructure search method usually based on fingerprints to rapidly eliminate molecules that do not contain the required substructure. However, due to the non-uniqueness of fingerprints, the method might return false positives. Therefore, a full subgraph isomorphism algorithm is used to check the remaining molecules for the substructure.

In both cases fingerprints are often the molecular descriptor of choice. The reason behind this is the idea that more similar molecules will have more individual substructures in common. Since fingerprints are naturally represented as bit vectors, the comparison of two is very fast using a measure like the *Tanimoto coefficient*. This distance measure was used by Willet et al [120] to rank the results that are returned from a chemical database searching system. Various other measures have been devised for assessing fingerprint similarity and these are discussed by Willet et al [119]. More recently, Salim et al [94] have shown how data fusion techniques can be used to combine a number of different similarity measures. However, they also point out that no combination can consistently outperform the Tanimoto coefficient.

2.2.3 Generation of Chemical Structure

The term *chemical space*[83, 71] has gained popularity over the past few years as a way of describing all possible chemical structures. Although chemical space is vast, there are specific regions of it that are desirable to explore in depth; one of these regions is the space of druglike molecules. It is currently unfeasible to enumerate such regions, however, approximately enumerating subsets of this region is possible

and indeed useful for discovering new drug leads.

Programs designed to generate chemical structure fall under three broad categories. The first, termed *CASE* (computer assisted structure elucidation) [82], consider the problem of finding a valid 3D conformation of a molecule given some experimental data. The second, known as *de novo* methods[97] attempt to generate active ligands given information about the structure of the target protein's active site. While the third, *library design* methods, produce a set of molecules that attempt to cover the diversity present in a small area of chemical space.

The structure elucidation of an unknown molecule is a long standing problem in chemistry. Typically some data is available describing the 3D structure of the molecule. An example of this type of experiment from the field of molecular spectroscopy is Carbon-13 NMR.

Early attempts to solve it by Carhart et al [23] resulted in a program CONGEN being produced and later GEONA [24]. Both these programs are capable of enumerating the possible structures of a molecule given its molecular formula by assembling fragments or atoms. However, rather than a systematic exploration of the required region of chemical space, they use a more heuristic method. A more systematic exploration of chemical space is given by Funatsu et al [45] in their structure elucidation program CHEMICS. CHEMICS uses data from molecule spectrums to construct an initial set of fragments, this set is reduced using the molecular formula and user entered constraints. CHEMICS can also avoid the possibility of generating overlapping fragments. Contreras et al [30] use a tree notation of molecules to define a base tree which is a representative of the space of molecules that are to be explored. They then explore the structure present by performing a depth first search.

Christie and Munk [26] describe a system (COCOA) that uses the reverse of structure generation; structure reduction. Commencing from a hyperstructure representation of a molecule, bonds are successively removed according to data collected from spectral experiments.

Meiler and Will [80] describe a genetic algorithm for structure elucidation from a molecular formula and C13 NMR data. Each chromosome represents a possible structure for the molecule and the initial generation is constructed by randomly selecting possible structures from the molecular formula. The fitness of a chromosome is assessed by checking its compatibility with the NMR data. The population

of chromosomes is then evolved using the standard methods of genetic search algorithms.

Porquet et al [87] describe a method of randomly generating 3D molecular structure by combining fragments from a fragment database defined by the user. The authors use a "self-generation algorithm" that builds the 3D structure step by step by attaching fragments to the growing molecule. The attachments are done according to covalent binding where fragments are attached to unused valencies in the growing molecule. The authors establish strong mathematical links between the fragment database and the set of generated structures. The role of the approach is to understand the size, shape and mass characteristics of the generated structures. This piece of work sits on the boarder line between CASE methods and library design.

De novo methods for ligand design have been in development for more than 15 years and have their roots in computer assisted chemical elucidation systems. Given the structure of an active site in a protein, the goal is to build chemically viable molecules that are successful at interacting with that protein. Initially, methods only implemented structural constrains but have since been extended to consider the synthesis route and druglikeness of the generated molecules.

Early approaches built the ligands inside the active site starting from *seed* atoms. By joining either atoms (GenStar[92]) or fragments (GroupBuild[93]) to the seeds a complete ligand could be produced that was complementary to the shape of the active site. LigBuilder[112] used a similar approach but the process was controlled by a genetic algorithm. GROW[81] uses an incremental approach but avoids some of the complications of fragment assembly by only joining amino acids. BUILDER[91] searches a database of structures which are then superimposed in the active site. Ligands are found by tracing paths through the superpositions. SPROUT[47] uses a two step approach, first a skeleton is built in the active site and then atoms are substituted into the skeleton to produce a ligand.

While all the above approaches construct the ligands with 3D constraints in place, 2D systems have also been proposed that make use of conversion algorithms to produce low energy 3D conformations from 2D descriptions during the process. In DBMAKER[52] and BOOMSLANG[31] generated molecules are described using SMILES strings and then converted into 3D structures using CONCORD. MOLMAKER[28] uses a graph theoretic approach to enumerate all graphs

with certain sets of vertex degrees. These graphs are then attributed with atom and edge labels and thus transformed into molecules.

More recently, attention has been paid to the requirement that *de novo* methods should produce ligands that have a realistic synthesis route. For example, DREAM++[77] produces new ligands through user specified chemical reactions which are then checked for compatibility with the active site. TOPAS[98] uses a large fragment library derived from current drugs and a restricted set of reactions to produce new structures. A template structure is presented to the system from which a number of new molecules are produced. An evolutionary algorithm is employed such that the best molecule from the current generation becomes the new template structure for the next generation. SYNOPSIS[110] also employs a cyclic generation procedure where new molecules are added to a pool that is used to construct later molecules.

The third requirement of druglikeness is generally only assessed in *de novo* methods through a simple filtering step based on a rule of thumb (e.g. Lipinski's "Rule of Five"[72]). However, a recent approach by Kutchukian et al[66] allows molecules with realistic synthesis routes to be assembled according to the class of molecules the system is trained on, such as drugs or natural products.

Targeted library design is the task of producing a set of molecules that cover the diversity present in an area of chemical space. Ideally, the library should express all the points of chemical variation present in the required area of chemical space in the minimum number of molecules possible. This is due to the costs of library synthesis. When little information is known about the target, a very diverse library of molecules is designed and conversely, when much structural information is known then a focused library of molecules is used. In both cases the representative power of the library is of key importance, in other words, the amount of chemical space covered by the molecules. This can be determined by *diversity analysis*[118].

Targeted libraries are usually produced by a molecule selection approach, for example performing a substructure search on a large set of molecules. However, methods that generate new chemical structures are also used. Leach et al [68] describe such a method of library enumeration using a reaction transform method. A reaction transform method works by reacting a set of user defined fragments according to a set of user defined rules to produce new chemical structures. The Daylight

reaction toolkit[2] is used to perform the reactions.

The authors also suggest a fragment marking approach to generating chemical structures. This works by segmenting the input molecules into core structures with additional groups attached (much like how Markush structures are described). New molecules can then be generated by specifying rules as to which fragments may be attached to which connection point on a core structure. While this is significantly faster than the reaction-transform approach, problems can arise where the structures must be corrected by hand.

2.2.4 Docking and Scoring

Docking [96] is the study of protein-ligand interactions using computational methods. Given an active site of a protein and a small molecule (a *ligand*), docking can explain how well the ligand binds to the active site. This is performed by exploring the conformational space of the ligand (and in some cases the protein) to find the optimal pose of a ligand in an active site. The success of the docking is computed by finding the energy of the binding and in turn this is related to the user using a scoring function.

Virtual high-throughput screening [76] (vHTS) is perhaps the most common use of docking methods. Traditional high-throughput screening [53] involves testing a large number of molecules against a specific target in the hope of finding a molecule that is active. In contrast, Virtual HTS uses computational models to find active molecules by testing a large number of molecules *in silico*. This screening can be quite simple, for example using fingerprint based screening methods described earlier, or more complex by using 3D molecule descriptions or performing a docking simulation. In all cases, the goal is to reduce the number of molecules that must be tested *in vitro*. For example, the molecule libraries produced using methods from the previous section might be subjected to vHTS before undergoing synthesis and subsequent tests in a laboratory.

A second application of docking methods is to predict the binding pose of a known active ligand. This provides insight to the specific mechanism that results in a ligand being active. The result of these two applications is that docking is used in some capacity by nearly all pharmaceutical companies.

The state of the art of docking technology as discussed by Leach et al [70]

shows that while the effectiveness of docking has increased considerably in the last 20 years, there are still significant improvements required before computational models will accurately reflect the true biochemistry taking place. The result of this, as discussed in detail by Kontoyianni et al [61], is that certain docking programs and scoring functions work well for specific targets. As of yet there is no universal way of predicting binding poses of ligands and the energy associated with the poses.

Nevertheless, many different programs to perform docking have been developed and evaluated. Rarey et al [88] describe an algorithm called *FlexX* that incrementally builds up the ligand using a greedy strategy to find the optimal pose. Kuntz et al [65] have designed an algorithm (*DOCK*) that uses shape-based approaches to compute the optimal pose. The approach has been extensively supported and is now in its 6th release. By combining Monte Carlo simulation with a shape comparison filter, Venkatachalam et al [108] have designed *LigandFit* to perform docking. Openeye chemical software have designed a program called *Fred* [3] that provides exhaustive searching of possible ligand poses.

These approaches explore the conformational space of the ligand but not the protein. As discovered by Koshland [62] in 1958, the simple *lock and key* idea of ligand-protein interaction where only the ligand undergoes conformational changes does not completely explain the situation. Instead, Koshland suggested the *induced fit model* that allows the active site to undergo structural changes as well. This is modeled in part by the *GOLD* docking program by Jones et al [60] that explores partial flexibility of the protein. A genetic algorithm is employed to search the space of possible protein-ligand poses.

The chosen scoring function is of great importance if the true value of a ligand binding pose is to be accurately described to the user. Bissantz et al [16] analyze seven different scoring functions over three different programs (Dock, FlexX and Gold). They report that using consensus scoring functions (combinations of single score functions) significantly improves the quality of scoring. McGaughey et al [79] compare a variety of screening methods using both 2D and 3D descriptions of ligands and a variety of docking methods. Interestingly they show that the 2D ligand descriptions can sometimes give a better indication of activity than more complex 3D descriptions and docking methods. However, it is generally agreed in the field that comparative studies of these kinds are very difficult due to bias arising in the

data set chosen.

Chapter 3

Mixing Spectral Representations of Graphs

3.1 Introduction

In this chapter we propose a method of computing the median graph of a set through averaging spectral matrices. The median graph (section 2.1.5) has attracted much attention recently as a method of constructing a representative for a set. Jiang et al [58] were the first to formalize the idea of median graphs and gave two definitions for median graphs. Both definitions state that the median graph should be the graph with the minimum sum of distances (SOD) to all graphs in the set, however they differ in the space from which the median graph may be drawn. The set median must be one of the graphs in the set while the generalized median can be any graph. The distances used to calculate the SOD for the median graph are usually expressed as graph edit distances (section 2.1.2) but different distance measures may be used.

The computational complexity of finding the generalized median is significantly higher than that of the set median. Accordingly, the generalized median gives a more accurate description of the graph set. Therefore, much work has gone into trying to find approximate solutions to the generalized median problem. For example, Bunke et al [21] describe a genetic algorithm to compute the generalized median graph. Following a series of theoretical results [39, 40] that place bounds on the median graph, a new genetic algorithm [41] has been proposed that can prune much more of the search space.

Using the dissimilarity graph embedding method of Riesen et al [90], another two methods for generalized graph computation have been developed. In the first, Ferrer et al [42] describe a method of searching for the generalized median graph in a reduced space. While in the second, Ferrer et al [43] use a triangulation procedure in the embedding space to construct an approximation of the generalized median.

The idea of applying spectral methods to the computation of median graphs has been used before. Ferrer et al [37] extended the definitions of the generalized median and set median to the spectral domain and gave an algorithm to compute the spectral median graph. The algorithm proceeds by using graphs from the set to perform incremental updates to the spectral median graph. However, due to the incremental nature of the algorithm the graph order used determines the final spectral median graph. Therefore, using this algorithm, it is possible to construct different spectral median graphs for the same set. See section 2.1.5 of the literature review for a full discussion of this approach.

In our approach we propose the direct averaging or mixing of the spectral matrices produced from graphs in the set. Since it is known that the different eigenmodes correspond to different levels of structure present in the graph, we hope that by mixing the spectral matrices we will be able to combine different structures from graphs. For example, it should be possible to combine the local structure of one graph with the global structure of another. If this is successful not only will we be able to generate new graphs in a rudimentary way but we will also show that this type of representation could be suitable for forming a generative model.

However, a spectral representation describing a mixed or averaged graph does not directly give us the graph it describes. This is because it is very unlikely that the mixed spectral matrices will display the properties we require from them (such as orthogonal eigenvectors). Therefore, we must define a reconstruction step that conditions these matrices and in effect projects our current description of the averaged graph onto the nearest correct graph.

The outline of the chapter is as follows; In section 3.2 we describe various matrix representations of graphs, which are key to the reconstruction process. In section 3.3.1, we describe how we align the spectral representations and mix them together. In section 3.3.2, the reconstruction process is detailed. In section 3.4 we provide some experimental analysis. Finally, in section 3.5 we draw our conclusions. A

table of the symbols used in this chapter is given in section 3.6 on page 61.

3.2 Graph Representation

A graph is a relational structure that consists of vertices (the objects being considered) and edges (the relations between the objects). For example, the World Wide Web is a large relational graph where the pages are vertices and the hyperlinks are edges. Since a relational structure is very general and expressive, graphs are used for storing and representing data in many different applications. The converse of this is that they are difficult to represent in a canonical form, generally an ordering over the vertices must be established or some form of vertex order invariant representation must be computed.

Formally, a graph G is defined to be a tuple $(\mathcal{V}, \mathcal{E}, w_{\mathcal{V}}, w_{\mathcal{E}})$ which has vertex set $\mathcal{V} = \{v_1, v_2, ..., v_n\}$, edge set $\mathcal{E} = \{e_1, e_2, ..., e_m\} \subset \mathcal{V} \times \mathcal{V}$, weight function on the vertices $w_{\mathcal{V}} : \mathcal{V} \to [0, 1]$ and weight function on the edges $w_{\mathcal{E}} : \mathcal{E} \to (0, 1]$. There are varying degrees of complexity that a graph can exhibit.

- **Simple Graph:** The most basic type is a graph with no weights on the vertices or on the edges, no direction associated with an edge and no edges that connect a vertex to itself. An edge is either present or not present. For this reason $w_{\mathcal{V}}$ maps all vertices to 0 and the weight function $w_{\mathcal{E}}$ maps any edges that exist to a value of 1. Furthermore, we represent the non-directed property by ensuring that if $(v_i, v_j) \in \mathcal{E}$ then $(v_j, v_i) \in \mathcal{E}$ is also true. Edges beginning and ending at the same vertex are forbidden by ensuring that $(v_i, v_i) \notin \mathcal{E}$.

- **Weighted Graph:** A weighted graph retains the non-directed and no self relating vertices properties from the previous type. However, the weight functions $w_{\mathcal{V}}$ and $w_{\mathcal{E}}$ are now free to take values in the range $[0, 1]$ and $(0, 1]$ respectively.

- **Weighted Directed Graph:** A weighted directed graph captures direction on the relationships between objects and thus (v_i, v_j) may appear in $\mathcal{E}$ without (v_j, v_i) appearing. The restriction that $(v_i, v_i) \notin \mathcal{E}$ remains.

There are of course other types of graphs possible like a general graph (with no restrictions), a weighted graph that only makes use of one of the weight functions or

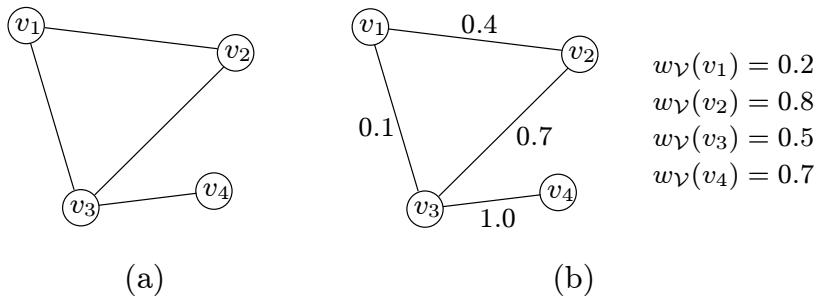


Figure 3.1: Two example graphs: (a) depicts a simple graph while (b) depicts a weighted graph with edge weights drawn on the graph and vertex weights shown to the right.

a non-weighted directed graph. However, we will only concern ourselves with the first two types, that is simple graphs and weighted graphs. An example of a simple graph and a weighted graph is given in figure 3.1 (a) and (b) respectively.

We will now proceed to describe various representation methods for graphs. These representations are exact in the sense that the graph can always be totally recovered, there is no loss of information.

A *matrix representation* of the graph is a $|\mathcal{V}|$ by $|\mathcal{V}|$ matrix $\mathbf{X}$, such that an element X_{ij} of this matrix represents some property of the pair of vertices i and j. Diagonal elements X_{ii} encode information about vertex i only. Since the order of vertices in the graph does not matter, if we permute the indices associated with the graph vertices then the graph remains the same. If $\mathbf{P}$ is the permutation matrix which re-orders the vertices, then

$$\mathbf{X}' = \mathbf{P}\mathbf{X}\mathbf{P}^T \tag{3.1}$$

represents the same graph as $\mathbf{X}$. As a result, there are many matrix representations of the same graph.

3.2.1 Adjacency Matrix

The most basic matrix representation is the *adjacency matrix*. The diagonal holds the weight of the vertices and the off-diagonal's hold the weights of the edges. The formal definition of the adjacency matrix $\mathbf{A}$ is:

$$A(u,v) = \begin{cases} w_{\mathcal{V}}(u) & \text{if } u = v \\ w_{\mathcal{E}}(u,v) & \text{if } u \neq v \text{ and } (u,v) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Clearly if the graph is undirected, the matrix $\mathbf{A}$ is symmetric. As a consequence, the eigenvalues of $\mathbf{A}$ are real. These eigenvalues may be positive, negative or zero and the sum of the eigenvalues is zero.

3.2.2 Laplacian Matrix

In some applications, it is useful to have a positive semidefinite matrix representation of the graph. This may be achieved by using the *Laplacian matrix*. We first construct the diagonal degree matrix $\mathbf{D}$, whose diagonal elements are given by the vertex degrees d_v. The vertex degree is the sum of the edges incident to that vertex, $d_v = \Sigma_{i=1}^{|\mathcal{V}|} A_{vi}$.

$$D(i,j) = \begin{cases} d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

The Laplacian matrix $\mathbf{L}$ is computed by subtracting the adjacency matrix from the degree matrix:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{3.4}$$

There are two more types of Laplacian matrices each with slightly different properties: the *signless Laplacian* and the *normalized Laplacian*.

In the signless Laplacian $|\mathbf{L}|$ the degree and adjacency matrices are simply added instead of subtracted.

$$|\mathbf{L}| = \mathbf{D} + \mathbf{A} \tag{3.5}$$

The normalized Laplacian $\mathbf{N}$ has its diagonals and off diagonals scaled. Briefly, the result of this is that the eigenvalues are now bounded in the interval $[0, 2]$, which can be useful for some algorithms and theoretical results (Chapter 1, p12 [27]).

$$\mathbf{N} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{3.6}$$

3.2.3 Spectral Matrices

The eigendecomposition of a matrix can be applied to one of the matrix representations listed above to produce a *spectral representation*. This has produced the field of spectral graph theory[27] which has lead to many different applications and approaches. If $\mathbf{X}$ is a matrix representation of a graph then the *spectral decomposition* $\mathbf{X} = \mathbf{V}\mathbf{E}\mathbf{V}^T$ results in a set of eigenvalues $\mathbf{E}$ and eigenvectors $\mathbf{V}$.

There are advantages to using a spectral representation of a graph. The first is that it provides a degree of protection against the arbitrary vertex order that a graph may display. By ordering the eigenvalues by magnitude they are invariant to vertex order. However, only the columns of the eigenvector matrix may be made invariant to vertex order by arranging them according to eigenvalue magnitude. To put this formally, from a matrix representation $\mathbf{X}$ we have a matrix of eigenvalues $\mathbf{E} = \text{diag}(e_1, e_2, ..., e_n)$. Associated with an eigenvalue e_i we have an eigenvector $\mathbf{v}_i$ and the matrix of eigenvectors is formed as $\mathbf{V} = (\mathbf{v}_1|\mathbf{v}_2|...|\mathbf{v}_n)$. If we ensure that the eigenvalue magnitudes are ordered so that $e_1 \leq e_2 \leq ... \leq e_n$ then the spectral decomposition of any permutation of the matrix $\mathbf{X}$, $\mathbf{P}\mathbf{X}\mathbf{P}^T$, will result in the same set of ordered eigenvalues and ordered columns of eigenvectors. The individual components of the eigenvector columns will however vary due to the permutation.

Another advantage of using a spectral decomposition to represent a graph is that the structural properties of the graph are more apparent than in the original matrix representation. For example, as we described in section 2.1.4 on graph segmentation, the second eigenvector of the Laplacian matrix can be used to logically segment a graph. Furthermore, if the original representation is a Laplacian matrix then the number of eigenvalues with zero magnitude relate to the number of disconnected elements present in a graph.

Sometimes it is preferable to use a single matrix to describe a graph rather than a separate one for the eigenvalues and eigenvectors. This can be accomplished by combining the eigenvalues and eigenvectors in the following way:

$$\mathbf{K} = \mathbf{V}\sqrt{\mathbf{E}} = (\sqrt{e_1}\mathbf{v}_1|\sqrt{e_2}\mathbf{v}_2|...|\sqrt{e_n}\mathbf{v}_n) \tag{3.7}$$

the original matrix representation can be recovered by:

$$\mathbf{X} = \mathbf{K}\mathbf{K}^T \tag{3.8}$$

The eigendecomposition can be performed by the QR algorithm [49] in $O(n^3)$ time.

3.2.4 Heat Kernel

The *heat kernel*, describes a graph by the way heat would spread through it, similar to the way heat would spread through a real world structure. The heat kernel is described in terms of the spectral decomposition of the Laplacian:

$$\mathbf{H} = \mathbf{V} \exp\left[-t\mathbf{E}\right]\mathbf{V}^T \tag{3.9}$$

The time parameter t can be used to obtain a representation that either favours local connectivity or global connectivity in the graph. If t is small the local connectivity of the graph is represented in the heat kernel and if t is large then the global connectivity of the graph is represented instead.

3.3 Method

The spectral representation is an interesting one in terms of mixing graphs for a number of reasons. As mentioned earlier, part of the correspondence problem is solved in the spectral representation; the columns of $\mathbf{V}$ are ordered by the associated eigenvalue magnitudes which are not affected by the vertex labeling. Secondly, in the Laplacian and related matrices, structures of different scales in the graph are associated with different eigenmodes. As a result, it is possible to mix different scales separately using the spectral representation.

3.3.1 Spectral Alignment and Mixing

We begin from a set of graphs $\mathcal{S}$ and compute a matrix representation $\mathbf{X}_k$ for each graph $S_k \in \mathcal{S}$. We take the eigendecomposition of each matrix representation $\mathbf{X}_k = \mathbf{E}_k\mathbf{V}_k\mathbf{E}_k^T$ to produce a set of eigenvalues $\mathbf{E}_k$ and eigenvectors $\mathbf{V}_k$ describing

the graph. The eigenvalues and associated eigenvectors are ordered by magnitude. If the graphs in the set contain different numbers of vertices then we pad the representations to accommodate the largest graph.

Before mixing the spectral representations of two or more graphs, we must first align the rows of each $\mathbf{V}_k$ so that they are in the same order. This may be done using a spectral graph matching algorithm such as Umeyama's method [106] or a variant such as that of Ferrer et al [37]. In order for the process of mixing spectral modes to be effective, the eigensystems of the graphs must be relatively similar. These spectral methods of alignment should be effective on such graphs. In cases where we are testing the limits of the approach, i.e. when the graph spectrums differ significantly within the set, we use the graph matching algorithm of Gold and Rangarajan [48] to overcome this limitation.

It is well known that the eigenvectors of the decomposition of a matrix are sign-ambiguous. In other words, the eigenvectors are recovered up to a sign factor of ± 1. It is necessary to determine these factors if we are to correctly mix the corresponding eigenmodes. Our method is based on identifying the largest component of an eigenvector and correcting the sign based on that coordinate. Given a set of spectral matrices $\{\mathbf{V}_1, \mathbf{V}_2, ..., \mathbf{V}_{|\mathcal{S}|}\}$, let $\mathbf{v}_{ij}$ be the jth eigenvector (mode) from $\mathbf{V}_i$. The kth component of this eigenvector can then be denoted v_{ijk}. We find the largest magnitude component for mode j from

$$l_j = \arg\max_k \sum_i |v_{ijk}| \tag{3.10}$$

We then correct the sign of the eigenvectors by ensuring that component l_j is positive for mode j in all the spectral matrices. In other words, we correct the eigenvectors based on the component that will cause the largest error if left unaligned. We denote the aligned sign-corrected eigenvectors of a graph S_k as $\mathring{\mathbf{V}}_k$.

Once aligned, the spectral matrices of two graphs S_i and S_j may be merged by simply taking the average of the matrices, i.e.

$$\mathbf{V}_m = \frac{1}{2}(\mathring{\mathbf{V}}_i + \mathring{\mathbf{V}}_j) \tag{3.11}$$

$$\mathbf{E}_m = \frac{1}{2}(\mathbf{E}_i + \mathbf{E}_j) \tag{3.12}$$

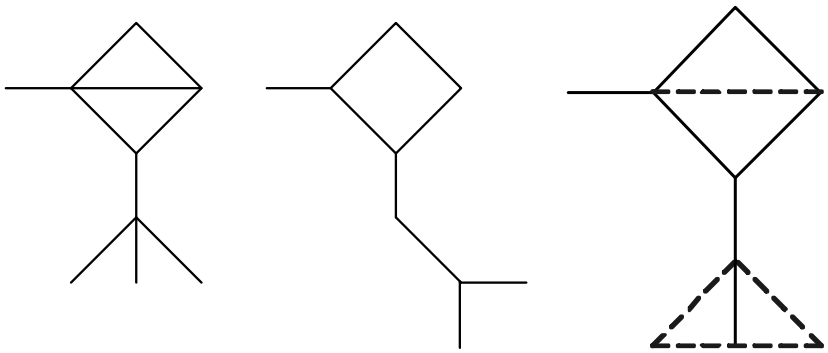


Figure 3.2: Sample graphs and the spectral average graph. Dotted edges have weights of close to $\frac{1}{2}$.

which will give a combination of two graphs. Figure 3.2 demonstrates an example of this mixing process. Figure 3.2 shows the original graphs and the result of reconstruction from the mixed spectral representation using the method detailed in the next section.

The spectral representation is a particularly flexible one for mixing graphs since there is a separation of different scales of the graph in the ordering of the eigenvalues. It is therefore possible to mix graphs by selecting different parts of the spectrum from each of the graphs being mixed. This enables the selection of different parts of the structure from the different graphs. In order to achieve this, we define a mixing matrix thus:

$$\mathbf{M}_k = \begin{pmatrix} f_{k,1} & 0 & \ldots & 0 \\ 0 & f_{k,2} & \ldots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \ldots & f_{k,n} \end{pmatrix}, \sum_{k=1}^{|\mathcal{S}|} \mathbf{M}_k = \mathbf{I} \tag{3.13}$$

The diagonal elements f_{kj} define the fraction of mode j which is selected from the spectrum of graph k. The mixed spectral matrices are then defined as:

$$\mathbf{V}_m = \sum_{k=1}^{|\mathcal{S}|} \mathring{\mathbf{V}}_k \mathbf{M}_k \tag{3.14}$$

$$\mathbf{E}_m = \sum_{k=1}^{|\mathcal{S}|} \mathbf{E}_k \mathbf{M}_k \tag{3.15}$$

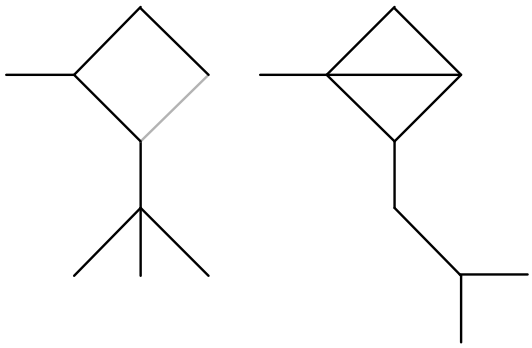


Figure 3.3: Mixing spectral modes in different proportions from two graphs

Figure 3.3 shows the results of mixing the two example graphs from Figure 3.2 in different proportions. On the left, the modal proportions are $[1, 0.5, 0, 0, 0, 0, 0, 0]$ and $[0, 0.5, 1, 1, 1, 1, 1, 1]$ from the first and second graphs respectively. On the right, the proportions are reversed, i.e. $[0, 0.5, 1, 1, 1, 1, 1, 1]$ and $[1, 0.5, 0, 0, 0, 0, 0, 0]$. By selecting the proportions we are able to mix key structures from each of the two graphs.

3.3.2 Reconstruction

With the merged spectral representations to hand, we can reconstruct a graph using the reverse of the eigendecomposition:

$$\mathbf{X}_m = \mathbf{V}_m \mathbf{E}_m \mathbf{V}_m^T \tag{3.16}$$

In general, the averaging process will lead to a matrix $\mathbf{X}_m$ which does not have the required properties. Firstly, the mixed spectral matrix $\mathbf{V}_m$ will not be orthonormal. In order to correct this we apply an orthogonalization procedure and a normalizing step to $\mathbf{V}_m$ to obtain an orthonormal spectral matrix $\mathbf{V}'_m$:

$$\mathbf{V}'_m = (\mathbf{V}_m \mathbf{V}_m^T)^{-\frac{1}{2}} \mathbf{V}_m \tag{3.17}$$

Secondly, the reconstructed matrix representation may not be consistent with the chosen representation. For example, if we are operating with Laplacians, the diagonal elements will not be the vertex degrees. In addition, the edges will be weighted and unless we are using weighted graphs this must be corrected. There is therefore a need to project $\mathbf{X}_m$ onto the nearest graph.

It is tempting to interpret the weights as edge probabilities. In fact we could select edges in the final conditioned Laplacian with a probability equal to their weights in $\mathbf{X}_m$. This approach would generate sets of random graphs which are close to the original graph. On the other hand, this ignores the co-occurrence of edges in the original graphs. A simple and satisfactory solution is to use a threshold for edges and non-edges. We define a function θ that operates on a matrix and thresholds the elements based on their magnitude:

$$\theta : A'_{ij} \rightarrow \begin{cases} 0 & |A_{ij}| < \beta \\ 1 & |A_{ij}| \geq \beta \end{cases} \tag{3.18}$$

If an element is greater than an adjustable parameter β then we record an edge, otherwise no edge is recorded. Although this function is only applicable to adjacency matrices, it can be extended to recover different matrix representations.

The final mixed matrix representation $\mathbf{X}'_m$ can be recovered as follows, where the correct thresholding function is selected for the matrix representation used.

$$\mathbf{X}'_m = \theta(\mathbf{X}_m) \tag{3.19}$$

3.3.3 Spectral Median Graph

We now have all the ingredients we require to give our formal definition of the spectral median graph G_S. Note that this differs from the definition given by Ferrer et al [37] (see section 2.1.5). Using the aligned sign-corrected spectral decomposition of each graph in the sample set we compute the following matrices:

$$\mathbf{V}_S = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \mathring{\mathbf{V}}_k \tag{3.20}$$

$$\mathbf{E}_S = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \mathring{\mathbf{E}}_k \tag{3.21}$$

We then apply the reconstruction procedure described above, that is, a) orthogonalize the matrix $\mathbf{V}_S$, b) use the reverse of the spectral decomposition to find the graph matrix representation and c) apply the threshold function to the recovered matrix. This yields the spectral median graph G_S.

3.4 Experimental Results

In this section we work with the normalized Laplacian matrix representation of a graph. In the first set of experiments, we take a graph set and determine the spectral median graph G_S by the method described above. In order to visualize the results, we produce MDS plots based on the edit distances between the graphs. For comparison purposes, we have found the generalized median graph as given by Jiang et al[58], which is defined by:

$$G_M = \underset{S \in \mathcal{U}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{S}|} e(S, S_i) \tag{3.22}$$

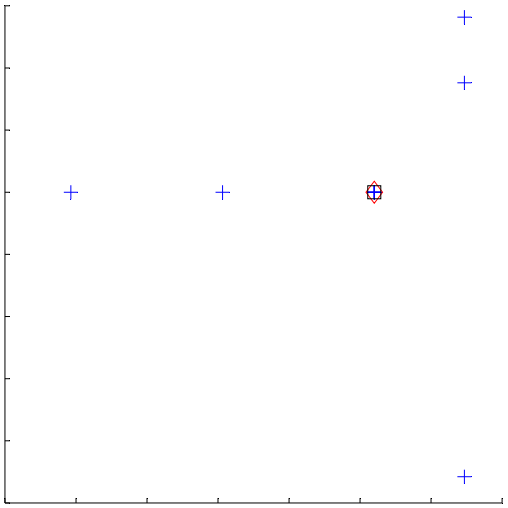
where $e(.,.)$ is the edit distance and $\mathcal{U}$ is the set of all graphs. We can also use the sum of distances to all graphs $S \in \mathcal{S}$ in the set as a measure of the quality of a median graph. It is defined as:

$$SOD(G, \mathcal{S}) = \sum_{k=1}^{|\mathcal{S}|} e(G, S_k) \tag{3.23}$$

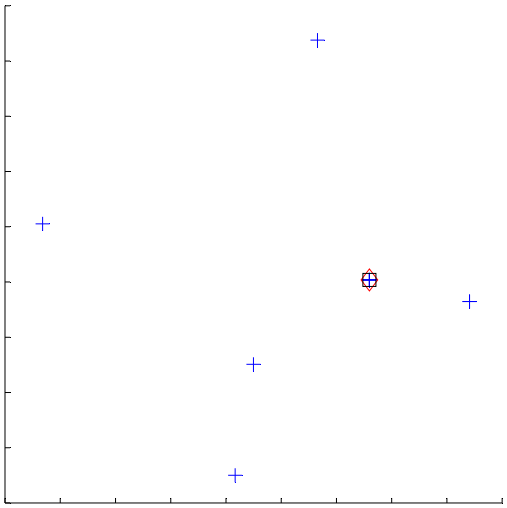
As discussed in section 2.1.5 finding the generalized median graph is a computationally complex problem. Therefore, we limit our data sets to graphs with 7 vertices in cases where we must calculate the generalized median graph.

In Figure 3.4 we give results for a set of 10 graphs each with 6 vertices. The 10 graphs are generated by computing a seed graph and then perturbing the edges with a specific probability. This ranges from 5% of edges being perturbed in the first plot to 30% in the fourth plot. For each plot the following information is given: a) the sum of distances between the graphs in $\mathcal{S}$ and the generalized median G_M: $SOD(G_M, \mathcal{S})$, b) the sum of distances between the graphs in $\mathcal{S}$ and the spectral median G_S: $SOD(G_S, \mathcal{S})$ and c) the edit distance between the generalized median and the spectral median: $e(G_M, G_S)$.

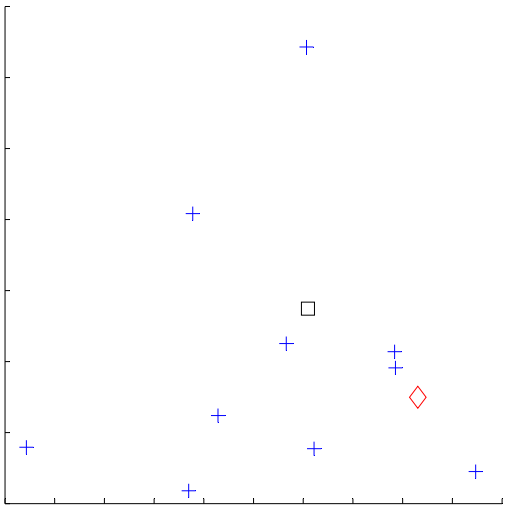
When only 5% or 10% of the edges are perturbed the spectral median graph we calculate is equal to the generalized median. However, as the noise level increases (20% and 30% of edges perturbed) the distance between G_M and G_S increases. In all the plots the generalized median (black square marker) is relatively near the center. On the other hand, the spectral median (red diamond marker) at high noise levels is usually found nearer the edge. As the noise level increases, the spectral rep-



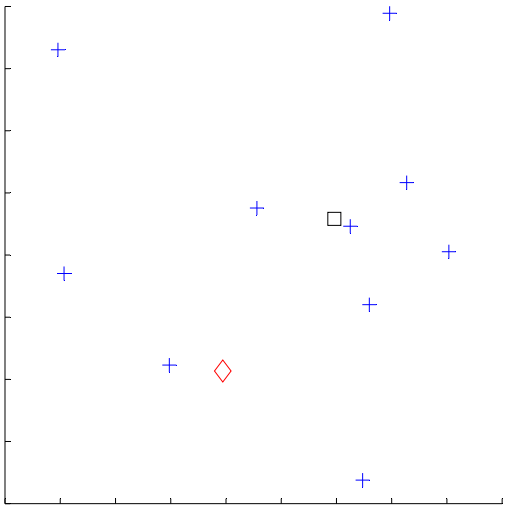
Edges perturbed:	5%
$SOD(G_M, \mathcal{S})$:	6
$SOD(G_S, \mathcal{S})$:	6
$e(G_M, G_S)$:	0



Edges perturbed:	10%
$SOD(G_M, \mathcal{S})$:	10
$SOD(G_S, \mathcal{S})$:	10
$e(G_M, G_S)$:	0



Edges perturbed:	20%
$SOD(G_M, \mathcal{S})$:	32
$SOD(G_S, \mathcal{S})$:	42
$e(G_M, G_S)$:	2



Edges perturbed:	30%
$SOD(G_M, \mathcal{S})$:	39
$SOD(G_S, \mathcal{S})$:	55
$e(G_M, G_S)$:	5

Figure 3.4: Spectral median graph results plotted using MDS. The set $\mathcal{S}$ consists of 10 graphs each with 6 vertices. Markers are as follows: blue plus signs - graphs from the set $\mathcal{S}$, red diamond - spectral median graph G_S and black square - generalized median graph G_M.

resentation of each sample graph describes increasingly different structural patterns. This results in errors in our approach of calculating the spectral median through a direct average.

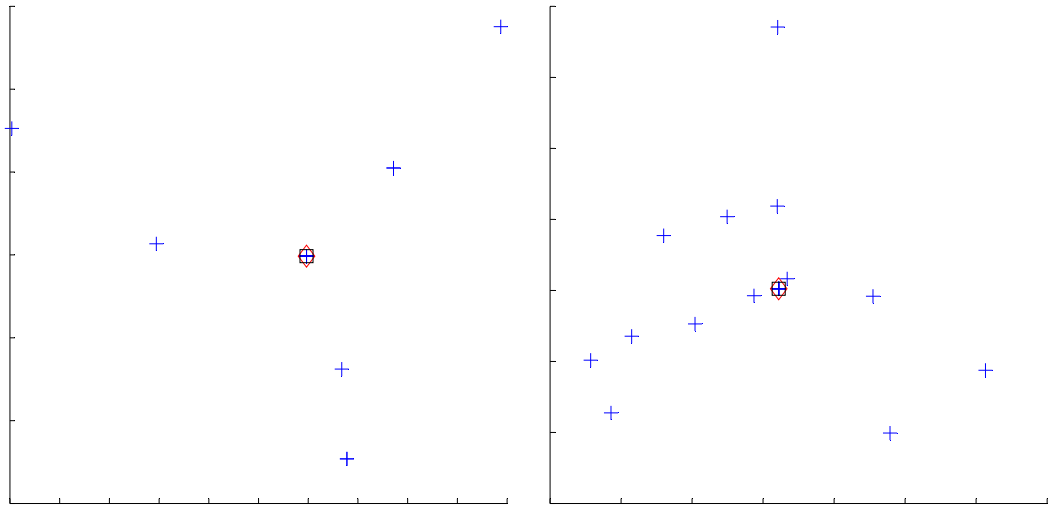
We perform the same experiment with a set of 20 graphs and the results of this are given in Figure 3.5. The noticeable aspect of this is that the increased number of graphs used to compute the averages do not seem to have a negative effect on the quality of the spectral median. Although it is important to remember we are working with small graphs here due to computational constraints.

In Figures 3.6 ($|\mathcal{S}| = 10$) and 3.7 ($|\mathcal{S}| = 20$) we describe the results when the graphs in the set are composed of 7 vertices. Again, we see broadly the same results as those described above across the different levels of corruption.

Finally, in Figure 3.8 we describe results for a set of 20 graphs where half of the set is produced from one seed graph and the other half is produced from a different seed graph. This experiment is designed to test the limits of the approach, i.e. when the graphs in the set do not represent similar strictures. We do not expect the approach to be effective here since the spectral representation of each graph in the sample set will no longer describe similar structures. This has been observed in other spectral approaches. Umeyama[106] commented that the effectiveness of his approach is inherently limited by the structural similarity of the two graphs that are being matched.

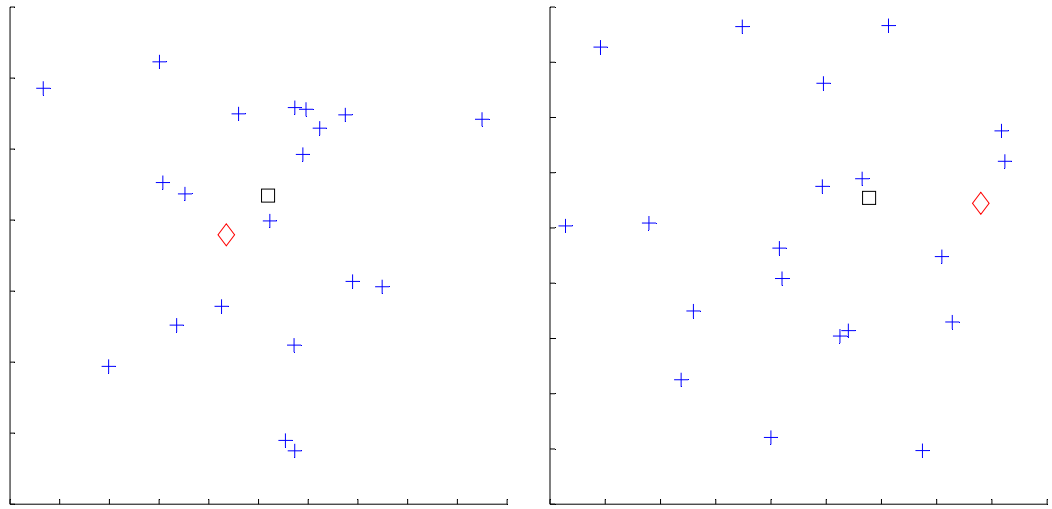
We see the expected results here. In all the previous tests the spectral graph was equal to the generalized median graph at noise levels of 5% and 10%. However, here there is a difference between the structure of the spectral median and the generalized median at a noise level of 10%. Furthermore, the edit distance $e(G_M, G_S)$ is quite large considering that the graphs only have 6 vertices.

One of the key benefits of mixing graphs in spectral representations is the ability of the representation to separate different structural parts of the graph. In order to examine the effectiveness of this approach, we have constructed the following experiment. In this experiment we generate two graph classes. Both classes are generated by joining two seed structures by a common set of vertices. Let the seed graphs be C_1, C_2 and C_3. Then set one is constructed by joining C_1 and C_2, and set two by joining C_1 and C_3. The edges of the joined graphs are then perturbed to provide some structural variation within each set. We construct 10 graphs for each



Edges perturbed:	5%
$SOD(G_M, \mathcal{S})$:	12
$SOD(G_S, \mathcal{S})$:	12
$e(G_M, G_S)$:	0

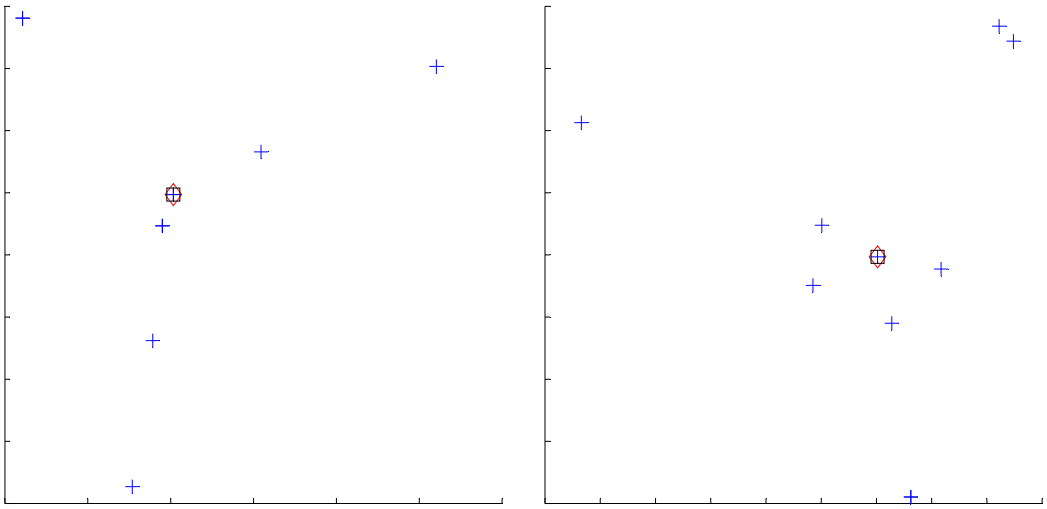
Edges perturbed:	10%
$SOD(G_M, \mathcal{S})$:	27
$SOD(G_S, \mathcal{S})$:	27
$e(G_M, G_S)$:	0



Edges perturbed:	20%
$SOD(G_M, \mathcal{S})$:	58
$SOD(G_S, \mathcal{S})$:	68
$e(G_M, G_S)$:	1

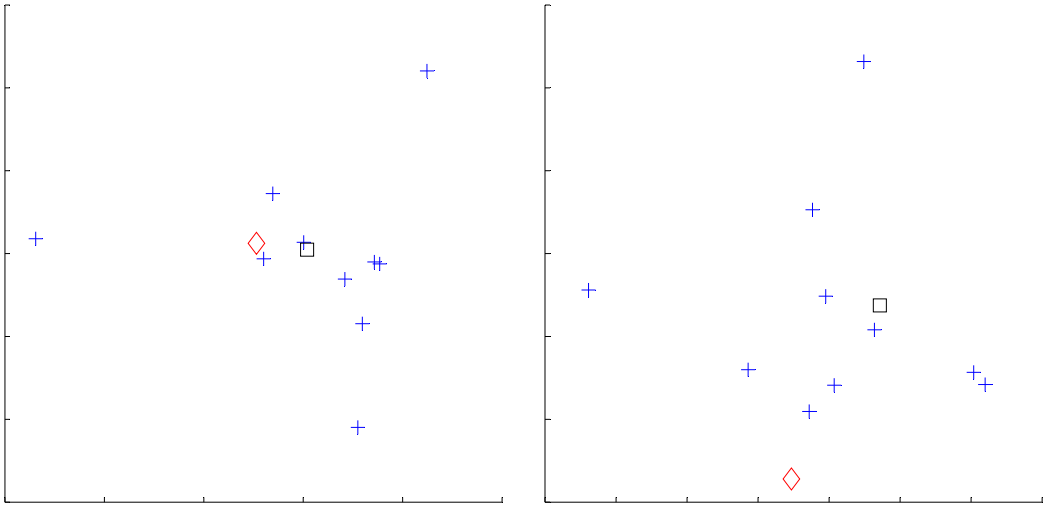
Edges perturbed:	30%
$SOD(G_M, \mathcal{S})$:	82
$SOD(G_S, \mathcal{S})$:	96
$e(G_M, G_S)$:	2

Figure 3.5: Spectral median graph results plotted using MDS. The set $\mathcal{S}$ consists of 20 graphs each with 6 vertices. Markers are the same as the previous plot.



Edges perturbed:	5%
$SOD(G_M, \mathcal{S})$:	13
$SOD(G_S, \mathcal{S})$:	13
$e(G_M, G_S)$:	0

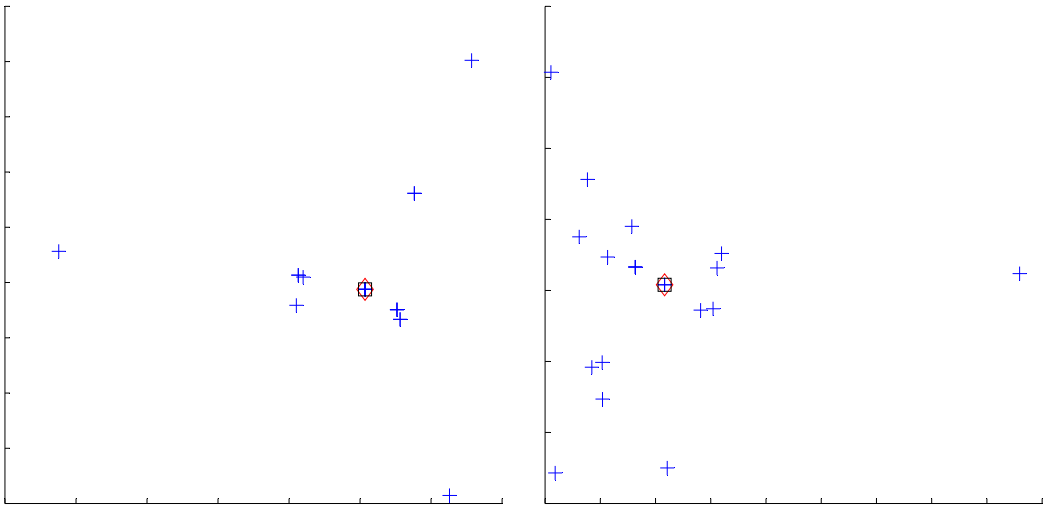
Edges perturbed:	10%
$SOD(G_M, \mathcal{S})$:	21
$SOD(G_S, \mathcal{S})$:	21
$e(G_M, G_S)$:	0



Edges perturbed:	20%
$SOD(G_M, \mathcal{S})$:	37
$SOD(G_S, \mathcal{S})$:	41
$e(G_M, G_S)$:	1

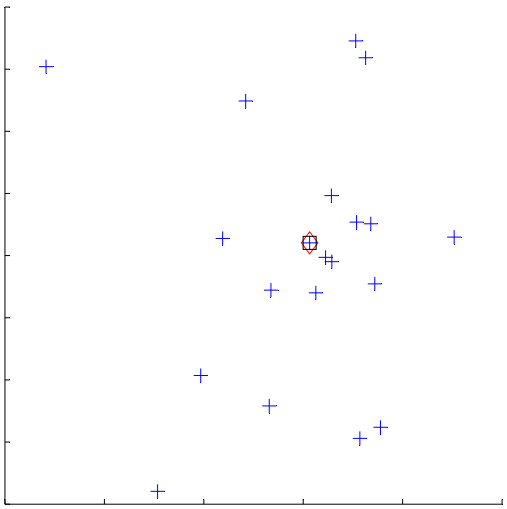
Edges perturbed:	30%
$SOD(G_M, \mathcal{S})$:	63
$SOD(G_S, \mathcal{S})$:	79
$e(G_M, G_S)$:	7

Figure 3.6: Spectral median graph results plotted using MDS. The set $\mathcal{S}$ consists of 10 graphs each with 7 vertices. Markers are the same as the previous plots.

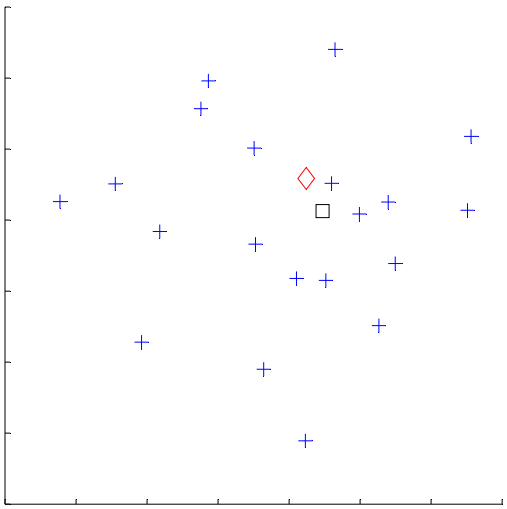


Edges perturbed:	5%
$SOD(G_M, \mathcal{S})$:	20
$SOD(G_S, \mathcal{S})$:	20
$e(G_M, G_S)$:	0

Edges perturbed:	10%
$SOD(G_M, \mathcal{S})$:	38
$SOD(G_S, \mathcal{S})$:	38
$e(G_M, G_S)$:	0



Edges perturbed:	20%
$SOD(G_M, \mathcal{S})$:	63
$SOD(G_S, \mathcal{S})$:	63
$e(G_M, G_S)$:	0



Edges perturbed:	30%
$SOD(G_M, \mathcal{S})$:	137
$SOD(G_S, \mathcal{S})$:	169
$e(G_M, G_S)$:	7

Figure 3.7: Spectral median graph results plotted using MDS. The set $\mathcal{S}$ consists of 20 graphs each with 7 vertices. Markers are the same as the previous plots.

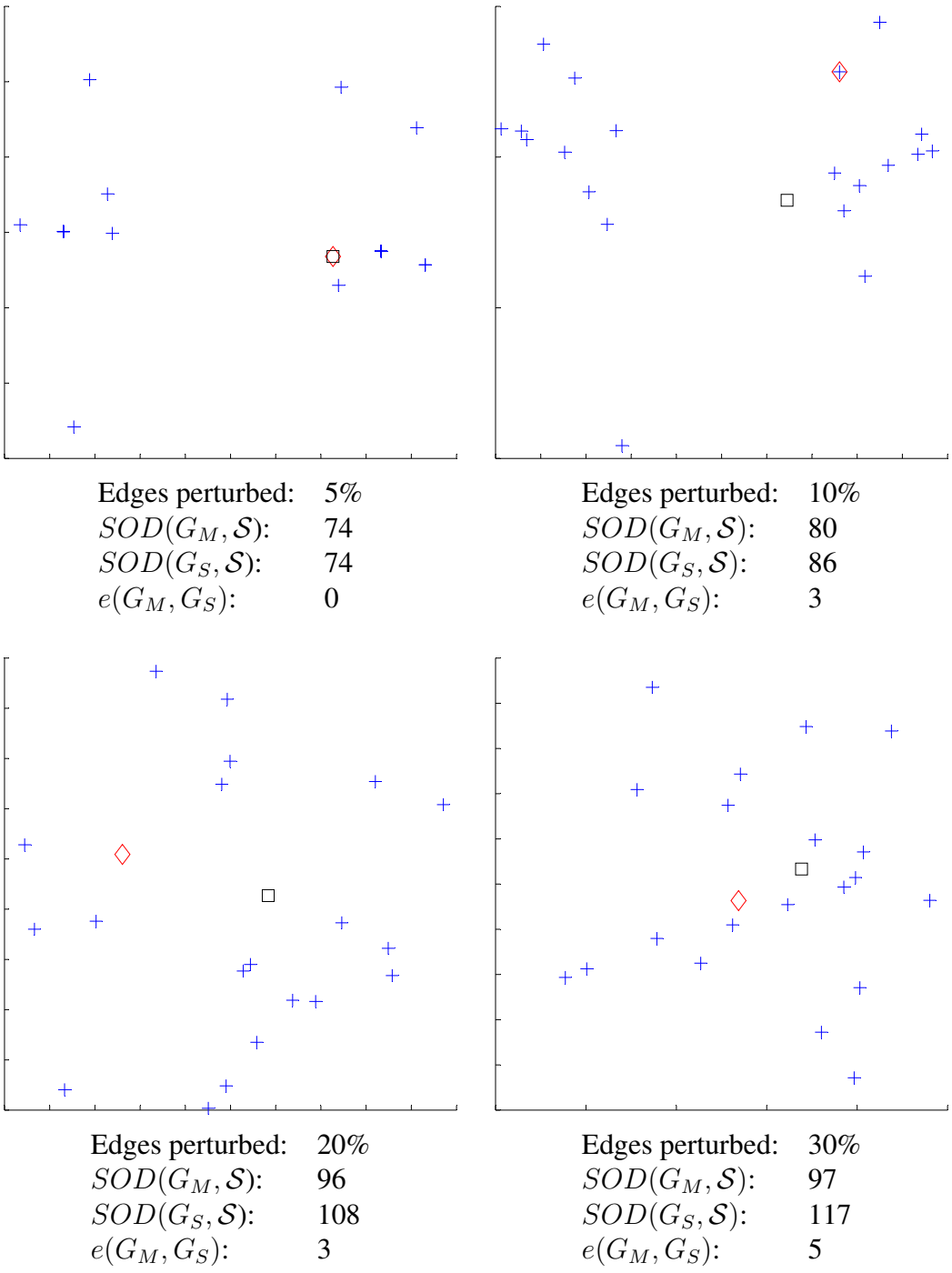


Figure 3.8: Spectral median graph results plotted using MDS. The set $\mathcal{S}$ consists of 20 graphs in total where 10 are constructed by perturbing one seed graph and the other 10 are constructed by perturbing a second seed graph. All graphs have 6 vertices.

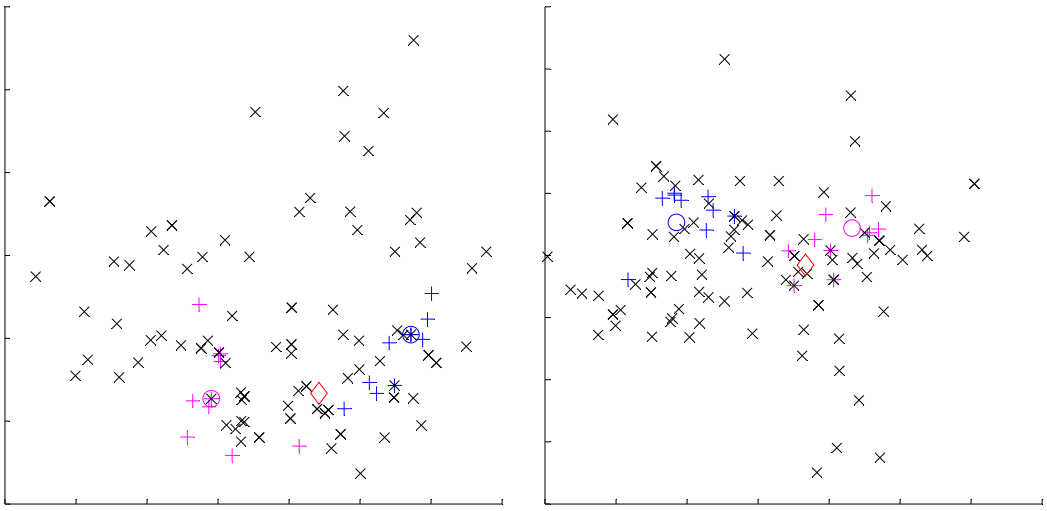


Figure 3.9: Mixing the spectral modes of two graph sets. Set one is shown using blue markers and set two is shown using magenta. Plus signs denote sample graphs (the color determines the set), a circle denotes the spectral median graph for a specific set, the red diamond indicates the spectral median graph for both sets combined and black crosses denote graphs constructed using the mixing approach. Two results are shown for the same experimental parameters: graph size is 10, edge edit rate is 10% and the size of each sample set is 10.

set which gives a total of 20 sample graphs.

We construct a mixed graph by mixing the spectral representation of one graph from each class. To do this we use two mixing matrices $\mathbf{M}_1$ and $\mathbf{M}_2$. The modal proportions chosen for the mixed graph are randomly generated with the following restrictions: 1) each weight may be drawn from $\{0, 1\}$ and 2) mixing weights must always decrease or increase along the main diagonal. Combined with the requirement that the mixing matrices must sum to $\mathbf{I}$, $\mathbf{M}_1 = \text{diag}([1, 1, 0, 0])$ and $\mathbf{M}_2 = \text{diag}([0, 0, 1, 1])$ are an acceptable pair but $\mathbf{M}_1 = \text{diag}([1, 0, 1, 0])$ and $\mathbf{M}_2 = \text{diag}([0, 1, 0, 1])$ are not. We impose this restriction since, in most cases, it does not make sense to interleave different levels of structure.

We visualize the results of these experiments by using MDS to embed the sample and generated graphs. We also compute the spectral median of each set separately and the spectral median of both sets combined. In Figure 3.9 we show two results using graphs of size 10 (the size of each seed graph is 5). It is clear that the mixed graphs span the space well both around and in-between the two sample sets. Furthermore, the three computed spectral medians all lie in the expected positions.

In Figure 3.10 we show results for a sample set containing graphs of 16 vertices

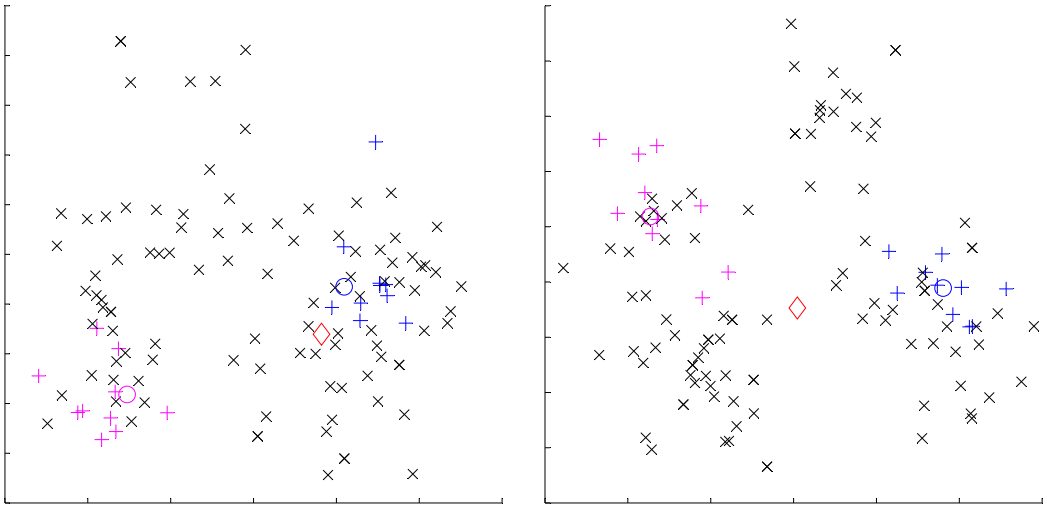


Figure 3.10: Mixing the spectral modes of two graph sets. Markers are as on the previous plot. Two results are shown for the same experimental parameters: graph size is 16, edge edit rate is 10% and the size of each sample set is 10.

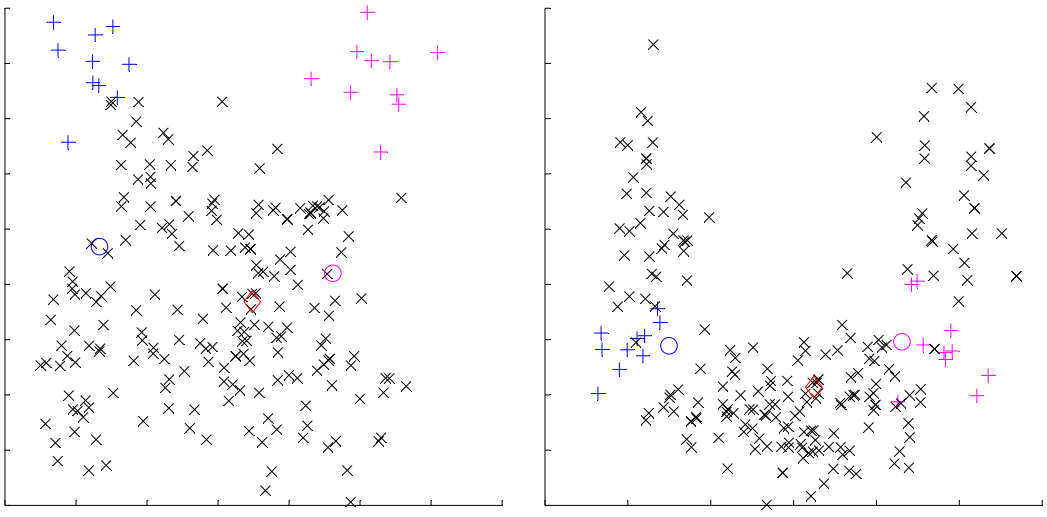


Figure 3.11: Mixing the spectral modes of two graph sets. Markers are as on the previous plot. Two results are shown for the same experimental parameters: graph size is 24, edge edit rate is 10% and the size of each sample set is 10.

(each seed graph is now of size 8). Again we construct mixed graphs that span the space well, but not as effectively as before indicating that the difference in structure present in the two sample sets is beginning to cause problems. This result is reinforced when we examine the results for graphs of size 24 (Figure 3.11). In these plots it is clear that we are no longer successfully generating structures that span the space or lie around the sample sets. Clearly the structural variations between the two sets are too large for our method to be effective. Note however that the spectral medians all still seems to appear in the appropriate positions.

Given the results of this last experiment we evaluated how effective the approach was as the graphs become significantly larger. We draw examples from a single seed graph here to reduce the structural variations present. The results of this experiment are shown in Figure 3.12. For graph sizes of 20 and 30 vertices ((a) and (b) in Figure 3.12) the mixing approach still seems successful. However, when the graph size is increased to 40 (Figure 3.12(c)) it is clear that mixed graph are no longer being generated that are near to those of the sample set.

This does not necessarily mean that the approach cannot handle graphs this large, instead it suggests that there is a vast number of different structural patterns that may be generated using this data set. To ensure the method only generates structures that are similar to those of the input we must turn to the generative models described later in this thesis. These approaches actually go a step further from generating just similar structures, they produce structures that are drawn from the distribution of structural variation described by the sample set.

We make use of one more experiment to assess the performance of our approach. In this case we are interested in the ability to mix structure and construct useful spectral medians when the sample set contains graphs of greatly differing sizes. To investigate this we use the approach of joining core graphs from the earlier experiment. In this case, one set is produced from core graph C_1 and the second set is produced from C_1 joined with C_2. This ensures that we are not trying to mix graphs describing totally different structure.

In Figure 3.13(a) we give results for $|C_1| = 10$ and $|C_1| + |C_2| = 20$. When the graphs differ considerably in size it seems easier to construct mixed graphs near each sample set as opposed to in-between the two sets. It is also interesting to note that the combined spectral median always lies closer to the set consisting of smaller

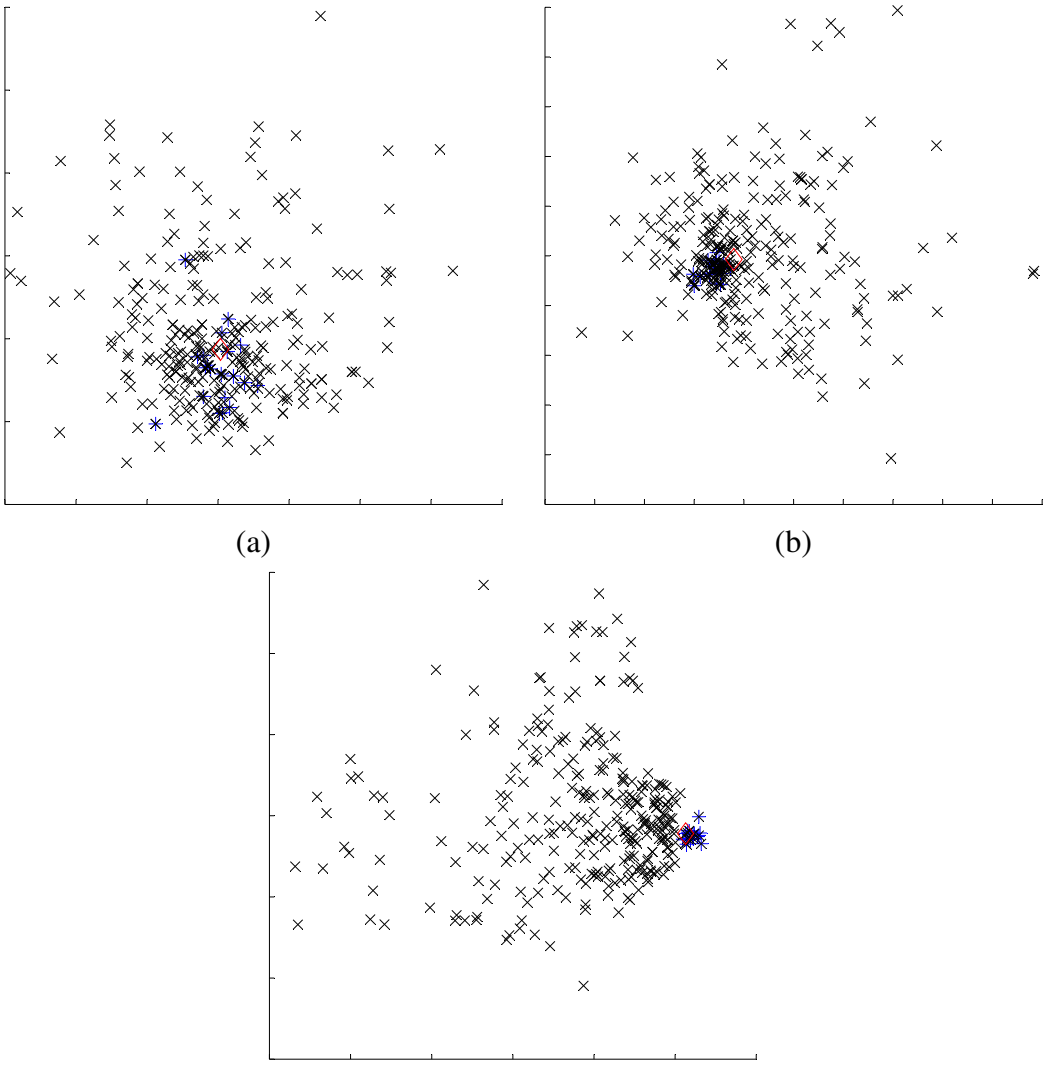


Figure 3.12: Mixing the spectral modes of one graph set. Markers are as on the previous plot (although only one set is used here). The graph sizes vary in each plot as follows: (a) 20, (b) 30, (c) 40. The edge edit rate is 5% and the size of each sample set is 20.

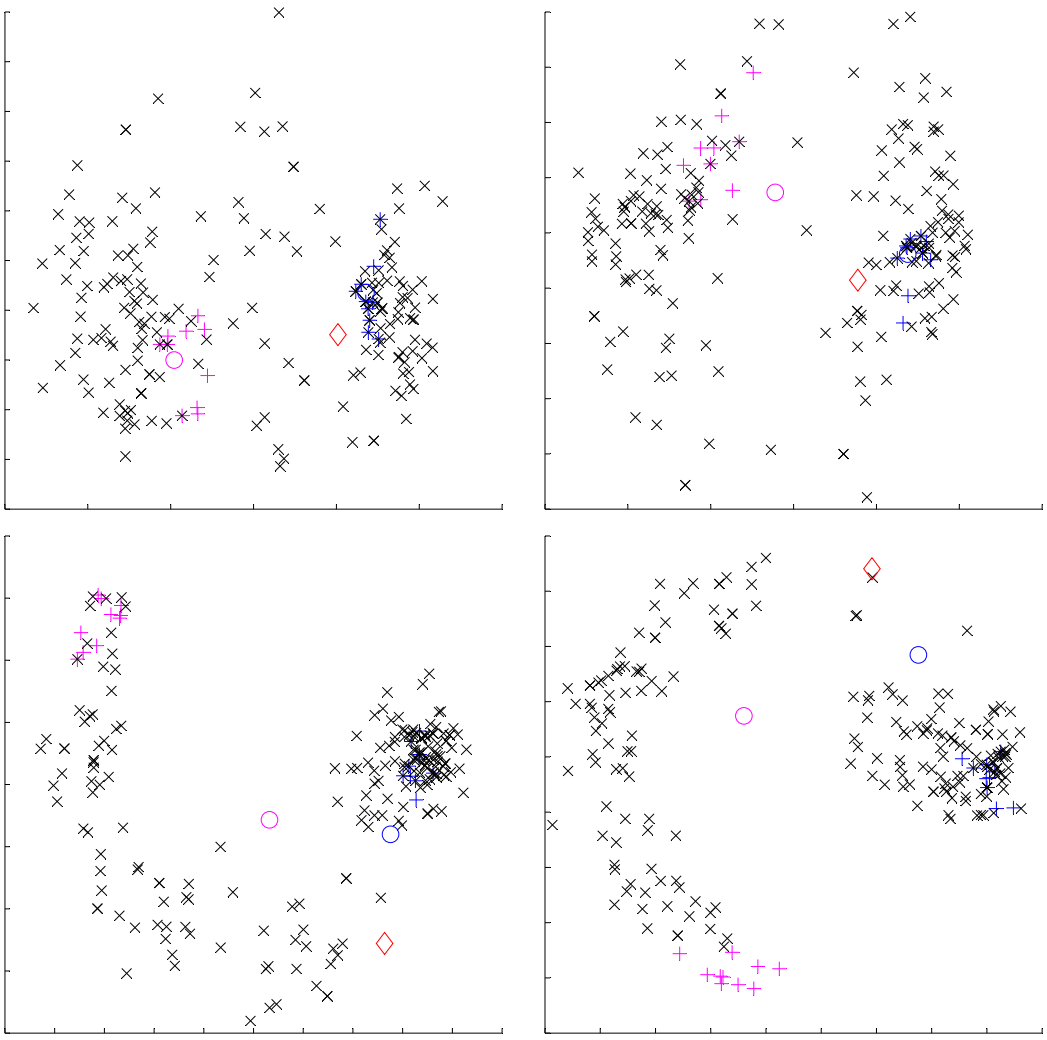


Figure 3.13: Mixing the spectral modes of two graph sets containing graphs of different sizes. On the top row, set one has 10 vertices (blue markers) and set two has 20 vertices (magenta markers). On the bottom row, set one has 15 vertices (blue markers) and set two has 30 vertices (magenta markers). The edge edit rate is 10% and the size of each sample set is 10.

graphs (blue markers indicate the smaller set). The same is true for the combined spectral median in Figure 3.13(b). Here $|C_1| = 15$ and $|C_1 + C_2| = 30$. However, when there is a larger difference in graph size, the mixed graphs are only successful when the modal proportions favour the smaller graph.

3.5 Conclusion

In this chapter we have described a method of mixing graph structure using the spectral representation. This method can be used to compute the median of a set containing two or more graphs. Furthermore, by employing the spectral representation we gain the ability to mix different levels of structure, by combining, for example, the global structure of one graph with the local structure of another. Of course, since our approach relies on the spectral representation of each graph in the set being similar, this method is only appropriate on graphs displaying similar structure.

For our approach to work we must also solve the correspondence problem between a set of graphs. The spectral decomposition gives us an ordering over the eigenvectors but not over the components of each eigenvector. We must make use of a graph alignment algorithm to complete the alignment. Furthermore, since the eigenvectors are only computed up to a sign factor ± 1, we must also perform a sign-correction step. To do this we align each eigenmode by correcting the component that would cause the largest error if left uncorrected.

With the aligned spectral representations to hand, we can either proceed to construct a median graph by averaging the spectral representations or to construct mixed graphs by combining eigenmodes of sample graphs in different proportions. In both cases, the reconstruction of a mixed graph from the computed spectral representation is not a simple step. We must first form a set of orthogonal eigenvectors by applying an orthogonalization procedure. To recover a correct matrix representation of our mixed graph we also apply a thresholding step.

To evaluate the quality of the spectral median graph we used two different approaches. In the first we compared our spectral median graph to the generalized median. In these experiments we observed that, at low noise levels, the spectral median equals the generalized median. However, due to the restrictive computational

complexity of finding the generalized median graph, these experiments were only performed on small graphs. To assess the performance when working with larger graphs we examined MDS embeddings of the sample set and spectral median. This allowed us to visualize the relative distances between the spectral median and the other graphs in the set.

To evaluate our second contribution, constructing new graphs by mixing spectral representations, we performed experiments that involved mixing structures from two graphs. Again these new structures, as well as the sample graphs, were visualized using MDS. To provide challenging data for our algorithm we used data sets that contained a) similar structures, b) different structures of the same size and c) structures of different sizes.

Ferrer et al [38] has performed a comparison between the approach detailed in this chapter and their method of constructing the spectral median graph through incremental updates [37]. The test domain is that of graphical symbol recognition. They report that both methods perform similarly except under high levels of noise in which their method outperforms the one described in this chapter.

3.5.1 Future Work

It would be interesting to extend our experiments on median graph computation to larger sets consisting of larger graphs. With the new faster algorithms available for exact generalized median computation [41, 42] we would be able to compare the spectral median graph that our approach produces with the exact one produced through search based approaches. This would give a much better indication of the applicability of this approach on more realistic data.

3.6 Symbols

$\mathcal{S}$	A set of graphs.
S_k	A graph drawn from the set $\mathcal{S}$.
G_M	The generalized median graph of a set $\mathcal{S}$.
G_S	The spectral median graph of a set $\mathcal{S}$.
G	A graph with vertex set $\mathcal{V}$, an edge set $\mathcal{E}$, a weight function on the vertices $w_{\mathcal{V}}$ and a weight function on the edges $w_{\mathcal{E}}$.
$\mathbf{X}$	A matrix representation of a graph.
$\mathbf{P}$	A permutation matrix.
$\mathbf{A}$	An adjacency matrix representation of a graph.
$\mathbf{L}$	A Laplacian matrix representation of a graph.
$\|\mathbf{L}\|$	A signless Laplacian matrix representation of a graph.
$\mathbf{N}$	A normalized Laplacian matrix representation of a graph.
$\mathbf{E}, \mathbf{V}$	The set of eigenvectors $\mathbf{E}$ and eigenvalues $\mathbf{V}$ resulting from the eigen-decomposition of a matrix representation of a graph. Individual eigenvalues and associated eigenvectors are denoted e_i and $\mathbf{v}_i$ respectively.
$\mathring{\mathbf{V}}$	An aligned sign-corrected set of eigenvectors.
l_j	The component of eigenmode j that has greatest magnitude when summed over a set of eigenvectors from that mode.
$\mathbf{K}$	A single matrix spectral representation of a graph, $\mathbf{K} = \mathbf{V}\sqrt{\mathbf{E}}$.
$\mathbf{H}$	The heat kernel matrix representation of a graph.
$\mathbf{E}_m, \mathbf{V}_m$	A set of mixed eigenvalues and eigenvectors.
$\mathbf{E}_S, \mathbf{V}_S$	The eigenvalues and (non-orthogonalized) eigenvectors of the spectral median graph.
$\mathbf{V}'_m$	A set of mixed orthonormal eigenvectors.
$\mathbf{M}_k$	The spectral mixing matrix for sample graph S_k.
θ	A thresholding function for a matrix representation. A parameter β determines the threshold level.
$\mathbf{X}'_m$	A thresholded matrix representation of a mixed graph.
$e(.,.)$	Graph edit distance.

Chapter 4

Vectorial Generative Models for Graphs

4.1 Introduction

Generative models are well-known tools for representing patterns which reside in a vector space. Models such as the multivariate normal distribution are easily defined in a vector space from the statistics of sample patterns. However, defining a generative model for graph data is more complex, since the sample data does not originally reside in a vector space and concepts such as mean and variance are not so easily defined. This arises from the arbitrary ordering that may be placed over the vertices of a graph.

The approach we will be taking in this chapter is to carry out an alignment step on the sample graphs which brings them all into a canonical order. The graphs can then be represented by vectors in the canonical order and distributions can be defined over the vector space. We are specifically interested in the representations and the structure of the distributions. We will also insist that the model is fully generative in the sense that a graph can be reconstructed from its representation, and so new graphs can be constructed from the distributions.

We present six different models in total each differing in the way the vector space is constructed. Simple models are based directly on the different matrix representations of a graph, the adjacency matrix and the Laplacian matrix. We obtain models in the spectral domain by performing an eigendecomposition on a simple matrix representation. In one model we construct a single distribution over a combined spectral matrix and in the other we construct a separate distribution for the

eigenvalues and eigenvectors. While both these models perform adequately, the matrices recovered from generated vectors fail to satisfy all the properties that are present in the original matrix representation. Therefore, we consider vector spaces derived from manifolds that allow us to enforce certain properties of the recovered matrices.

We evaluate the various models on three different data sets. The first two involve synthetic data which consists of random graphs and graphs constructed by performing Delaunay triangulations on point sets. The third data set consists of graphs constructed from real-world images taken from the COIL database.

To construct a vector space, we need to place the graphs in a common representation. This is a difficult problem because there is no explicit labeling of the parts. To find a common labeling, a measure of similarity is required; for example the graph edit distance provides a well defined way of measuring the similarity of two graphs. As an example, Sanfeliu and Fu[95] employed the concept of graph edit distance, giving separate edit costs for relabeling, insertion and deletion on both nodes and edges. A search is necessary to locate the set of operations which have minimal cost. The literature on graph matching is extensive and we refer the reader to section 2.1.3 of the literature review for a more in-depth discussion. We use the method of Gold and Rangarajan[48] to perform the alignment step. Their method is an optimization based approach that uses two-way assignment constraints to constrain the vertices of both graphs and graduated non-convexity to prevent early convergence.

Computing distributions of graphs has proved difficult because of their non-vectorial nature. Operations which are simple in a vector space, such as computing the mean, become more problematic on graphs. Recently, the construction of graph means and medians has received significant attention [42, 43, 41]. However, the construction of the most representative graph median, the generalized median, remains a computationally expensive procedure. Graph medians have been covered extensively in the literature review and so we refer the reader to section 2.1.5 for a complete discussion.

As discussed earlier, we require a way of representing graphs in a vectorial space so we can make use of standard vectorial methods for constructing generative models. There are a number of methods in the literature for representing graphs in a vector space. Caelli and Kosinov[22] have used properties of the spectral decom-

position to represent graphs and Shokoufandeh et al[101] have used the eigenvalues of shock graphs to index shapes. Wilson et al [123] have shown how permutation invariant polynomials can be used to derive features which describe graphs and make full use of the available spectral information. Riesen et al [90] have used graph dissimilarity measures to describe a graph embedding approach. However, it is difficult, if not impossible, to reconstruct graphs from these representations.

The closest work in the literature to the methods presented here are the papers of Luo et al [75] and Xiao and Hancock [126]. Luo et al [75] directly exploit the adjacency matrix by converting it into a long-vector. An initial correspondence step is used to align the adjacency matrix and the long-vectors are analyzed using the eigenmodes. Xiao and Hancock [126] have used the eigenvalues and eigenvectors of the heat kernel to construct the requisite vectors before constructing a normal distribution in the vector space. There are however problems with accurately reconstructing new graphs with these methods.

An alternative method of constructing a generative model for graph structure has been described by Torsello [102]. Based on an importance sampling framework, the existence of each vertex is modeled as a Bernoulli trial. However, this requires the assumption that vertices and edges are independent and therefore the foundations of variations in graph structure, the co-occurrences of edges and vertices, are not modeled. Furthermore, variations in graph structure can only be produced by removing vertices from the model graph. This means that a vertex is required in the model for every vertex in the sample graphs, whether that vertex is key to the structure represented in the set or just noise.

Torsello and Dowe [103] extend the model and correspondence step using an EM like approach. As above they model the existence of each vertex as a Bernoulli trial but to relax the requirement that every vertex must be represented in the model graph, they introduce a two part model. This consists of a structural part of the model, that represents the core structural variations in the set, and a noise model, that allows the generation of vertices that do not correspond to any vertex in the core model graph. They also improve on the previous approach by the inclusion of attributes on the vertices and edges.

While it is possible to sample from both these models, the lack of information describing the co-occurrences of edges or vertices makes generation of examples

drawn from the input distribution unlikely. These methods for constructing generative models of graphs are discussed in detail in section 2.1.6.

The remainder of this chapter is laid out as follows. In section 4.2 we outline the steps required in constructing our generative model. Before describing our generative models we first give our alignment procedure in section 4.2.1. In section 4.2.2 we describe the first of our generative models while in section 4.2.3 we consider the more advanced models based on manifolds. In section 4.3 we give experimental results of the performance of the various models on two types of synthetic data and real world data from the COIL data set. Finally, in section 4.4 we give our conclusions and suggest some directions for future work. A table of the symbols used in this chapter is given in section 4.5 on page 95.

4.2 Method

There are essentially four steps in constructing our generative models for graphs. Firstly, since graphs are defined as relational structures and do not have a canonical description, we must find a common space in which to represent the graphs in our data set. This can be achieved by *graph alignment* which is discussed in section 4.2.1. Secondly, we need a *representation* of the graph in this space, in practical terms this means a matrix or vector representing the graph relations. There are a number of possible representations, such as the adjacency matrix or Laplacian. Matrix representations for graphs were discussed in section 3.2.

These first two steps can be formalized as follows. Commencing from a set $\mathcal{S}$ of sample graphs, we construct a matrix representation $\mathbf{X}_k$ for each $S_k \in \mathcal{S}$. We then perform our alignment step resulting in an aligned matrix representation $\mathring{\mathbf{X}}_k$. From this we construct one or more long vectors $\mathbf{x}_k$ for each sample graph which will be used as input for constructing the model.

The next step is to define a *parameterized distribution* in the representation space which can be learned from sample vectors. Here we use a normal distribution on the vector space defined by a vectorization of the graphs. We discuss these distribution in section 4.2.2. However, some of the representations cannot be represented directly in vector space; we propose a solution to this problem in section 4.2.3. Finally, if we want to use the generative model in the forward sense, we

need a way of *reconstructing graphs* from vectors drawn from this distribution. This is discussed throughout the following sections as different representations require different reconstruction steps.

4.2.1 Alignment

Clearly, forming a vector representation directly from a graph is unsatisfactory, since we do not know the order of the vertices and it is likely that equivalent vertices come in different orders in the different sample graphs. Because of this, the information contained in a graph may appear in different places from sample to sample. To overcome this problem, we must perform an alignment step to bring each sample graph into the same order. Given a set of graphs $\mathcal{S}$ we align the graphs to some reference graph $R \in \mathcal{S}$. The choice of reference graph in this case is the largest graph in the set, since this provides the most information for the other graphs to align with.

$$R = \underset{S_k \in \mathcal{S}}{\operatorname{argmax}} |S_k| \tag{4.1}$$

If the graphs differ in size then we first pad them all to the size of R. We then match all graphs in the set to the reference graph using the algorithm of Gold and Rangarajan [48]. Finally we use the matching order to permute the vertices into the same order as the reference graph.

It is worth noting at this point that the alignment is a fundamental part of the process for graphs, and not merely a preprocessing step. If we wish to compute the probability of a new graph according to our model, we must first align it to the reference graph before using the model; the reference graph R is therefore retained and is part of the model. In a classification problem, for example, the test graph must be aligned to the reference graphs for each of the classes in turn. This means that the class distributions reside in different spaces. Also note that if a graph is presented to the system to be classified which is larger than the reference graph, then information will be lost when it is aligned to the model.

4.2.2 Basic Graph Distributions

The next step is to define a distribution which models the sample graphs. There are a number of ways to do this, which depend on the nature of the graph data we are interested in modeling. These models are based on the different matrix representations which were described in section 3.2.

The Adjacency Model

The simplest vector representation of a graph comes from vectorizing the aligned adjacency matrix $\mathring{\mathbf{A}}_k$ for each graph $S_k \in \mathcal{S}$, i.e.

$$\mathbf{x}_k = \mathrm{vec}(\mathring{\mathbf{A}}_k) \tag{4.2}$$

where the dimension of the long vectors $\mathbf{x}_k$ is n. This is essentially the model adopted in [75], although our alignment method is different to the one used in that paper. Adjacency matrices are symmetric and have zero diagonal. Since matrix addition preserves these properties, adjacency matrices naturally form a Euclidean vector space in which we can use the normal formulae for mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of the vectors $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{|\mathcal{S}|}\}$.

We then propose a normal distribution in the vector space. This procedure has been applied successfully in the literature in [75] and [126].

$$p(\mathbf{x}_k; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}\sqrt{|\boldsymbol{\Sigma}|}} \exp[-\frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_k - \boldsymbol{\mu})] \tag{4.3}$$

The mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ are computed using the standard formulas:

$$\boldsymbol{\mu} = \frac{1}{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|} \mathbf{x}_k \tag{4.4}$$

$$\boldsymbol{\Sigma} = \frac{1}{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|} (\boldsymbol{\mu} - \mathbf{x}_k)(\boldsymbol{\mu} - \mathbf{x}_k)^T \tag{4.5}$$

Every point in the space spanned by $\boldsymbol{\Sigma}$ represents the adjacency matrix of some weighted graph. In order to sample a graph from this distribution, we must first generate a sample vector drawn from Eqn. 4.3. In practice this means performing PCA on the vector space by obtaining the eigendecomposition of the covariance matrix

as $\mathbf{\Sigma} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T$. We then draw a new parameter vector $\mathbf{b}$ where each component is generated from a 1D normal distribution with variance given by the corresponding diagonal element of $\mathbf{\Lambda}$:

$$b(i) \sim \mathcal{N}(0, \Lambda(i,i)) \tag{4.6}$$

The generated vector is then:

$$\mathbf{x}' = \boldsymbol{\mu} + \mathbf{\Phi}\mathbf{b} \tag{4.7}$$

We can then generate a new graph by devectorizing $\mathbf{x}'$, but because of the continuous distribution we will obtain a matrix with non-$\{0,1\}$ entries. This is sufficient if we are considering weighted graphs, but if we wish to reconstruct discrete graphs we must define a thresholding procedure. As discussed earlier there is also the option of interpreting the values as edge probabilities but this ignores edge co-occurrences. We make use of the thresholding function defined in 3.3.2, and recover the new adjacency matrix $\mathbf{A}'$ as follows:

$$\mathbf{A}' = \theta[\mathrm{devec}(\mathbf{x}')] \tag{4.8}$$

We call this model the *Adjacency Model* (A).

The Laplacian model

The graph may also be represented by a Laplacian. The Laplacian model is identical to the adjacency model with the substitution of the aligned Laplacian matrix $\mathring{\mathbf{L}}_k$ for the adjacency matrix $\mathring{\mathbf{A}}_k$. As before, if we wish to recover discrete graphs then we require a thresholding step. The threshold function θ is reused but the parameter β is altered to account for the different values allowed in the Laplacian matrix. We refer to this as the *Laplacian Model* (L).

However, Laplacians do not reside in a vector space. Laplacians are symmetric positive semidefinite (PSD) matrices, but a linear combination of two such matrices does not necessarily give a PSD matrix. As a result, not all points in the distribution represent legitimate Laplacians for some weighted graph. An alternative to the straightforward model is to describe Laplacian matrices on the manifold of positive definite (PD) matrices. This is discussed in section 4.2.3. We will provide an

experimental comparison later to determine the importance of these issues.

The Spectral Model

Spectral representations have attracted considerable recent interest[123, 101, 74, 22]. In certain circumstances, this type of representation seems to have advantages over the adjacency or Laplacian matrices. Specifically, spectral analysis reveals sets of vertices which are grouped together by many interconnections. Many types of graphs which occur in practical problems have this kind of grouping property; for example, local parts of an object are inter-related and do not change much between different samples, although at the global level there may be significant changes. We refer to these type of graphs as "structural". This is to be contrasted with random graphs, where edges are chosen randomly between pairs of vertices and there is no local structure to the graphs. We would anticipate that spectral representations will be effective for structural graphs but not for random graphs.

In the spectral representation, we use a spectral decomposition of the graph structure into eigenmodes, and then model the variation in the eigenmodes. The concept behind the spectral approach is that the decomposition represents the graph in spectral modes which represent significant structures in the graph. These structures may be more stable under realistic noise models of graphs in typical problems. We commence with the aligned normalized Laplacian $\mathring{\mathbf{N}}_k$ of a graph S_k and then find the eigendecomposition of the graph:

$$\mathring{\mathbf{N}}_k = \mathbf{V}_k \mathbf{E}_k \mathbf{V}_k^T \tag{4.9}$$

As described in the chapter on mixing spectral representations of graphs, the eigenvectors given by the eigendecomposition are produced up to a sign factor of ± 1. If left uncorrected this noise would destroy any model we fit to the vectors. Therefore, we must apply a *sign correction* step. We use the same method as that described in section 3.3.1 which is based on correcting the component of each eigenmode that would cause the largest error.

In Xiao and Hancock [126] the eigenvalues and vectors were combined by considering a single matrix based on the heat kernel. Here, we simply use the matrix $\mathbf{V}\mathbf{E}^{\frac{1}{2}}$ produced from the normalized Laplacian to represent the graph. This matrix

can be vectorized as such:

$$\mathbf{x}_k = \mathrm{vec}(\mathbf{V}_k \mathbf{E}_k^{\frac{1}{2}}) \tag{4.10}$$

The model then proceeds as above by constructing a normal distribution over the vector set. To recover generated graphs in this representation we apply the threshold function (equation 3.18) to the generated matrix. We must modify the function to handle the different values that appear in normalized Laplacian matrices. The threshold parameter β used in the threshold function can be set by considering the average off diagonal value in the normalized Laplacian matrices of the sample graphs. We term this model the *Spectral Model* (S).

The Dual Spectral Model

We have found that the spectral model will not lead to a satisfactory vector space, since combining the eigenvalues and eigenvectors in this way distorts the distribution of the eigenvalues. This can be seen by mixing the eigenvectors from two slightly different graphs. If we average these vectors and they are not pointing in the same direction, the resultant length will be shorter than the lengths of the originals. Since the eigenvalue is encoded in the length, this results in smaller eigenvalues and unsatisfactory reconstructions.

An alternative is to model the eigenvalues and eigenvectors separately. We define two vector spaces, one for the eigenvalues and one for the eigenvectors. For each graph $S_k \in \mathcal{S}$ we compute two vectors $\mathbf{x}_{Ek}$ and $\mathbf{x}_{Vk}$ as follows:

$$\mathbf{x}_{Ek} = \mathrm{diag}(\mathbf{E}_k) \tag{4.11}$$

where $\mathrm{diag}(\mathbf{X})$ produces a vector of the main diagonal of a matrix $\mathbf{X}$. The second vector space is given by the eigenvector matrix:

$$\mathbf{x}_{Vk} = \mathrm{vec}(\mathbf{V_k}) \tag{4.12}$$

We therefore have two sets of long vectors, one based on the eigenvalues $\{\mathbf{x}_{E1}$, $\mathbf{x}_{E2}$, ..., $\mathbf{x}_{E|\mathcal{S}|}\}$ and a second based on the eigenvectors $\{\mathbf{x}_{V1}, \mathbf{x}_{V2}, ..., \mathbf{x}_{V|\mathcal{S}|}\}$. We then create two normal distributions over these vector sets with $\boldsymbol{\mu}_E, \boldsymbol{\Sigma}_E$ and $\boldsymbol{\mu}_V, \boldsymbol{\Sigma}_V$ respectively.

In order to be a valid decomposition, it is clear that any eigenvector matrix should always be an orthogonal matrix. This is a similar problem to the one identified with the Laplacian model. In fact these matrices lie on the manifold of orthogonal matrices and not in vector space. It is not clear that it is a good thing to represent them in a vector space. For example, neither the mean nor a generated sample will be orthogonal and will need conditioning before it can be used to reconstruct a graph. In order to sample a graph, we draw new vectors $\mathbf{x}'_E, \mathbf{x}'_V$ from the normal distributions as before.

The elements of $\mathbf{x}'_E$ are the diagonal elements of the diagonal eigenvalue matrix $\mathbf{E}'$. We recover the generated long vector describing the eigenvectors as such:

$$\mathbf{V}' = \mathrm{devec}(\mathbf{x}'_V) \tag{4.13}$$

We condition the generated eigenvectors using an orthogonalization step which produces an orthogonal set of vectors.

$$\mathbf{V}'' = (\mathbf{V}'\mathbf{V}'^T)^{-\frac{1}{2}}\mathbf{V}' \tag{4.14}$$

We normalize the orthogonal vector set and then reconstruct the graph by recovering the normalized Laplacian matrix. This matrix is thresholded using θ:

$$\mathbf{N}' = \theta[(\mathbf{V}'')\mathbf{E}'(\mathbf{V}'')^T)] \tag{4.15}$$

We will refer to the this spectral model with two distributions as the *Dual Spectral Model* (DS).

4.2.3 Manifold Graph Distributions

Many types of data are constrained in some way. A classic example of this is given by surface normals - these are vectors for which the direction is important but which should always be unit length. Unit vectors form a subspace or manifold of normal vector space, since only unit vectors are allowed. This manifold is non-linear or "curved" since a linear combination of unit vectors does not always produce another unit vector. It is difficult to model the distributions of data on these curved manifolds as the distributions no longer have simple forms and it is hard to compute correct

statistics.

One approach that can be applied when the manifold is a hypersphere (as in the case of unit vectors) is to use directional statistics to construct the distribution directly on the manifold. Distributions such as the Fisher-Bingham distribution[78] are capable of modeling data on this type of manifold but the estimation of model parameters [64] becomes problematic when working with high dimensional data.

The exponential map offers a different solution to these problems by providing a projection from the manifold onto a linear vector space. This vector space is in fact the tangent space to the manifold at a particular point. The projection has the following very useful properties: a) It is bijective[1], which is essential for reconstructing new graphs and b) The geodesic distance from the point of projection across the manifold to another point is equal to the Euclidean distance from the center to the projection in the tangent space. As a result, if the data mean is the center of projection, then we can compute central moments (such as the covariance matrix) in the normal way using positions in the tangent space. We can then use a standard distribution such as the normal distribution in the tangent space. The map requires two elements, firstly the *log* map, which projects a point $\mathbf{v}$ from the manifold to the tangent vector space resulting in a vector $\mathbf{x}$:

$$\mathbf{x} = \log_{\mathbf{m}} \mathbf{v} \tag{4.16}$$

where $\mathbf{m}$ is the center of projection. Note that this formula is just notation, the actual solution would be computed using geometry. Secondly, the *exp* map goes from tangent space back onto the manifold:

$$\mathbf{v} = \exp_{\mathbf{m}} \mathbf{x} \tag{4.17}$$

With these ingredients, we can compute the geodesic mean, which is the mean of the points on the manifold. This is defined as:

$$\mathbf{m}^* = \arg\min \sum_i d^2(\mathbf{m}, \mathbf{v}_i) \tag{4.18}$$

where $\mathbf{v}_i$ is a sample point and $d(\cdot, \cdot)$ is the distance on the manifold. This can

[1]If the manifold has a cut locus then technically the mapping is only bijective if a) the cut locus is avoided and b) we only work within the tangential cut locus in the tangent space.

be computed via the following iterative formula:

$$\mathbf{m}_{t+1} = \exp_{\mathbf{m}_t} \frac{1}{|\mathcal{S}|} \sum_i \log_{\mathbf{m}_t} \mathbf{v}_i \tag{4.19}$$

The iteration stops when the new mean $\mathbf{m}_{t+1}$ is within a certain tolerance of the old mean $\mathbf{m}_t$. With the mean to hand, the covariance matrix (on the tangent plane) is simply:

$$\mathbf{\Sigma} = \frac{1}{|\mathcal{S}|} \sum_i (\log_{\mathbf{m}} \mathbf{v}_i)(\log_{\mathbf{m}} \mathbf{v}_i)^T \tag{4.20}$$

One way to use this map to extend our dual spectral model would be to enforce the property that generated eigenvectors are of unit length. We first note that eigenvectors reside on the manifold of unit length vectors. We can then use the exp and log maps to define a mapping between the manifold of unit length vectors and a linear tangent space.

To proceed, we take the eigenvector from mode i from each sample graph and construct a mapping to a tangent space. This tangent space is based about the mean vector from that mode which can be computed using the iterative approach above. With all the eigenvectors from mode i projected into the linear tangent space, we can use standard statistical methods to construct a normal distribution over the points in the tangent space. After a vector has been sampled from this distribution, it is projected back onto the manifold of unit length vectors thereby ensuring it has the required unit length property.

However, this approach will only construct a set of unit length vectors and an orthogonalization step will still be required. Indeed, we have found that this approach adds little to the model and ideally we require a set of orthogonal eigenvectors to be generated directly. We discuss such an approach next.

The Orthogonal Map

We first note that the eigenvector matrix $\mathbf{V}$ of a sample graph resides on the manifold of orthogonal matrices. The group of *special orthogonal* matrices, $SO(n)$ is a well studied group because of its importance in physics, and an exponential map is already known for this group. Our matrices are orthogonal, with $\mathbf{V}\mathbf{V}^T = \mathbf{I}$ and not special orthogonal, which carries the extra condition $|\mathbf{V}| = 1$, but by flipping the

sign of one of the eigenvectors in $\mathbf{V}$ we can easily enforce this condition.

Now, if we let $\mathbf{X}$ be a real antisymmetric matrix, then $\mathbf{V} = \exp(\mathbf{X})$ is a special orthogonal matrix since:

$$\mathbf{V}\mathbf{V}^T = \exp(\mathbf{X})\exp(\mathbf{X}^T) = \exp(\mathbf{X} + \mathbf{X}^T) = \mathbf{I} \tag{4.21}$$

In fact, the space of antisymmetric matrices is the Lie algebra for the space of orthogonal matrices. For this manifold, the exponential and logarithm maps[86] are:

$$\mathbf{X} = \log_{\mathbf{M}} \mathbf{V} = \log \mathbf{M}^T\mathbf{V} \tag{4.22}$$

$$\mathbf{V} = \exp_{\mathbf{M}} \mathbf{X} = \mathbf{M} \exp \mathbf{X} \tag{4.23}$$

The geodesic mean of the projection $\mathbf{M}$ can be computed using a formula similar to the iterative one detailed above:

$$\mathbf{M}_{t+1} = \exp_{\mathbf{M}_t} \frac{1}{|\mathcal{S}|} \sum_i \log_{\mathbf{M}_t} \mathbf{V}_i \tag{4.24}$$

While the matrix exponential is simple to compute, the matrix logarithm has multiple solutions, only one of which is the required real anti-symmetric matrix (the others may be complex). This matrix is not trivial to find; we use the method of Gallier and Xu[46]. Using this map with the dual spectral model gives us the *Dual Spectral, Orthogonal map model* (DSO).

The Positive-definite Map

In section 4.2.2, we modeled the Laplacian of the graph using simple vectorization. This model generates non-Laplacian matrices since the generated matrices are not always PSD. We could solve this problem using an approach similar to that described above for orthogonal matrices. However, the manifold of PSD matrices is not appropriate for this type of modeling since the matrices have no inverses and therefore do not form a continuous group.

A solution to this problem is to instead construct the tangent space on the manifold of PD matrices. We can map Laplacians onto the space of PD matrices by

adding a small diagonal offset, $\mathbf{L} \rightarrow \mathbf{L} + \epsilon\mathbf{I}$. In effect, this offset is related to the cost of padding a graph with extra vertices.

The exponential and logarithm maps for the manifold of PD matrices[86] is:

$$\mathbf{X} = \mathbf{M}^{\frac{1}{2}} \log \left[\mathbf{M}^{-\frac{1}{2}}(\mathbf{L} + \epsilon\mathbf{I})\mathbf{M}^{-\frac{1}{2}}\right] \mathbf{M}^{\frac{1}{2}} \tag{4.25}$$

$$\mathbf{L} = \mathbf{M}^{\frac{1}{2}} \exp \left[\mathbf{M}^{-\frac{1}{2}}\mathbf{X}\mathbf{M}^{-\frac{1}{2}}\right] \mathbf{M}^{\frac{1}{2}} - \epsilon\mathbf{I} \tag{4.26}$$

The geodesic mean $\mathbf{M}$ on this manifold can be computed using equation 4.24 with the formulas for the exp and log maps replaced by those in equations 4.25 and 4.26.

Applying this map to the Laplacian model gives us the *Laplacian, positive definite map model* (LM).

4.3 Experimental Results

We evaluate the generative graph models against a number of criteria, using both synthetic data and graphs derived from images. In particular, we examine the classification performance, basis restriction error, and the distance distributions, as well as visualizing the generated distribution of graphs. We evaluate these criteria for all the generative models described above. The adjacency model (A) is the straightforward vector space of $\mathbf{A}$. The Laplacian model (L) is the vector space of $\mathbf{L}$, whereas the Laplacian map model (LM) uses the PD logarithmic map before defining the vector space. The spectral models use the spectral decomposition of the normalized Laplacian for modeling. The spectral model (S) uses a single vector space for both the eigenvalues and eigenvectors, whereas the dual spectral model (DS) uses separate vector spaces for each element. Finally, we have the model which includes the exponential map on the eigenvector matrix; the dual spectral model with orthogonal matrix mapping (DSO).

4.3.1 Data Sets

To assess the effectiveness of our models we use three different data sets each containing a different type of graph.

Random Data Set

The random graph data set is created using a reference graph of 20 vertices with pairs of vertices connected at random with probability 0.6. We then create a number of edited versions of the reference graph by swapping edges and non-edges at random with a probability which varies with the level of error we wish to produce. For the classification experiments, this edit rate varies and for the other experiments it is 0.04. On average, a graph has 114 edges and 4.6 edits in this case.

Delaunay Data Set

The Delaunay data set uses 20 randomly selected 2D points which are connected via their Delaunay graph. To vary the graphs, we move a fixed number of points to a random location in the space and then recompute the Delaunay graph. For one moved point, the average number of edges is 49 and the average number of edge edits is 4.37.

COIL Data Set

The COIL database consists of images of a number of objects, with 72 different views of each object. We find the corner points in the image and form a graph from them using a Delaunay triangulation[125]. In our experiments, we use four objects which are shown with their associated Delaunay graphs in figure 4.1. The pairs chosen for our classification experiments are of similar sizes and have similar numbers of edges, in order to provide a difficult classification problem. These pairs are objects 8 and 16, and objects 4 and 13.

4.3.2 Classification

Our first investigation uses the generative models in a traditional way as a classifier. We use a Gaussian classifier with the distribution for each class defined as in equation 4.3. The mean and covariance is estimated from a training set for each class and new examples classified according to the most probable class. There is one additional complication due to the alignment problem; each test sample must first be aligned to the reference graph for a particular class and, in the case of the spectral methods, sign-corrected before computing the class probability.

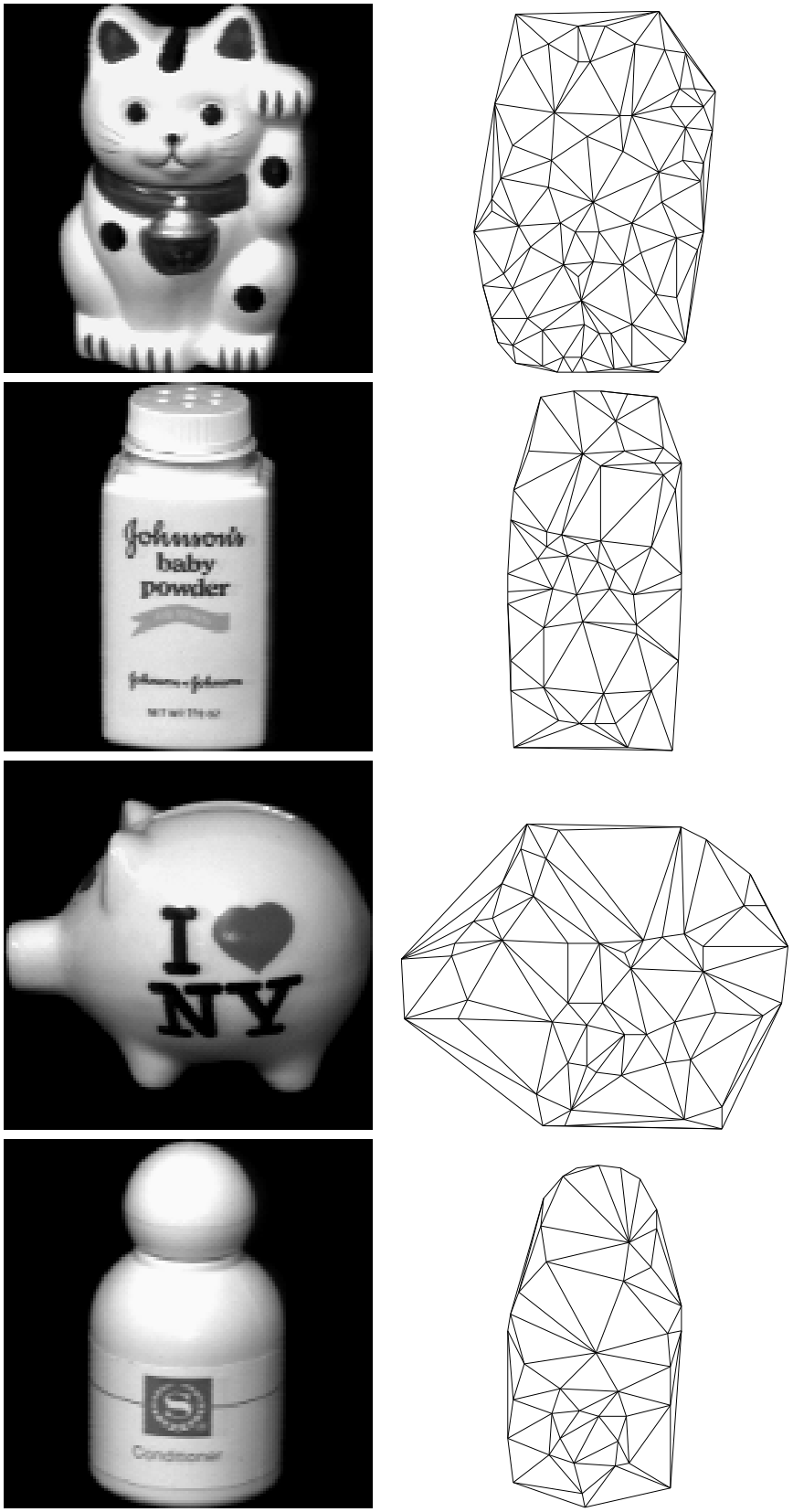


Figure 4.1: COIL data used: Objects 4,8 13 and 16. A image from each set is shown on the left and the resulting Delaunay graph is shown on the right.

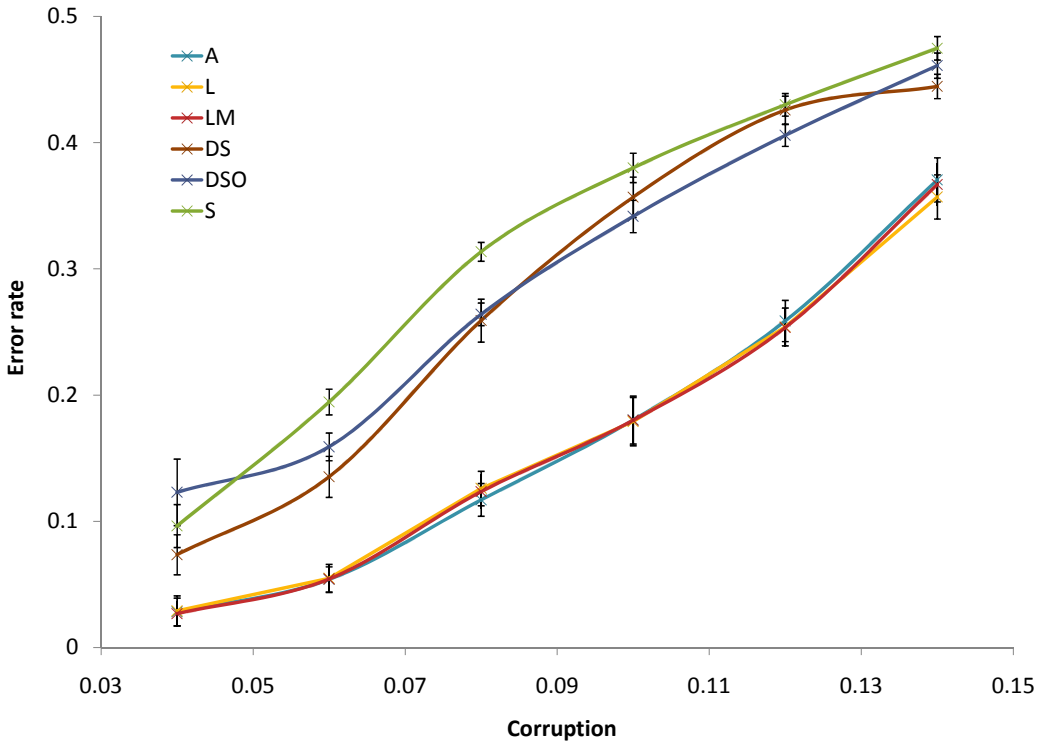


Figure 4.2: Classification results using the random data set.

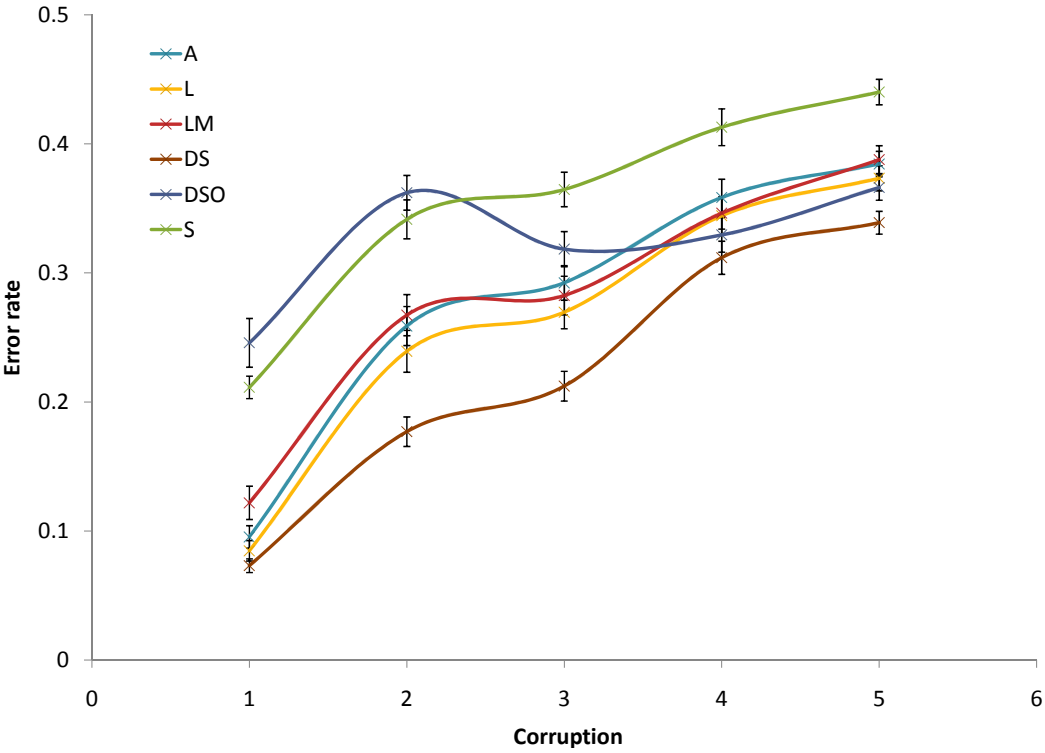


Figure 4.3: Classification results using the Delaunay data set.

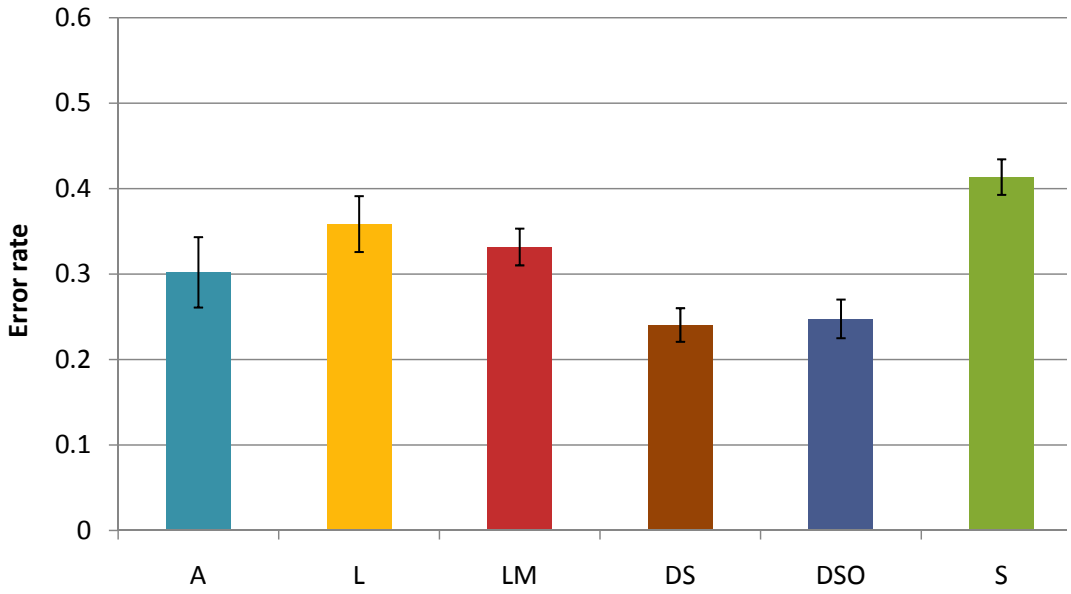


Figure 4.4: Classification results using graphs from objects 8 and 16 from the COIL data set.

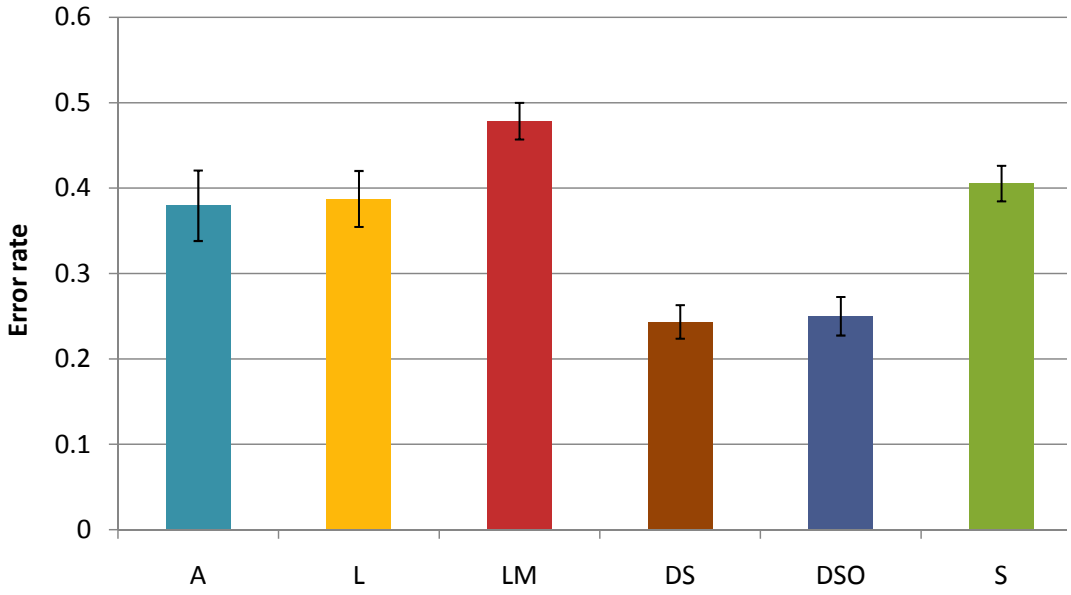


Figure 4.5: Classification results using graphs from objects 4 and 13 from the COIL data set.

In the case of the synthetic data, we perform the training with 100 graphs and produce another set of 200 graphs to test the trained models. We introduce increasing levels of corruption into the graphs of the synthetic data sets to understand how effective our models are at handling noise. This corruption is provided by the methods discussed in 4.3.1. For the COIL data set, we do not vary the level of corruption and since we do not have the opportunity to generate new data, 10-fold cross-validation is used to obtain estimates of the error rate.

In figure 4.2 we give the classification results for the random data set. In this data set the non-spectral models perform very similarly and significantly better than any spectral model. Of the spectral models, DS and DSO have approximately equal performance while S is generally the worst model.

Figure 4.3 shows the classification results for the Delaunay data set. The non-spectral methods are still grouped and we see slightly worse performance from all three models. The dual-spectral model provides the best classification accuracy, outperforming all other methods across all levels of corruption. Adding the orthogonal map to this model seems to hinder classification performance and produces unexpected results at low levels of corruption. In fact, the performance when 3, 4 or 5 points are moved is better than when only 2 points are moved. However, the observed classification accuracy across all models does seem to be worse than the trends suggest it should be at a corruption of 2 points.

The classification results for the COIL data set are shown in figures 4.4 and 4.5. In the first experiment (objects 8 and 16) the non-spectral models produce an error rate in the range of 0.3 to 0.35. Adding the PD logarithmic map to the Laplacian model reduces the error slightly. In the second experiment (objects 4 and 13) the error rates for the non-spectral models are significantly worse, with both A and L achieving an error rate of approximately 0.38. Using the LM model here reduces the classification accuracy compared to the standard Laplacian model and results in a very poor error rate of almost 0.5. On the other hand, the spectral models DS and DSO achieve a low, constant error rate on both experiments. The model based on the single spectral matrix produces consistently poor results throughout both experiments.

There are a number of conclusions we can draw from these results. For random graphs, the non-spectral models perform best; the A, L and LM models all perform

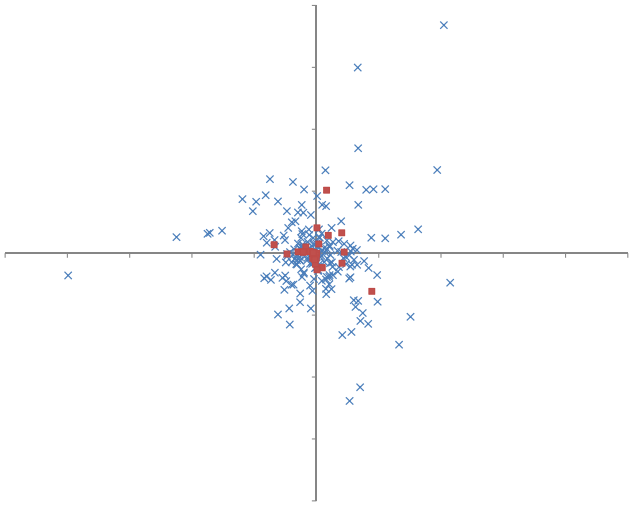
similarly. The spectral models do not perform as well. This is unsurprising since our initial hypothesis was that the spectral decomposition may represent preserved parts of the graph better; the random editing procedure has no regard for structure in the graph and seems to generate a set of connections which behave like a random vector. This situation is reversed with the Delaunay graphs; the edit operation here will preserve some parts of the graph while changing others. Here the DS model is superior although adding the orthogonal map produces some unexpected results. In the COIL data set, the data is even more structured, with parts of the graph remaining relatively invariant as a particular view of an object remains in the image. Here both the DS and DSO models perform similarly and very well. In all cases the S model does not perform well.

4.3.3 Distributions

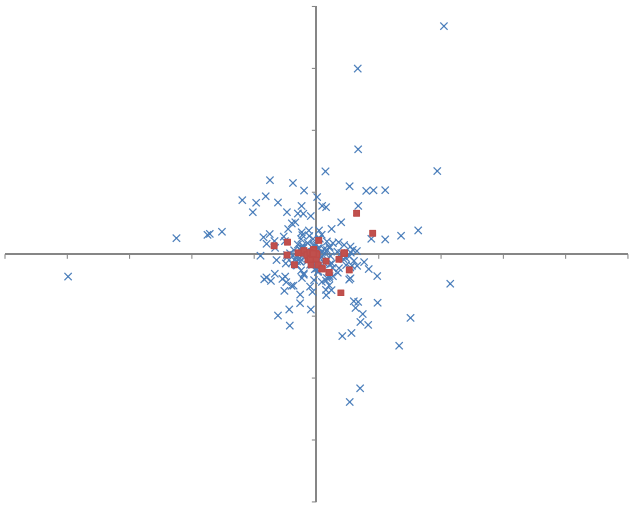
We show distributions for all six models on the random and COIL data sets. This allows us to make a visual check on the form of the distributions. Both the sample graphs and a set of generated graphs are shown for each model to see if the generated graphs approximately match the distribution of sample graphs.

The projection space for each data set is established by performing PCA on the adjacency matrices of the sample graphs. Generated graphs are reconstructed back to an adjacency matrix and are also projected into this space. With the exception of model A, it is important to make the distinction that we are not visualizing the vector spaces that the models are based in, we are only visualizing the sample and recovered adjacency matrices.

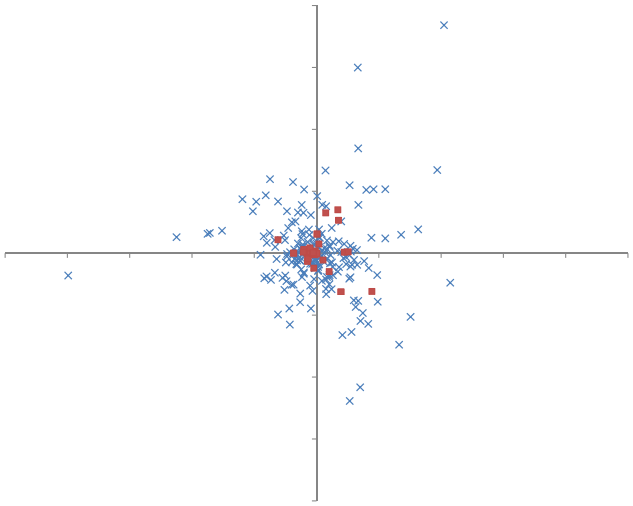
We now present figures showing the distributions. In all cases blue crosses indicate sample graphs and red squares indicate generated graphs. We begin with figure 4.6 which shows the distributions of the non-spectral models on the random data set. In these plots, generated graphs lie within the sample distribution but do not fill the space. On the other hand, when the DS and DSO models are applied to this data set the generated graphs fill the space of sample graphs much more effectively. The spectral results for the random data set are shown in figure 4.7. Graphs generated using the S model fall outside the distribution of sample graphs. This is due to the eigenvalue magnitude problems mentioned earlier which shift the generated distribution away from the sample distribution. In this data set the DSO



Random Data - Model A

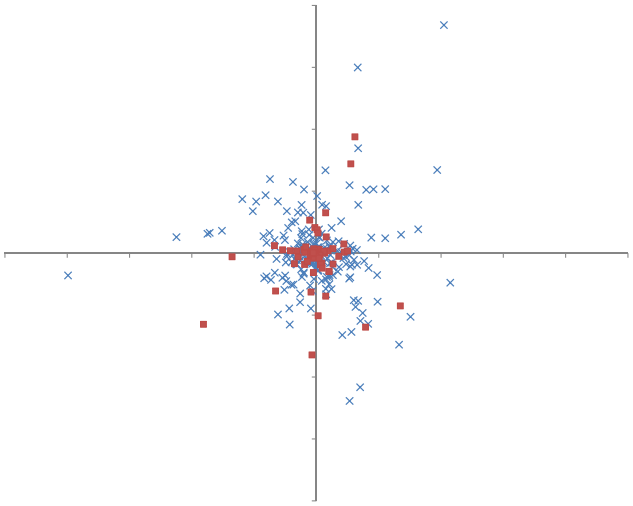


Random Data - Model L

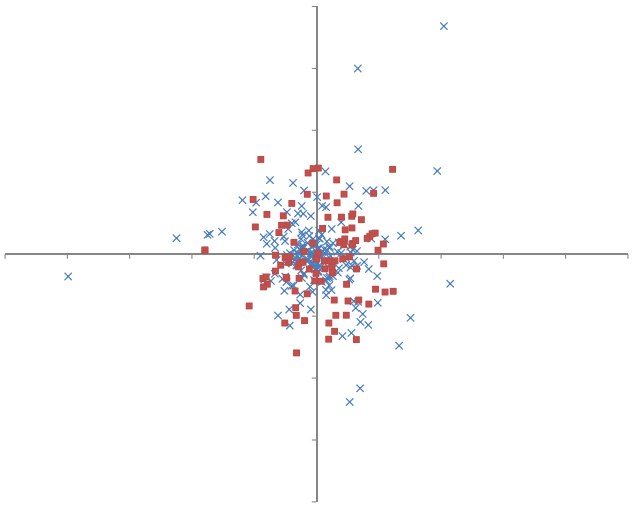


Random Data - Model LM

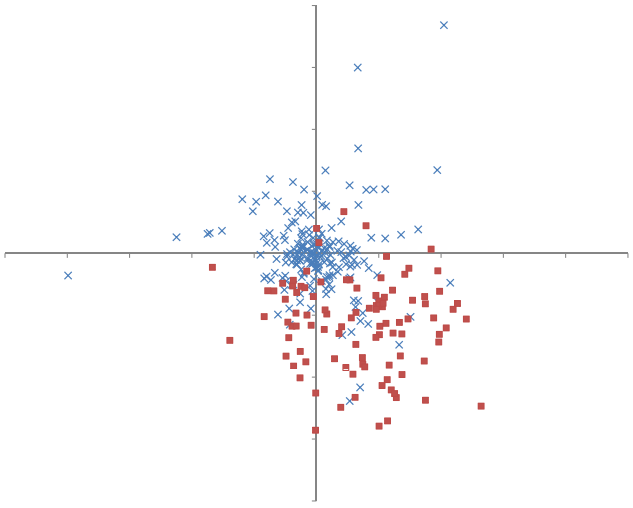
Figure 4.6: Distributions of the non-spectral models on the random data set. Blue crosses indicate sample graphs and red squares indicate generated graphs.



Random Data - Model DS

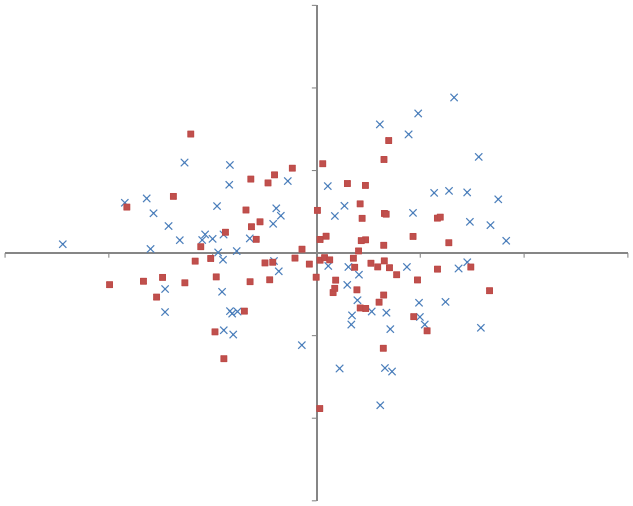


Random Data - Model DSO

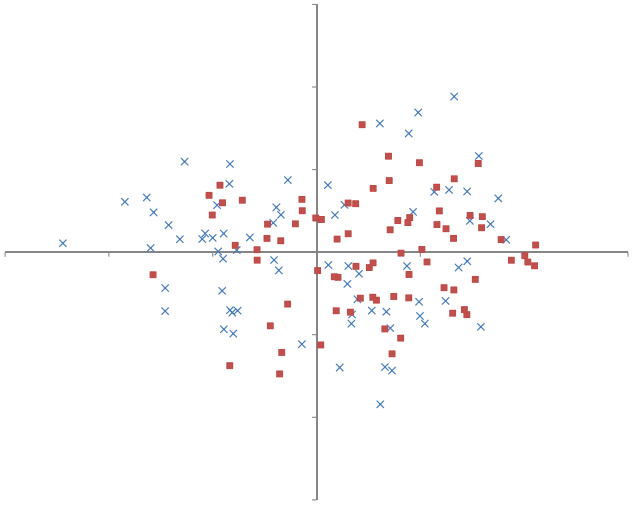


Random Data - Model S

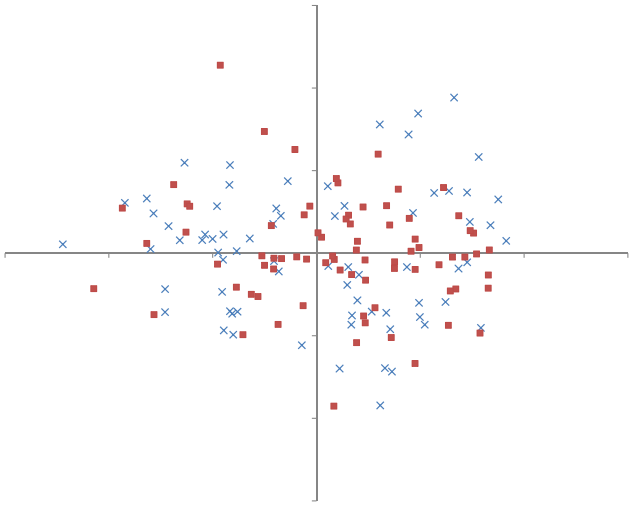
Figure 4.7: Distributions of the spectral models on the random data set. Blue crosses indicate sample graphs and red squares indicate generated graphs.



COIL Data - Model A

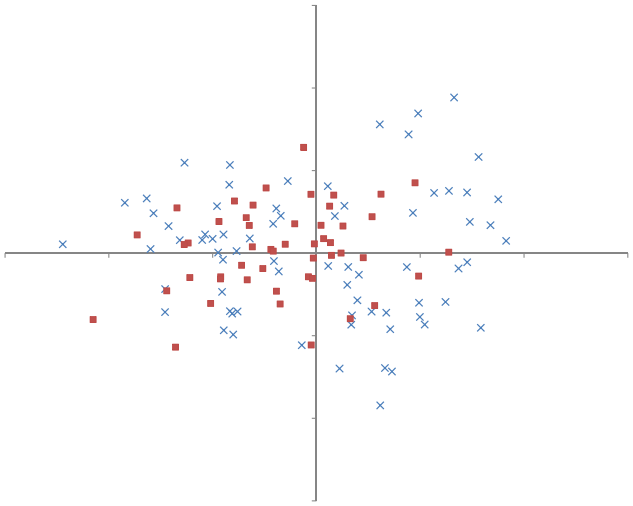


COIL Data - Model L

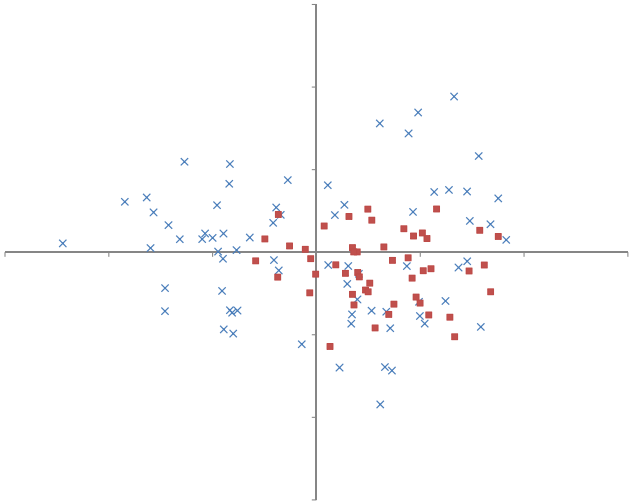


COIL Data - Model LM

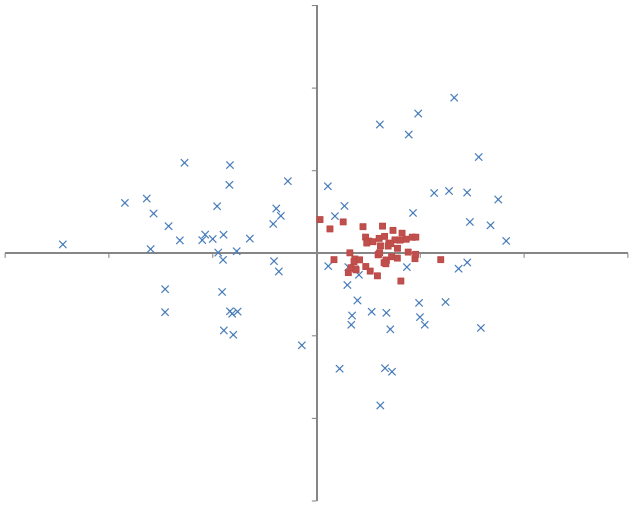
Figure 4.8: Distributions of the non-spectral models on the COIL data set. Blue crosses indicate sample graphs and red squares indicate generated graphs.



COIL Data - Model DS



COIL Data - Model DSO



COIL Data - Model S

Figure 4.9: Distributions of the spectral models on the COIL data set. Blue crosses indicate sample graphs and red squares indicate generated graphs.

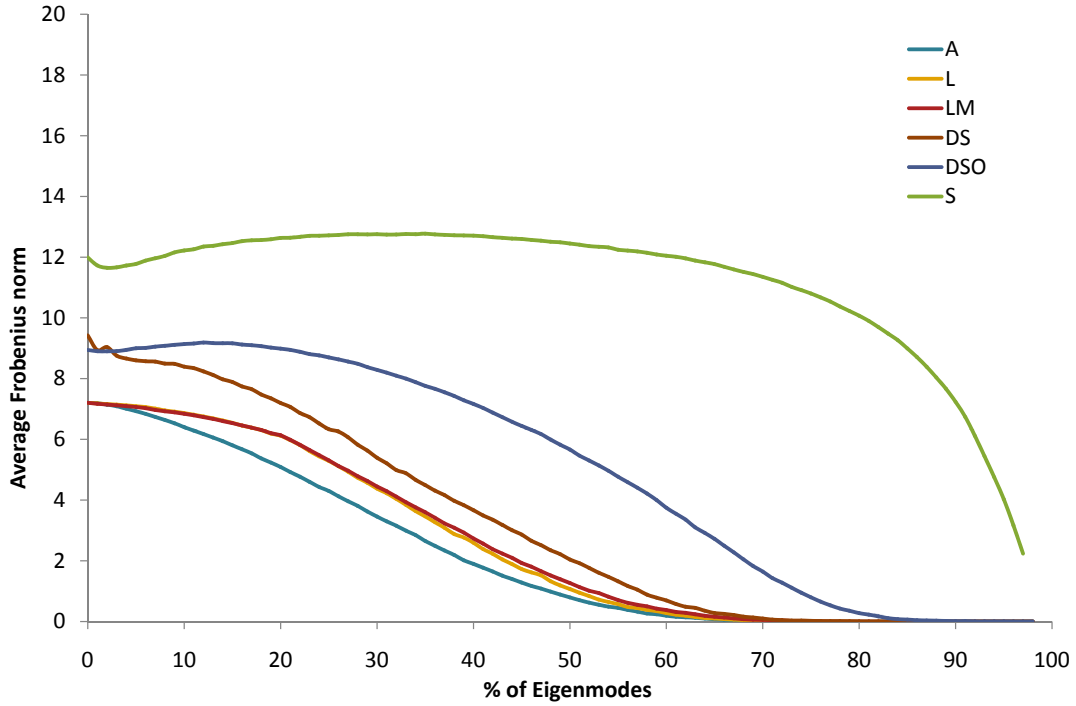


Figure 4.10: Basis restriction error using the random data set.

model appears to offer the set of graphs that most accurately match the distribution of sample graphs.

In figure 4.8 we show distributions for the non-spectral models when applied to the COIL data set. Compared to the random data set, the models appear to be more successful at generating graphs across the whole space of sample graphs. Turning to the spectral models, which are shown in figure 4.9, the generated graph distributions are still within the space of sample graphs but cover it slightly less successfully. Again, using model S results in a poor distribution of generated graphs.

4.3.4 Basis Restriction Error

We can gain some measure of the quality of the distributions by how compact the representation is. This essentially means reconstructing the sample graphs, using only a restricted number of principal components of the distribution. The process involves firstly projecting a sample graph into the vector space, retaining only a limited proportion of the principal components of the model, and then reconstructing the sample graph. The error is the average Frobenius norm between the matrix representations of the initial and reconstructed graphs.

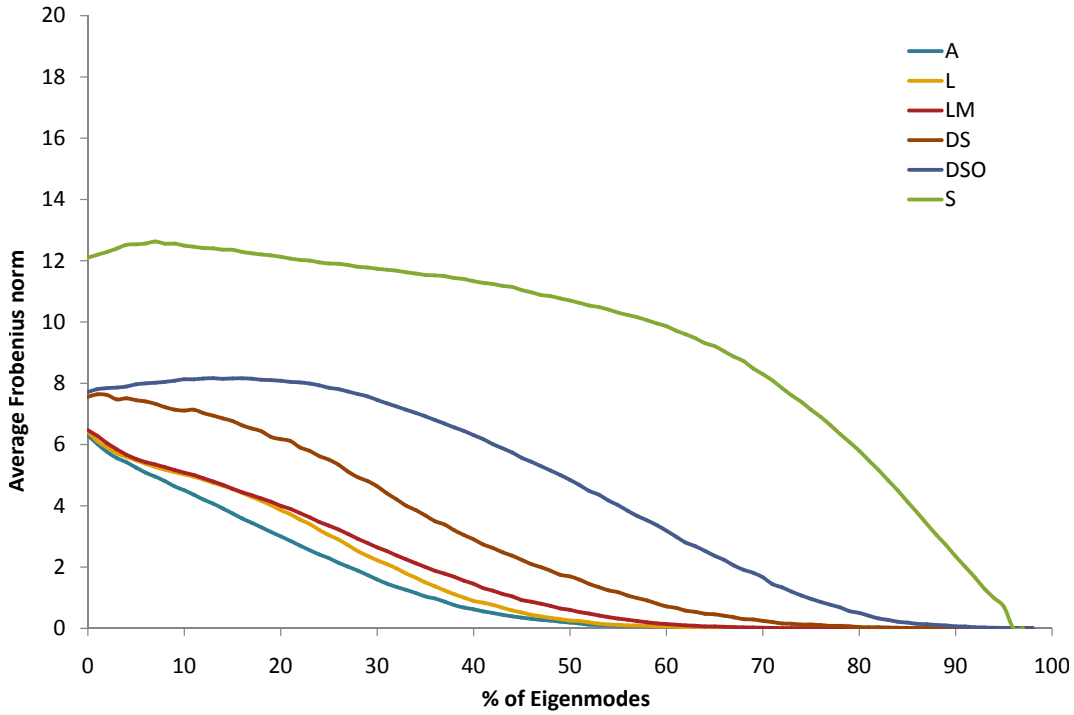


Figure 4.11: Basis restriction error using the Delaunay data set.

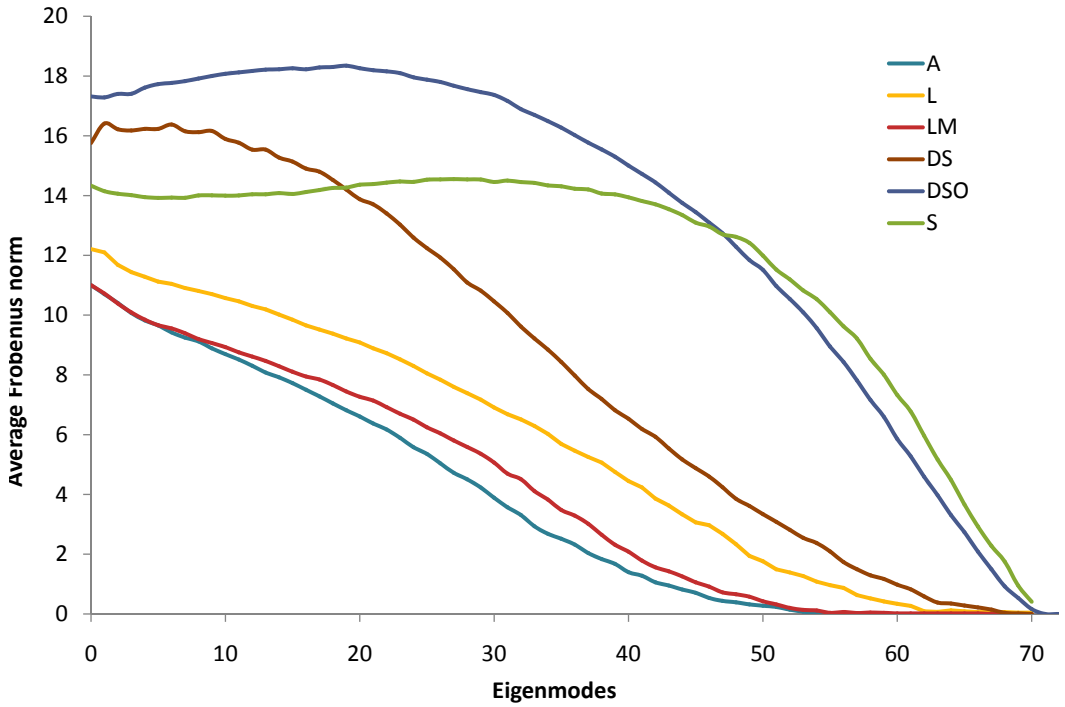


Figure 4.12: Basis restriction error using graphs from object 16 from the COIL data set.

Since we are reconstructing the sample graphs with a limited amount of information, the matrices describing the reconstructed graphs can be very poor descriptions of the graphs they represent. This is especially true in the case of the normalized Laplacian since it has non-discrete off diagonal elements. This makes its recovery almost impossible when the majority of information is removed from the model. To allow the basis restriction tests to be applied to all models, including those based on the normalized Laplacian, we substitute the Laplacian representation for the normalized Laplacian when needed.

The results of the basis restriction tests for the random data set are shown in figure 4.10. Model A results in the most compact model closely followed by L and LM. The DS model is next best although adding the orthogonal map reduces the compactness of the model. The S model is the least compact. The same results are observed for the Delaunay data set in figure 4.11. Some of the errors increase as more information is included in the model, this is seen in DSO and S in both the random graphs and the Delaunay graphs. This is due to problems accurately recovering the matrix representations from the reconstructed vectors and is similar to the problems associated with the normalized Laplacian mentioned above.

In figure 4.12[2] the results of the basis restriction tests are given for object 16 of the COIL data set. The differences in the compactness of the models are less than the previous data sets. Again model A results in a very compact model. Using model L reduces the compactness but applying the PD map improves it almost to the level of model A. The DS model starts quite poorly but improves significantly as more information is included. Models DSO and S are the least compact and improve at a much slower rate than DS.

Over all three data sets, the non-spectral models result in the more compact models. Of the spectral models, only the DS model comes close to matching the compactness of the non-spectral models. Adding the orthogonal map reduces the compactness of the DS model in all cases. Model S is generally the least compact model.

[2]Note the difference in the axis scale for the basis restriction error plots: the plot representing the random and Delaunay data has the percent of all eigenmodes while the COIL data has a fixed number of eigenmodes.

4.3.5 Distance Distribution

Since we are modeling the graphs by normal distributions over some vector space, it is important to establish whether the distributions do in fact conform to a normal distribution. One way to do this is to look at the distribution of standardized distances within each model. The standardized distance is:

$$d(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \tag{4.27}$$

If the distribution is normal, this should be a chi-squared distribution with n degrees of freedom, where n is the number of principal components of the distribution that we use. Clearly we would expect the distribution to become more normal as we reduce the number of components; here we are interested in showing how much the distributions deviate from normality as we fix the number of components. We select the first 10 principle components.

The distance distributions for the random data set are given in figure 4.13. Here we see that the eigenvalues and eigenvectors of the DS and DSO models are very close to normal (middle and bottom graphs). Model S is also close to normal but the non-spectral models deviate slightly from the chi-squared distribution (top graph).

Figure 4.14 shows the distance distribution for the Delaunay data set and we see broadly the same results. All models follow the chi-squared distributions slightly worse than in the random data.

In figure 4.15 we give the results for object 8 from the COIL data set. Here, the non-spectral models perform poorly with the exception of model L which is reasonably close to the chi-squared distribution. Of the spectral models, model S performs the worst while the eigenvalues of the DS and DSO models are quite good. When the eigenvectors are considered, we see that the eigenvectors of the DSO model slightly outperforms the eigenvectors of the DS model.

Throughout the distance distribution tests the spectral models, specifically DS and DSO, have performed very well indicating that the distribution of vectors is closer to normal than those of the non-spectral models. However, this is not as clear in the non-spectral models with the exception of model L which does seem to perform well in the random and COIL data sets.

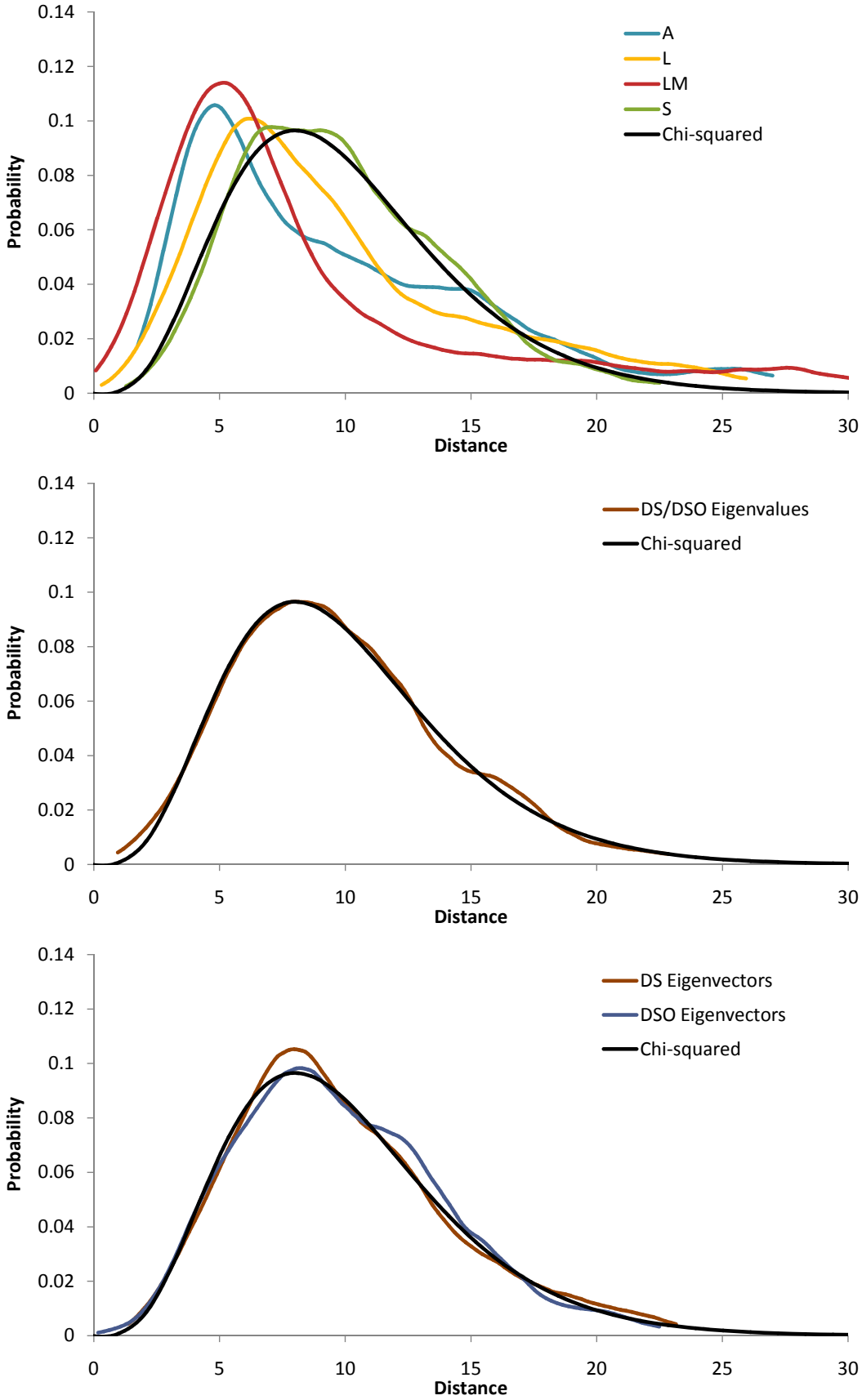


Figure 4.13: Distance distributions on the random data. The results are split across three graphs to aid interpretation. The top graph describes models A, L, LM and S. The second graph shows the distance distribution for the eigenvalues used in the DS and DSO models while the third graph shows the eigenvectors for DS and DSO.

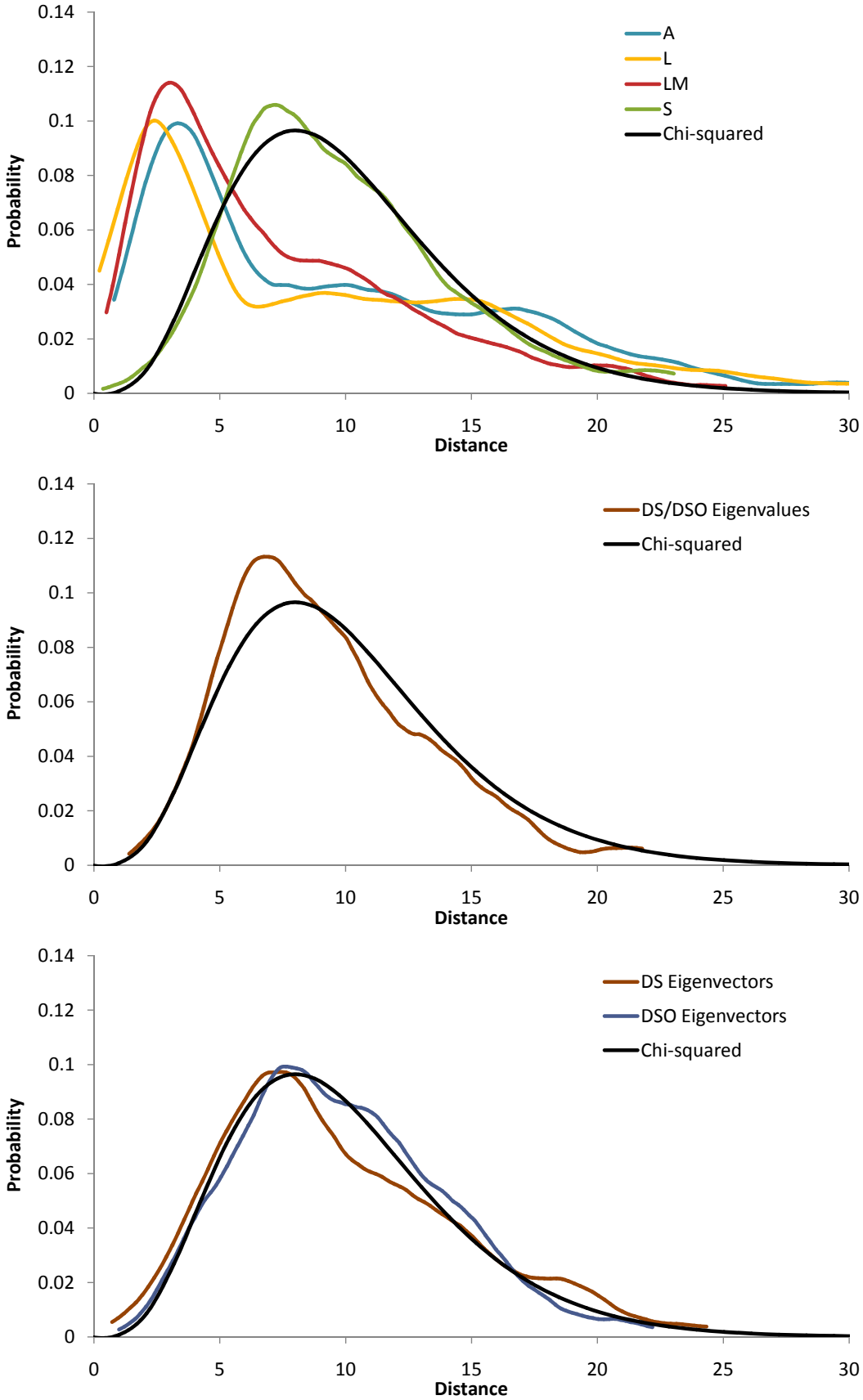


Figure 4.14: Distance distributions on the Delaunay data.

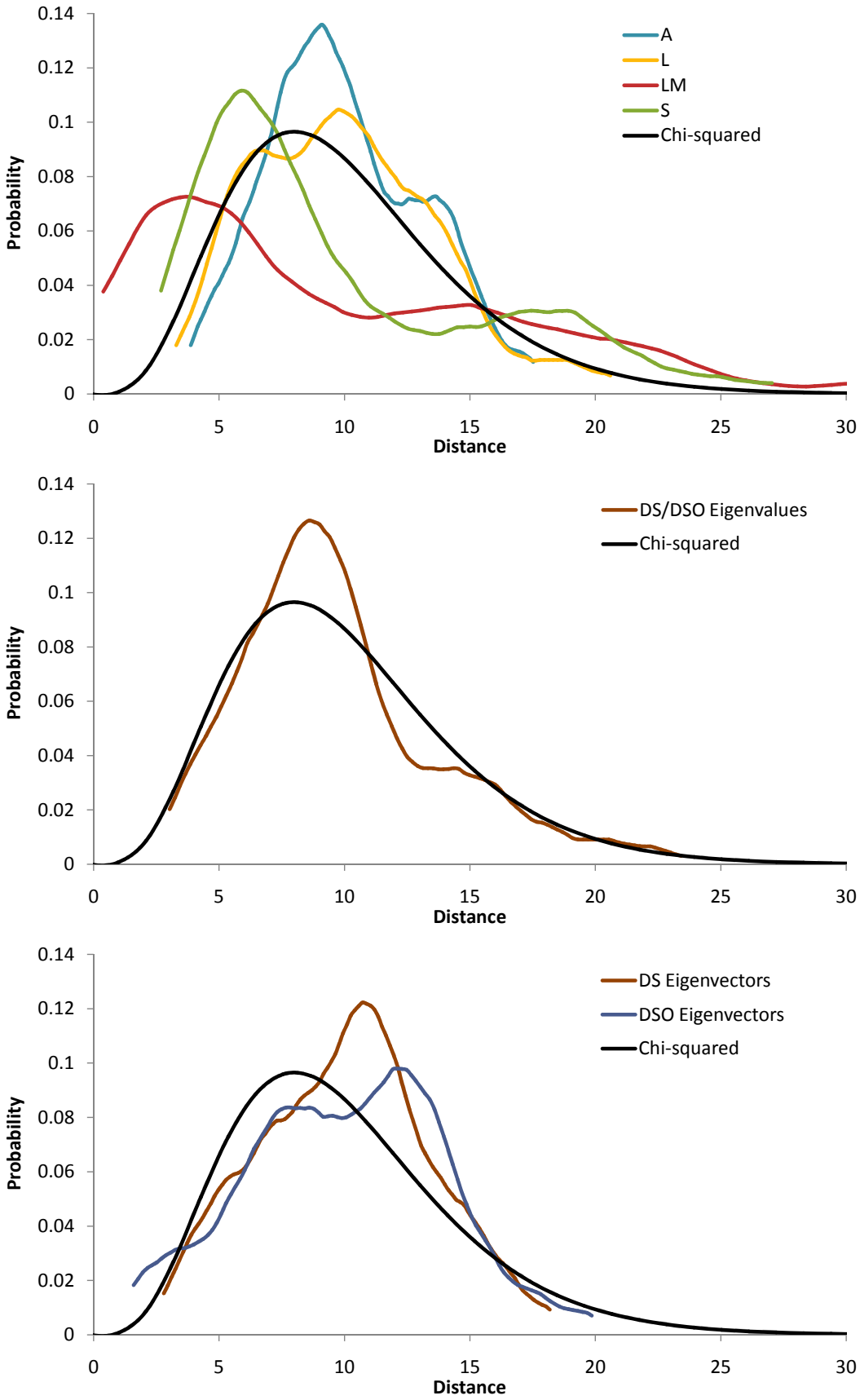


Figure 4.15: Distance distributions on the COIL data using graphs from object 8.

4.4 Conclusions

We have proposed and examined a number of different models on aligned graphs. By vectorizing the canonical matrix representations we are able to construct a vector space suitable for basing a generative model on. In this vector space we are able to estimate statistical quantities such as the mean and covariance, a task that is very complex to perform on non-transformed relational data. Using the mean and covariance we form a model representing the underlying structural variations present in the set of graphs. The models are fully generative in the sense that we can use them to generate new graphs, although in some cases a generated representation may require a recovery step before it represents a valid graph.

The different models we construct range in complexity. The simplest are those formed on the adjacency and Laplacian matrices. While these form adequate models they require a conditioning step to transform a generated vector into a valid graph. Next, we considered models based on the spectral decomposition of a graph. We expected that these will perform better as input to our approach due to the success of spectral methods at explaining graph structure. They are limited however by constraints on the spectral matrices and generated examples do not satisfy these. By using the exponential map we were able to overcome these difficulties.

Of the non-spectral models, both A, L and LM seem to perform quite similarly although using the PD map gives a slight improvement in compactness in certain situations at the expense of normality. For the spectral models, it is clear that S gives an unsatisfactory model of the distribution. The DS and DSO models give more normal distributions but are substantially less compact than the non-spectral models. This is true in both the synthetic data and the real world data. In the classification experiments, the DS and DSO models perform significantly better on the COIL data set than the non-spectral methods. In the Delaunay data set the DS model performs very well, but adding the orthogonal map seems to reduce the performance. Finally, in the random data set, we see the non-spectral methods significantly outperform the spectral methods.

The plots describing the distribution of sample and generated graphs generally show that the generated graphs fall within the distribution of sample graphs. In these tests, the DS and DSO models perform well on data from both the random and COIL data sets. In contrast, the distribution of generated graphs resulting from the S

model is significantly different to that of the sample graphs. This is due to problems described earlier with encoding the eigenvalue magnitude in vector lengths.

The results reinforce the idea that the spectral methods are better at explaining structural data. In general we see the performance of the non spectral methods decrease when moving from random data to more structural data, i.e. random graphs to Delaunay graphs and finally to graphs from the COIL data set, while we observe the opposite for the good spectral methods (DS and DSO). Clearly any useful real world data will be more on the structural end of this spectrum and therefore the DS and DSO methods would be more appropriate for modeling it and generating new examples.

4.4.1 Future Work

We would like to extend this work with a further investigation into the nature of applying the orthogonal map to the dual spectral method. In theory, it is not clear why it should reduce the performance of the dual spectral method and indeed, we expected it to enhance the performance. By basing the distribution in the tangent space to the manifold of orthogonal matrices, we are computing the mean and covariance more accurately than is possible in the original space. This should have translated into an increase in performance when we performed our experimental analysis but in the best case the method only equals the performance of the DS model. One could argue that the distance distribution tests indicate that the normality of the DSO model is superior than the DS model, but it is a very minor improvement. In any case a further investigation is required.

4.5 Symbols

$\mathcal{S}$	A set of graphs.
R	The reference graph from the set $\mathcal{S}$.
S_k	A graph from the set $\mathcal{S}$.
$\mathbf{X}_k$	A matrix representation of graph S_k.
$\mathbf{x}_k$	A vectorial representation of graph S_k.
$\mathring{\mathbf{A}}$	An aligned adjacency matrix representation of a graph.
$\mathring{\mathbf{L}}$	An aligned Laplacian matrix representation of a graph.
$\mathring{\mathbf{N}}$	An aligned normalized Laplacian matrix representation of a graph.
$\mathbf{E}, \mathbf{V}$	The set of eigenvalues $\mathbf{E}$ and eigenvectors $\mathbf{V}$ resulting from the eigendecomposition of an aligned normalized Laplacian matrix representation of a graph $\mathring{\mathbf{N}}$.
$\mathbf{x}'$	A generated vectorial representation of a graph.
$\mathbf{A}'$	An adjacency matrix representation of a generated graph.
$\mathbf{E}', \mathbf{V}'$	A generated set of eigenvalues $\mathbf{E}'$ and eigenvectors $\mathbf{V}'$.
$\mathbf{V}''$	A generated set of orthonormal eigenvectors.
$\mathring{\mathbf{N}}'$	A generated normalized Laplacian matrix representation of a graph.
θ	A thresholding function for a matrix representation. A parameter β determines the threshold level.
$\boldsymbol{\mu}$	A mean vector.
$\boldsymbol{\Sigma}$	A covariance matrix.
$\boldsymbol{\Lambda}, \boldsymbol{\Phi}$	The set of eigenvalues and eigenvectors resulting from an eigendecomposition of the covariance matrix $\boldsymbol{\Sigma}$.
$\mathbf{b}$	A parameter vector of a multivariate normal distribution.
$\mathbf{m}$	The geodesic mean vector.
$\mathbf{M}$	The geodesic mean matrix.

Chapter 5

Parts Based Generative Models for Graphs

5.1 Introduction

The construction of a generative model for graph structure can proceed in a number of different ways. In the previous chapter we described a generative model that considered graphs as whole entities and sought to model the structural variations in a set by using a normal distribution on vectorial representations of graphs to identify the principle components of variation. We adopt a different approach here. Instead of constructing a generative model over the whole graph, we subdivide the sample graphs into a number of logical substructures and model the variation in these substructures and the manner in which these substructures are connected. In short, this involves the construction of three generative models in total that when combined give us the ability to generate complete graph structures.

The idea of decomposing graph structure into a number of logical substructures to aid interpretation, identification and pattern recognition algorithms is not new. The field of graph segmentation (section 2.1.4) has long tried to find suitable approaches to the decomposition of an image or more generally graphs. Applications of image segmentation include analyzing medical images, locating features in satellite imagery, detecting objects in scenes and fingerprint recognition. Perhaps the best known and most widely applicable is Shi & Malik's normalized cut algorithm [100]. This algorithm uses graph spectral methods to find the optimal way to

cut a graph. Furthermore, its cut criterion is normalized by the number of vertices on either side of the cut. Therefore, it does not repeatedly cut very small structures as some other cut criteria such as Wu & Leahy's minimum cut [124] can do.

Our approach of parts-based structure generation resembles those of chemical structure elucidation[82] and *de novo* approaches[97] where a new molecule is built up sequentially from a number of smaller fragments (section 2.2.3). Generally, given some data on the molecule in question, these approaches produce a structural representation of the molecule most consistent with the data. One of the earliest of these programs was CONGEN [23] which was shortly followed by GEONA [24]. Later CHEMICS [45] was developed which explored the fragment combinations using a more systematic method than GEONA. More recently, Porquet et al [87] describe a "self generation algorithm" which seeks to build large 3D molecules by the selective joining of small fragments.

However, our approach differs in a number of ways. Instead of selecting the best fragments (substructures) that fit the data, we select substructures according to the distribution of substructures present in the sample graphs. Furthermore, we do not use an incremental joining strategy, the connections between substructures chosen for a new graph are all produced simultaneously when we sample from our model of substructure connection.

Therefore, our approach lends itself to graph types that are easily decomposed, for example, graphs of chemical structures or point sets of objects. Clearly this approach will only be successful when the graphs in the sample set have a reasonable degree of similarity and therefore subgraphs that are similar will be found in many of the sample graphs. The same applies to the connections between subgraphs. The algorithm excels on graphs with small groups of densely connected vertices where the small groups are sparsely connected.

Furthermore, by decomposing the graphs we greatly reduce the complexity of the generative model. This in turn reduces the errors when we sample from the distributions and allows us to generate graphs that more accurately reflect the original distribution of graphs. Moreover, the computational complexity of the algorithm is reduced and the approach may be used on large graphs.

The remainder of this chapter is laid out as follows; in section 5.2 we describe our method which is split into two parts, the construction of the models and gener-

ating new graphs from the models. In section 5.3 we provide experimental results for our approach on both synthetic data and real-world data. Our real-world data comes from examining different views of articulated objects. Finally in section 5.4 we draw our conclusions and suggest some future directions for this work. A table of the symbols used in this chapter is given in section 5.5 on page 124.

5.2 Method

We describe our algorithm in two sections. First we discuss how the sample graphs are prepared for use with the model, then we describe the models themselves and how we can sample from them to generate a new graph.

We provide an overview of the approach here. The algorithm proceeds from a sample set of graphs. Each graph in the set is decomposed into a set of subgraphs by performing a number of graph cuts. The subgraphs that result from this and the cut connections between subgraphs are recorded. These two sets will form the basis of the model.

To determine which subgraphs represent similar structure, we perform clustering on the subgraphs produced from all sample graphs. We perform an alignment step by choosing a reference graph for each cluster and aligning all subgraphs to their associated reference graph. The matrices storing the cut connections are also permuted to correspond to the new alignments.

Once the clustering is complete and each subgraph is assigned to a cluster we can change our view of the cut connections from *connections between two subgraphs* to *connections between two clusters*. We can also change our high-level view of each sample graph from *consisting of a set of subgraphs* to *consisting of a set of clusters*. This view is essential to accurately construct the model and allows us to think about each sample graphs in terms of a set of clusters and a set of connections between clusters.

Using this information we can construct three models that describe the structural variation present in the sample graphs.

- A distribution over which set of clusters is present in each sample graph.
- The distribution of subgraphs in each cluster.

- The distribution of connections between clusters.

The process of generating a new graph by sampling from the three models above is given in section 5.2.2. First we find a configuration of clusters according to the distribution of clusters present in the sample graphs. Next a subgraph must be chosen to represent each cluster present in the new graph. Finally, we generate the connections between each pair of clusters found in the new graph using the distribution of the cut connections.

5.2.1 Constructing the Generative Model

Conditioning the Graphs

We must first condition the graphs for use with the generative model. This involves segmenting a graph to find the subgraphs and resulting connections between the subgraphs. These are recorded using a matrix representation. To simplify working with the matrix representations we pad them with dummy vertices so they are all equal size.

The algorithm commences from a sample set of non-directed graphs $\mathcal{S}$ which may or may not be weighted. We compute the adjacency matrix representation (section 3.2.1) $\mathbf{S}_k$ for each graph $S_k \in \mathcal{S}$.

We seek a partitioning of each sample graph S_k such that each sub-structure i present in the graph is represented by a subgraph F_{ki} with associated adjacency matrix $\mathbf{F}_{ki}$. Each subgraph adjacency matrix represents the subgraph vertices in the partition and the edges wholly within the partition. The set of subgraphs for a graph S_k are given by the set $\mathcal{F}_k$. Clearly this process fails to capture information about the way sub-structures are connected to each other, so we store connections between each pair of subgraphs F_{ki} and F_{kj} in a *connection matrix* $\mathbf{C}_{kij}$. The connection matrices are created by recording the cut connections between two subgraphs. If a connection is cut between vertex $u \in F_{ki}$ and $v \in F_{kj}$ then $C_{kij}(u, v) = 1$ otherwise $C_{kij}(u, v) = 0$.

This partitioning process is quite flexible and adaptable to the specific data set the process is employed on. For example, we do not require an equal number of partitions for each graph in the sample set allowing graphs with vastly different levels of structure to be accurately represented. The partitioning process itself can

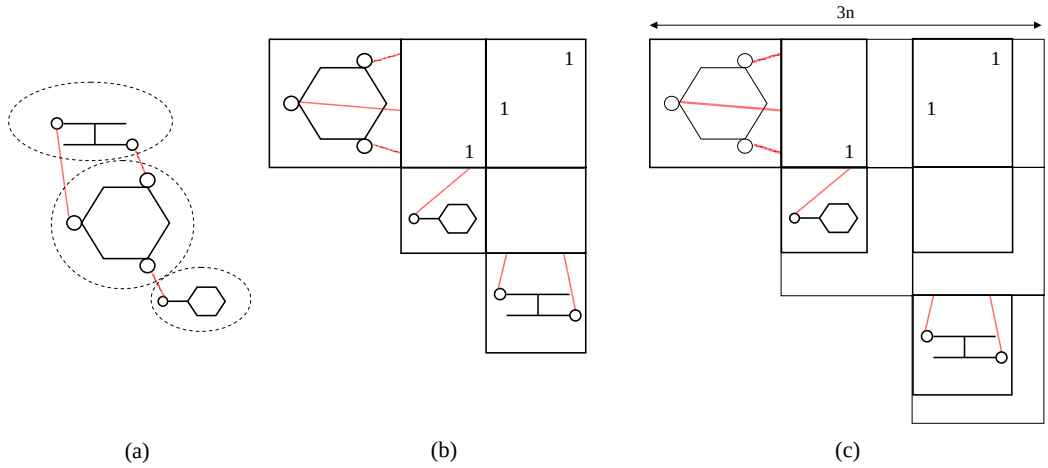


Figure 5.1: An example showing (a) the original graph, (b) the partitioned adjacency matrix of the graph and (c) the padding of the subgraph adjacency matrices and connection matrices.

be done using conventional methods such as Shi & Malik's normalized cut [100] or a more application specific method.

Due to the sample graphs varying in size and the partitioning process producing subgraphs of various sizes, the subgraphs can vary dramatically in size. To simplify working with these different sized subgraphs and connection matrices we find the size n of the largest subgraph and then pad all other subgraph adjacency matrices $\mathbf{F}_{ki}$ and connection matrices $\mathbf{C}_{kij}$ with zeros so they are $n \times n$ in size. The adjacency matrices $\mathbf{S}_k$ are also expanded with dummy vertices to accommodate the padded subgraph adjacency matrices and connection matrices. The result of this are padded sample graph adjacency matrices $\check{\mathbf{S}}_k$, padded subgraph adjacency matrices $\check{\mathbf{F}}_{ki}$ and padded connection matrices $\check{\mathbf{C}}_{kij}$.

Figure 5.1 shows an example of the partitioning and padding process. Figure 5.1(a) shows the graph with partitions as ellipses. Small circles and red edges indicate the vertices with cut connections. Figure 5.1(b) is a representation of the adjacency matrix of a sample graph ($\mathbf{S}_k$) that has been partitioned. The small graph pictures indicate the subgraphs and the red lines indicate where they connect to other subgraphs. The connection matrices use a 1 to indicate that a connection is present between those two vertices and the remainder of the connection matrix (all zeros) is omitted for visibility. Since the graphs are undirected only the connection matrices in the upper triangle of the adjacency matrix are shown. Figure 5.1(c) is the adjacency matrix of the padded sample graph ($\check{\mathbf{S}}_k$) and is $3n \times 3n$ in size. Each

subgraph and connection matrix has been copied into the appropriate place. In this example the subgraph in the top left corner is the largest in the whole set of subgraphs and therefore n is equal to the size of this subgraph. This is why there is no padding around this subgraph in figure 5.1(c).

Clustering the Subgraphs

In this section we describe how we produce clusters of subgraphs such that subgraphs belonging to the same cluster represent similar structure. This information is used to organize the input of subgraphs and connection matrices to the generative model. Furthermore, it allows us to view the connections between subgraphs as connections between clusters instead.

To cluster the subgraphs we compute the distance $d(.,.)$ between every pair of subgraphs using the weighted graph matching approach of Gold & Rangarajan [48]. The set of all subgraphs is given by $\mathcal{F} = \bigcup_{k=1}^{|\mathcal{S}|} \mathcal{F}_k$. The distance between two subgraphs F_{ki} and F_{lj} is given by $d(F_{ki}, F_{lj})$. The correspondence matrix obtained from each match is transformed into a permutation matrix using the Hungarian method to obtain the best alignment between the two subgraphs. The permutation matrix for matching F_{lj} to F_{ki} is given by $\mathbf{M}(F_{ki}, F_{lj})$. The distances given by $d(.,.)$ between the subgraphs are used to form an affinity matrix and Normalized Cut clustering is performed on this matrix. Using the result of the clustering we define a function ψ that maps a subgraph F_{ki} to the cluster y that it belongs to.

$$\psi : F_{ki} \rightarrow y \tag{5.1}$$

The number of cuts we allow here corresponds to the number of different structures in the sample graphs that are needed for the generative model to accurately convey the sample graph's structures. This is best chosen using some domain knowledge about the problem in question however, a limit such as the cut threshold used in Normalized Cut clustering could be used.

With the clusters of subgraphs to hand we compute a reference subgraph for each cluster. The other subgraphs in the cluster will be matched to this. The reference subgraph R_c for a cluster c is chosen to be the one with the smallest distance to all other subgraphs in that cluster:

$$R_c = \underset{A \in \mathcal{X}_c}{\operatorname{argmin}} \sum_{B \in \mathcal{X}_c} d(A, B) \tag{5.2}$$

where $\mathcal{X}_c = \{\forall F_{ki} \in \mathcal{F} | \psi(F_{ki}) = c\}$ is the set of subgraphs belonging to cluster c.

We can then use R_c to simplify the equations involving match matrices $\mathbf{M}$. We define the new match matrices to be $\mathbf{M}'(F_{ki})$ which holds the alignment between subgraph F_{ki} and its associated cluster reference subgraph R_c where $c = \psi(F_{ki})$. In other words:

$$\mathbf{M}'(F_{ki}) = \mathbf{M}(R_c, F_{ki}) \tag{5.3}$$

Each subgraph adjacency matrix is aligned to its cluster's reference subgraph using $\mathbf{M}'$:

$$\mathring{\mathbf{F}}_{ki} = \mathbf{M}'(F_{ki}) \check{\mathbf{F}}_{ki} \mathbf{M}'(F_{ki})^T \tag{5.4}$$

where the aligned padded subgraph is now denoted $\mathring{\mathbf{F}}_{ki}$.

Since the vertices of each subgraph have been rearranged, the connection matrices must also be permuted. The permutation of the connection matrix between subgraphs F_{ki} and F_{kj} is given below:

$$\mathring{\mathbf{C}}_{kij} = \mathbf{M}'(F_{ki}) \check{\mathbf{C}}_{kij} \mathbf{M}'(F_{kj}) \tag{5.5}$$

The full adjacency matrices representing each sample graph are updated with the alignments and padding and are denoted $\mathring{\mathbf{S}}_k$.

A problem can arise due to the fact that a number of equally valid alignments can be returned by the matching algorithm. Although these alignments are all valid in the context of matching subgraphs, they are not all valid when the connection matrices are permuted.

To illustrate this problem we provide the following example. Suppose from sample graph S_1 we have two subgraphs $F_{11}, F_{12} \in \mathcal{F}_1$ and a connection matrix $\check{\mathbf{C}}_{112}$. From S_2 we also have two subgraphs $F_{21}, F_{22} \in \mathcal{F}_2$ and a connection matrix $\check{\mathbf{C}}_{212}$. The subgraph pair F_{11} and F_{21} are from the same cluster and the pair F_{12} and F_{22} are from a different cluster. Figure 5.2 shows this arrangement diagrammati-

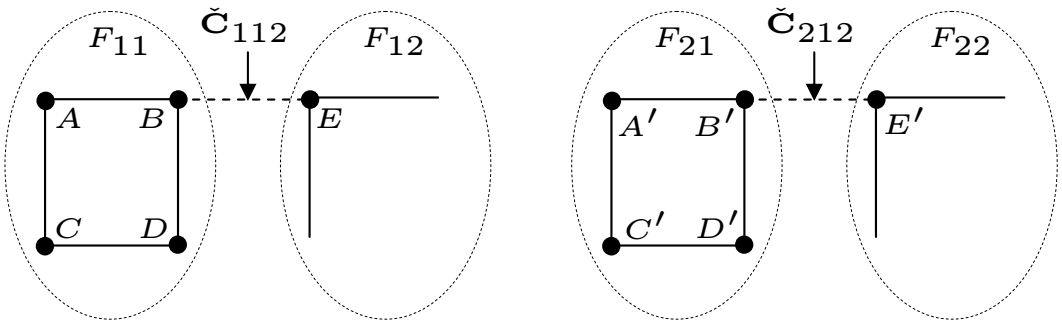


Figure 5.2: An alignment problem: if care is not taken when aligning isomorphic subgraphs then an incorrect alignment can be produced when the connection matrices are considered.

cally. The vertices in F_{11} are labeled A, B, C, D and the vertex in F_{12} is labeled E. The subgraphs from S_2 are labeled identically but with primes.

Say F_{21} is the cluster reference and we align F_{11} to it. Since both subgraphs are isomorphic and ring shaped there are many possible valid matches. Suppose that the matching algorithm returns the following mapping: $A \rightarrow D'$, $B \rightarrow C'$, $C \rightarrow B'$ and $D \rightarrow A'$. While this matching is valid in the context of matching the two subgraphs, if we apply this mapping to the connection matrices we find this match is invalid. Consider the connection matrices before the match (only the first column is shown):

$$\check{\mathbf{C}}_{112} = \begin{array}{c} E \\ \left(\begin{array}{ll} 0 & \ldots \\ 1 & \ldots \\ 0 & \ldots \\ 0 & \ldots \end{array}\right) \end{array} \begin{array}{l} A \\ B \\ C \\ D \end{array} \qquad \check{\mathbf{C}}_{212} = \begin{array}{c} E' \\ \left(\begin{array}{ll} 0 & \ldots \\ 1 & \ldots \\ 0 & \ldots \\ 0 & \ldots \end{array}\right) \end{array} \begin{array}{l} A' \\ B' \\ C' \\ D' \end{array}$$

Now we permute $\mathbf{C}_{112}$ using the alignment given above:

$$\mathring{\mathbf{C}}_{112} = \begin{array}{c} E \\ \left(\begin{array}{ll} 0 & \ldots \\ 0 & \ldots \\ 1 & \ldots \\ 0 & \ldots \end{array}\right) \end{array} \begin{array}{l} A \\ B \\ C \\ D \end{array}$$

Clearly, this alignment is invalid if we want to form a model over the connection matrices. Only matches that align $B \rightarrow B'$ are valid for the connection matrices as well as subgraphs.

Cluster 1 SG	Cluster 1-2 CM	Cluster 1-3 CM	Cluster 1-4 CM	Cluster 1-4 CM
	Cluster 2 SG	Cluster 2-3 CM	Cluster 2-4 CM	Cluster 2-4 CM
		Cluster 3 SG	Cluster 3-4 CM	Cluster 3-4 CM
			Cluster 4 SG	Cluster 4-4 CM
				Cluster 4 SG

(a)

$\mathring{\mathbf{F}}_{12}$	$\mathring{\mathbf{C}}_{123}$	$\mathbf{0}$	$\mathring{\mathbf{C}}_{121}$	$\mathbf{0}$
	$\mathring{\mathbf{F}}_{13}$	$\mathbf{0}$	$\mathring{\mathbf{C}}_{131}$	$\mathbf{0}$
		$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
			$\mathring{\mathbf{F}}_{11}$	$\mathbf{0}$
				$\mathbf{0}$

(b)

Figure 5.3: An example of subgraph and connection matrix placement inside a padded sample graph adjacency matrix.

To force the matching algorithm to always return an alignment that considers the connection matrices we adjust the values on some edges as follows. For each subgraph, we add a small weight to the edges of a vertex that has cut connections recorded in the subgraph's connection matrices. This forces the best match between two subgraphs to take into consideration the way they connect to other subgraphs.

Using the information we have produced about the number of subgraph clusters we can pad the sample graph adjacency matrices one final time. Each sample graph is expanded up to the maximum number of clusters plus additional room if a sample graph contains two or more subgraphs from the same cluster. Let us denote this number m. For example, if there are 10 clusters identified and one sample graph contains two subgraphs from the same cluster then this step will expand all the sample graph adjacency matrices to accommodate $m = 11$ subgraphs. The relevant subgraph for each cluster is placed in one of these positions (if that sample graph contained a subgraph from that cluster). The connection matrices corresponding to the subgraphs are then placed. This representation results in a large amount of padding however using a sparse matrix representation this is easily accommodated. These fully padded sample graph adjacency matrices are denoted $\mathring{\mathbf{S}}'_k$ and are $nm \times nm$ in size. We now present some examples to illustrate this padding procedure.

Figure 5.3(a) shows where the subgraphs and connection matrices are placed to

$\mathring{\mathbf{F}}_{12}$	$\mathbf{0}$	$\mathring{\mathbf{C}}_{123}$	$\mathring{\mathbf{C}}_{121}$	$\mathring{\mathbf{C}}_{124}$
	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
		$\mathring{\mathbf{F}}_{13}$	$\mathring{\mathbf{C}}_{131}$	$\mathring{\mathbf{C}}_{134}$
			$\mathring{\mathbf{F}}_{11}$	$\mathring{\mathbf{C}}_{114}$
				$\mathring{\mathbf{F}}_{14}$

Figure 5.4: An example of subgraph and connection matrix placement highlighting the treatment of a sample graph containing multiple subgraphs from the same cluster.

make $\mathring{\mathbf{S}}'_k$ in the case of 4 clusters with subgraphs from cluster 4 appearing twice in one or more sample graphs. Since the graphs are undirected, only the connection matrices in the upper triangle are shown for clarity.

Figure 5.3(b) shows how a sample graph S_1 ($\mathring{\mathbf{S}}'_1$) is rebuilt into the previous examples cluster arrangement. It contains 3 subgraph adjacency matrices, $\mathring{\mathbf{F}}_{11}$ from cluster 4, $\mathring{\mathbf{F}}_{12}$ from cluster 1 and $\mathring{\mathbf{F}}_{13}$ from cluster 2. Three connection matrices describe how the subgraphs are connected. The spaces in the matrix not occupied by subgraphs or connection matrices are filled with $\mathbf{0}$, the $n \times n$ matrix of zeros.

Figure 5.4 shows a sample graph that contains 4 subgraph adjacency matrices, $\mathring{\mathbf{F}}_{11}$ and $\mathring{\mathbf{F}}_{14}$ from cluster 4, $\mathring{\mathbf{F}}_{12}$ from cluster 1 and $\mathring{\mathbf{F}}_{13}$ from cluster 3. This example highlights how a sample graph containing multiple subgraphs from the same cluster is handled.

Constructing the Models

In this section we describe how the models are created for both the clusters contained in a sample graph and the connections between subgraphs. The distribution of subgraphs in a cluster is easily defined and described in the next section. In figure 5.5 an overview of the models required for the parts-based approach is given. We describe the models (a) and (c) in this section and defer discussion of model (b) until the next section due to its simplicity.

Due to the clustering performed on the subgraphs we can now speak about the connection matrices in terms of *connections between clusters* instead of *connections between subgraphs*. We can also think of each sample graph being composed of a *set of clusters* rather than a *set of subgraphs*.

To create the model of how subgraphs are connected (model (c) in figure 5.5), we construct a long vector $\mathbf{s}_k$ describing all the connection matrices present for a sample graph S_k. The placement of each connection matrix in the long vector $\mathbf{s}_k$ is of great importance if we want each component of $\mathbf{s}_k$ to represent the same connection over the whole set of sample graphs. To accomplish this we divide the long vector into sections. Each section represents the connections between two subgraphs from a specific pair of clusters. We define a function $\xi(a, b)$ that returns the section of the long vector $i : j$ (from index i to index j) for a given pair of clusters a and b.

$$\xi : (a, b) \rightarrow i : j \tag{5.6}$$

where $a \leq b$ (since the graphs are undirected we only need to record the connections between two clusters in one direction).

We can then build $\mathbf{s}_k$ for each S_k using the connection matrix between each pair of clusters (a, b) where $a \leq b$:

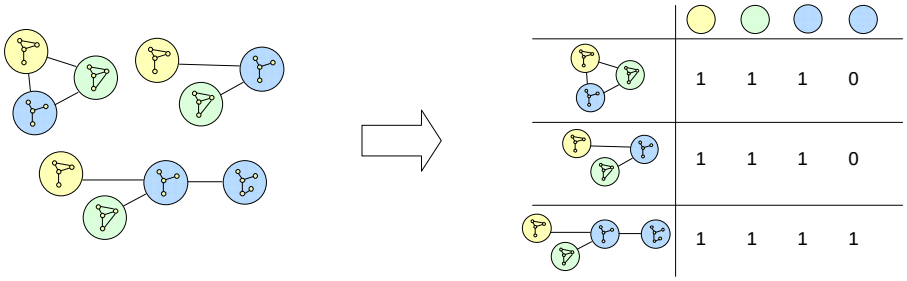
$$\mathbf{s_k}(\xi(a, b)) = \begin{cases} \text{vec}(\mathring{\mathbf{C}}_{kij}) & \text{if } \exists F_{ki}, F_{kj}.i \neq j \wedge \psi(F_{ki}) = a \wedge \psi(F_{kj}) = b \\ \text{vec}(\mathbf{0}) & \text{otherwise} \end{cases} \tag{5.7}$$

In other words, for every section $\xi(a, b)$ of the long vector $\mathbf{s}_k$ we either fill it with zeros (if there is no pair of subgraphs from cluster a and b) or a connection matrix (if a pair of subgraphs from cluster a and b exists).

Therefore, each $\mathbf{s}_k$ stores connections between all possible cluster pairs even though only a few cluster to cluster connections will probably be needed. This ensures that in each long vector all cluster to cluster connections are taken into account when constructing this part of the generative model.

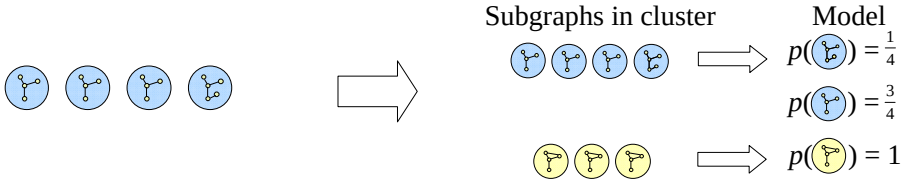
If a sample graph contains a cluster a twice or more, then we will have multiple connection matrices between the same cluster pair. For example, in figure 5.4

(a) Constructing the model of which set of clusters is present in each sample graph



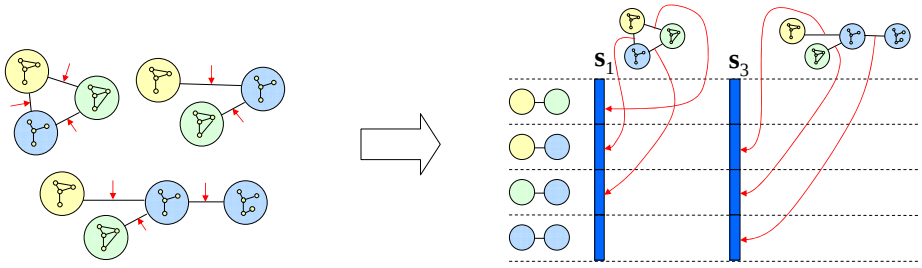
Modelled using a Gibbs Sampler. A binary state (a *cluster configuration*) is computed for each sample graph determining which clusters are present. The Gibbs Sampler produces a new cluster configuration by moving through a series of cluster configurations probabilistically.

(b) Constructing the model of subgraphs in each cluster



Modelled using a frequency selection method.

(c) Constructing the model of the connection matrices



Modelled using a normal distribution. The connection matrices for each sample graph are stacked into a long vector. This vector has room for all possible cluster to cluster connections even though most will be unused on a single sample graph. A normal distribution is then constructed over this vector space.

Figure 5.5: An overview of the models required for the parts-based approach.

subgraphs F_{11} and F_{14} were both from cluster 4. Therefore, connection matrices $\mathring{\mathbf{C}}_{131}$ and $\mathring{\mathbf{C}}_{134}$ both represented connections between cluster 3 and cluster 4. We choose to deal with this by merging multiple connection matrices between the same cluster pair into a single connection matrix. This single merged connection matrix may then be placed in the long vector as described earlier.

The merging process is relatively simple. Let us denote the number of connection matrices (in a single sample graph) between two clusters a and b as z_{ab}. We then merge them by dividing each connection matrix by z_{ab} and summing the resulting matrices. In terms of the example before, $a = 3, b = 4$ so $z_{ab} = z_{34} = 2$ meaning the connection matrix between cluster 3 and cluster 4 would be $\frac{1}{2}\mathring{\mathbf{C}}_{131} + \frac{1}{2}\mathring{\mathbf{C}}_{134}$.

However, this is not an ideal solution. We must make the assumption that multiple occurrences of subgraphs from the same cluster are connected in the same way. If this is not the case then merging is clearly not a good idea. Secondly, if we generate a graph that requires x multiple connection matrices between the same cluster then we must generate x long vectors. Clearly any additional long vectors generated will not take the co-occurrence of connections in the other long vectors into account. We discuss possible solutions to this problem in the section on future work.

With the long vectors $\{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{|\mathcal{S}|}\}$ to hand we proceed to construct a generative model over this vector set. We compute the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of the long vectors. With the covariance matrix to hand we perform an eigendecomposition resulting in two matrices, $\boldsymbol{\Phi}$ which contains the eigenvectors and $\boldsymbol{\Lambda}$ which contains the eigenvalues. We now have the information needed about the distribution of the connections between clusters in the sample graphs.

To model the distribution of which set of clusters are contained in each sample graph (model (a) in figure 5.5) we use a Gibbs Sampler. A Gibbs Sampler allows us to approximate sampling from a complex join probability distribution $p(x_1, x_2, ..., x_j)$ by sampling from a long sequence of conditionals. If we let $[x_1^{(0)}, ..., x_j^{(0)}]$ be a set of arbitrary initial values, we first sample $x_1^{(1)}$ from the conditional distribution $p(x_1|x_2^{(0)}, ..., x_j^{(0)})$, then sample $x_2^{(1)}$ from the conditional distribution $p(x_2|x_1^{(1)}, x_3^{(0)}, ..., x_j^{(0)})$ and so on until we sample $x_j^{(1)}$ from $p(x_j|x_1^{(1)}, , ..., x_{j-1}^{(1)})$. After repeating this for r iterations, we sample $[x_1^{(r)}, ..., x_j^{(r)}]$. If r is large enough, then $[x_1^{(r)}, ..., x_j^{(r)}]$ can be viewed as a sample from the original joint probability

distribution $p(x_1, x_2, ..., x_j)$.

To prepare the Gibbs Sampler for use we must setup a number of binary states that indicate which sample graph contains subgraphs from which cluster. Let us term this a *cluster configuration*. This information is stored in a matrix $\mathbf{U}$ that is $|\mathcal{S}| \times m$ in size. Each row $\mathbf{u}(k)$ indexes a sample graph k and its elements describe which clusters are contained in that sample graph.

$$U(k,c) = \begin{cases} 1 & \text{if } \exists F_{ki} \in \mathcal{F}_k | \psi(F_{ki}) = c \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

For example, if a sample graph S_1 contains a subgraph from cluster 3 then $U(1,3) = 1$, if this is not the case then $U(1,3) = 0$. Again, a complication arises in the case of multiple subgraphs from the same cluster appearing in a single sample graph. Since we require a binary vector, we cannot indicate the multiplicity of such subgraphs directly in the cluster configuration. For example, if S_1 has two subgraphs from cluster 4 then we cannot set $U(1,4) = 2$. Therefore, we use a single bit to represent the appearance of every possible cluster. In other words, if two subgraphs from the same cluster appear in one sample graph then we must allow two bits to represent this. To illustrate this we give the following table of cluster configurations (all subgraphs from other clusters are always present, only cluster 4 varies).

Cluster:	1	2	3	4	4
Zero subgraphs from cluster 4:	1	1	1	0	0
One subgraphs from cluster 4:	1	1	1	1	0
Two subgraphs from cluster 4:	1	1	1	1	1

5.2.2 Sampling and Reconstruction

To generate a new graph we must sample from three distributions. Figure 5.6 provides an overview of this sampling process. First we must find a cluster configuration using the Gibbs Sampler (model (a) in figure 5.5). For each cluster present in the new graph we choose a subgraph from the distribution of subgraphs present in each cluster (model (b)). Lastly, we generate the connections between all pairs of clusters in the cluster configuration (model (c)).

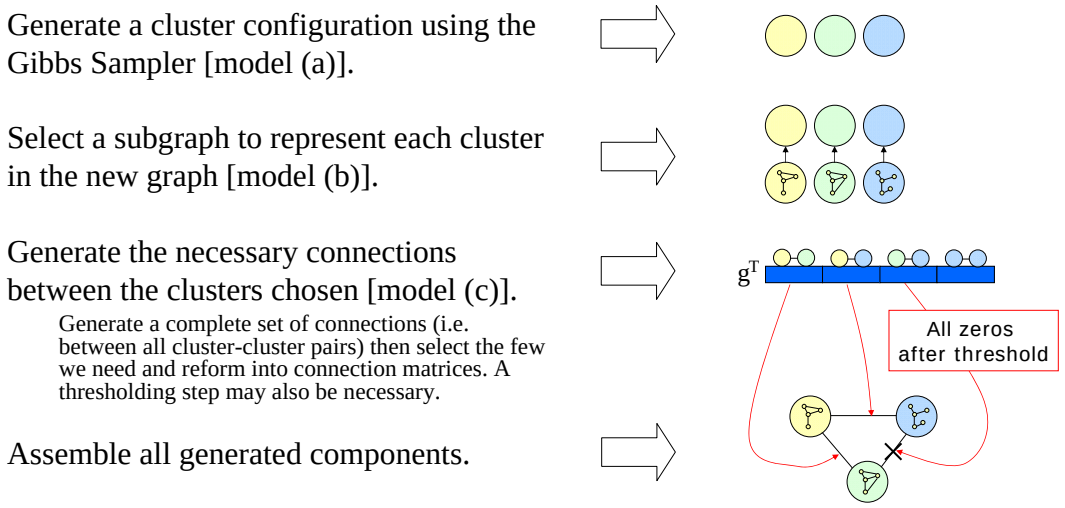


Figure 5.6: An overview of the process of sampling from the models to generate a new graph. The models given in brackets relate to those described in figure 5.5.

The Gibbs Sampler functions as follows. First we initialize the current state s to a randomly selected row of $\mathbf{U}$. On each iteration we create two new states (s_a and s_b) by flipping or not flipping a bit at index v. We then find the probability of each of these states conditionally on the other values of the components of the current state vector. This is done using a sum of exponentiated Hamming distances and then applying a normalization. A single iteration is defined as repeating this until a new value has been sampled for each component exactly once. The component for which a new value should be sampled is chosen sequentially:

$$v \leftarrow \text{mod}(v, m) + 1 \tag{5.9}$$

The sum of exponentiated Hamming distances for states s_a and s_b are computed as such:

$$H_{s_a} = \sum_{k=1}^{|\mathcal{S}|} e^{-\alpha h(s_a, \mathbf{u}(k))} \tag{5.10}$$

$$H_{s_b} = \sum_{k=1}^{|\mathcal{S}|} e^{-\alpha h(s_b, \mathbf{u}(k))} \tag{5.11}$$

where $h(a, b)$ is the Hamming distance between two binary states a and b and α is a tunable parameter set by hand. The two Hamming sums are normalized to find the probabilities of moving to s_a or s_b:

$$P_{s_a} = \frac{H_{s_a}}{H_{s_a} + H_{s_b}} \tag{5.12}$$

$$P_{s_b} = \frac{H_{s_b}}{H_{s_a} + H_{s_b}} \tag{5.13}$$

The state s_a is selected ($s \leftarrow s_a$) with probability P_{s_a} and the state s_b is selected ($s \leftarrow s_b$) with probability P_{s_b}.

After 100 iterations across all components of the vector, we take the current state which provides the configuration of clusters that will be used in our new graph.

With the cluster configuration of the new graph G_k decided we make a new adjacency matrix $\mathbf{G}_k$ of size $nm \times nm$ that will represent the newly generated graph. Next we must choose which subgraph to use for each cluster and generate the connections between them. Choosing the subgraph to be used for each cluster that appears in the new graph is a relatively simple process. We use a frequency selection method. For example if there is one cluster that contains 3 identical subgraphs then that subgraph is 3 times more likely to be selected for that cluster in the new graph than a subgraph that only appears once. As before, we let $\mathcal{X}_c$ be the set of subgraphs from cluster c.

$$\mathcal{X}_c = \{\forall F_{ki} \in \mathcal{F} | \psi(F_{ki}) = c\} \tag{5.14}$$

Therefore if we uniformly select a subgraph from $\mathcal{X}_c$, it will respect the probability with which that subgraph appears in the cluster.

To generate the connection matrices we generate a new set of connections between all cluster pairs and then select the connections that are required for our new graph. The generated set of connections between all cluster pairs is denoted $\mathbf{g}_k$ and computed using the distribution of vectors $\{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{|\mathcal{S}|}\}$:

$$\mathbf{g}_k = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{b} \tag{5.15}$$

where $\mathbf{b}$ is a parameter vector created by sampling from the normal distribution with zero mean and variance given by the diagonal elements in $\boldsymbol{\Lambda}$. It's components are sampled as follows:

$$b(i) \sim \mathcal{N}(0, \Lambda(i,i)) \tag{5.16}$$

We select slices of $\mathbf{g}_k$ to be reshaped back into matrices and used as connection matrices for the new graph. If a connection matrix $\mathbf{C}_g$ between cluster a and b is required then it is retrieved from $\mathbf{g}_k$ as follows:

$$\mathbf{C}_g = \text{devec}(\mathbf{g}_k(\xi(a, b))) \tag{5.17}$$

If the Gibbs sampler produces a cluster configuration containing multiple instances of a cluster then we will require multiple samples of the connections between those cluster pairs. To do this we draw as many long vectors from the distribution as required and take only the connections between the required cluster pair. As discussed earlier, this procedure results in the co-occurrence of connections in additional connection matrices being ignored as each long vector is sampled independently.

Our new graph is almost complete now with the cluster configuration decided, subgraphs to represent those clusters chosen and the connections between those subgraphs generated. All that remains is to finalize the connection matrices of the new graph. Due to the continuous nature of the normal distribution some noise in the connections might have been introduced in generating $\mathbf{g}_k$. If we require discrete edges then we must remove the noise by thresholding the values in every connection matrix such that an edge is only recorded if it is above a certain threshold. Let us denote a thresholded connection matrix $\tilde{\mathbf{C}}_g$:

$$\tilde{C}_g(u, v) = \begin{cases} 1 & \text{if } C_g(u, v) > \beta \\ 0 & \text{otherwise} \end{cases} \tag{5.18}$$

The value of β can be determined by the average value of the elements in the long vectors $\{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{|\mathcal{S}|}\}$. In the case of graphs with weighted edges it can still be a good idea to remove edges with very low weight that represent noise and do not contribute to graph structure. The thresholded connection matrix in this case would be:

$$\tilde{C}_g(u, v) = \begin{cases} C_g(u, v) & \text{if } C_g(u, v) > \beta \\ 0 & \text{otherwise} \end{cases} \tag{5.19}$$

5.3 Experimental Results

We show the utility of our approach by applying it to graphs constructed from point sets. Two data sets are used in this section: the first set is constructed from synthetically generated point sets while the second set uses points extracted from real-world images. We use point sets since it provides a way to visualize both sample and generated graphs.

To perform this visualization we use edge weights to compute the matrix $\mathbf{D}$ which contains the distances between all pairs of vertices. The drawing algorithm can then proceed in one of two ways. One approach is to use multi-dimensional scaling to embed the vertices in a space such that the distance between pairs of vertices is maintained. Although this does not always give an ideal drawing of a graph, it does ensure that the embeddings of sample graphs and generated graphs are similar and therefore provides a way to assess the intricacies of a generated graph.

If the graphs are more rigid then we use an optimization approach to draw the graphs. The process starts from a set of points that represent the vertices of the graph. We then move each point in turn in the direction that minimizes the squared error in the distance between the values in $\mathbf{D}$ and the current distances between points in the point set. We only move the point part of the way towards its minimum position to prevent early convergence to a sub-optimal solution. After many iterations the distances between the points in the point set will reflet those in $\mathbf{D}$ as accurately as possible and the drawing will be complete.

5.3.1 Synthetic Data

Our synthetic data is constructed from 5 different small point sets $x_1, ..., x_5$ each containing 15 points. The point sets $x_1, ..., x_5$ are arranged in a 2D space to make the full point set X_k. Each point set x_i will be represented by a cluster in the algorithm and the way they are arranged will be represented by the connection matrices. To provide some work for the matching step in the algorithm we apply a small amount of noise to the points in each X_k. We compute the Gabriel graph S_k for each point set X_k from which the algorithm commences. The weights on the edges computed by the Gabriel step are set to the distance between the two vertices they connect.

The partitioning of each graph is accomplished using Normalized Cut. The middle row in figure 5.7 shows the Gabriel graphs computed from point sets $X_1, ...X_4$. The position of each x_i in the point set X_k in shown above each graph. The bottom row shows the MDS embedding for each graph S_k. Note that the embeddings may be missing vertices or edges due to the MDS projection when compared to the Gabriel graphs above. However, the connections between the subgraphs are reasonably clear.

Figure 5.8 shows graphs generated using this data set which are drawn using the MDS embedding approach. The cluster configuration produced by the Gibbs sampler is given above every generated graph. Graphs G_1, G_2 and G_3 all use the cluster configuration of S_3 and S_4; graphs G_4 and G_5 use the configuration of S_2 and G_6 uses the configuration of S_1. While it is difficult to see due to the MDS drawing method, the majority of subgraphs used to represent each cluster display different slightly different structure due to the noise added to the sample point sets. The connections between clusters have the largest impact on the structure of the graph, in all graphs except G_5 the connections remain between a single cluster pair, while in G_5 all five clusters are connected by the bottom subgraph.

Graph G_1 provides a good example of how the connections in the sample graphs can be statistically mixed. Notice that the connections on the left of the subgraph indicated by x_5 are identical to those in S_3 and the connections on the right are identical to those in S_4. Also observe that the connections between x_2 and x_4 are different to those in any S_k and are in fact a combination of the connections between x_2 and x_4 in S_3 and S_4. Similar connection combinations can be seen in the other generated graphs but are sometimes hidden due to the drawing method.

We mentioned earlier the role of the thresholding step and how it is used to remove noise that has been introduced into the generated connections. In figure 5.9 we describe the role of the threshold parameter on a generated graph by showing the MDS drawing of the graph for three different threshold values. In figure 5.9(a) the threshold is set too low and too many edges that are noise are allowed to make it into the final graph. In figure 5.9(b) the parameter is set correctly by calculating the average value of the elements in the long vectors. Finally, in figure 5.9 the value is set too high and a crucial connection determining graph structure has been removed.

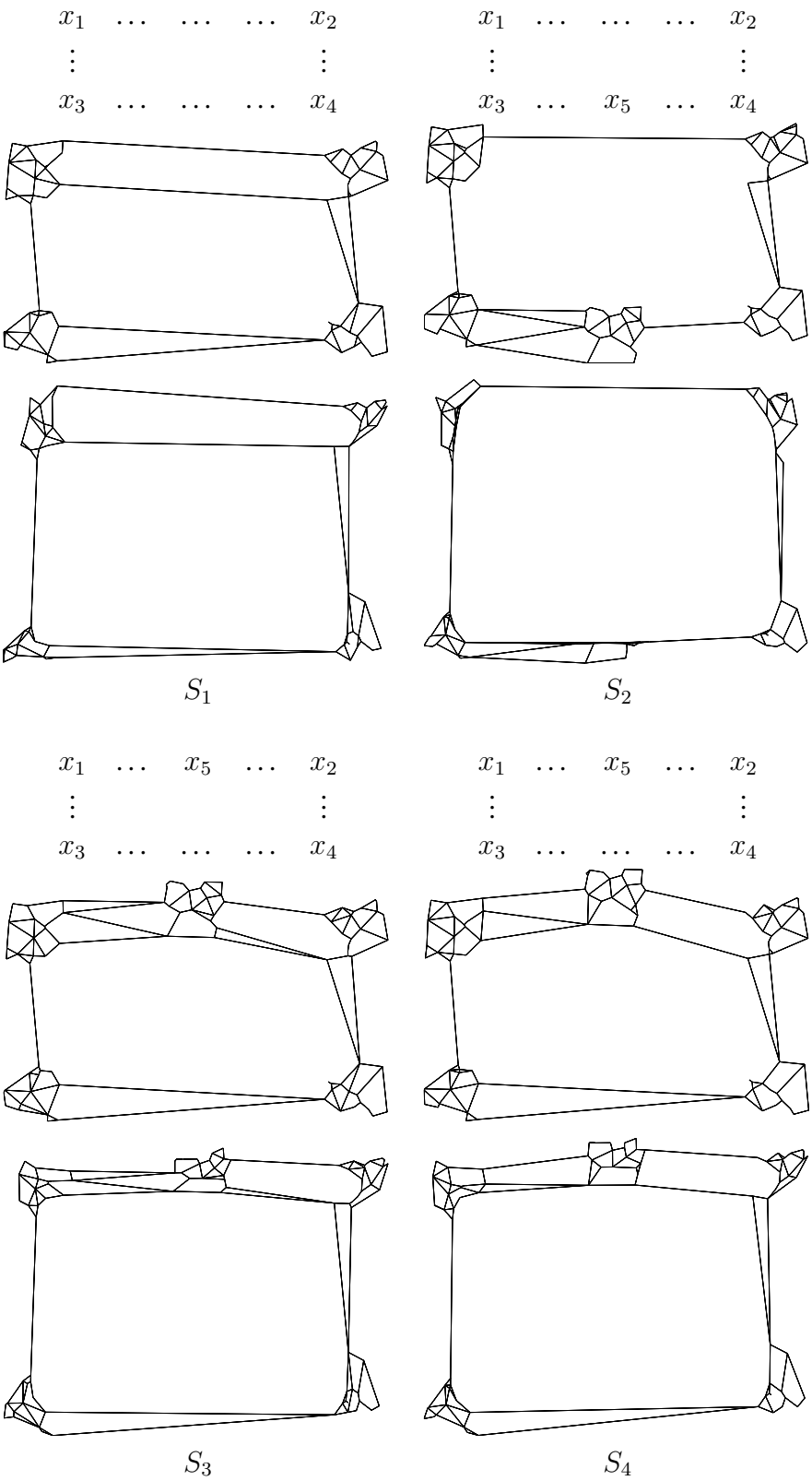


Figure 5.7: Synthetic graphs constructed for the sample set. For each graph the cluster configuration is given in the top row, the middle row gives the Gabriel graph computed from each point set and the bottom row shows the MDS embedding of the graph above it.

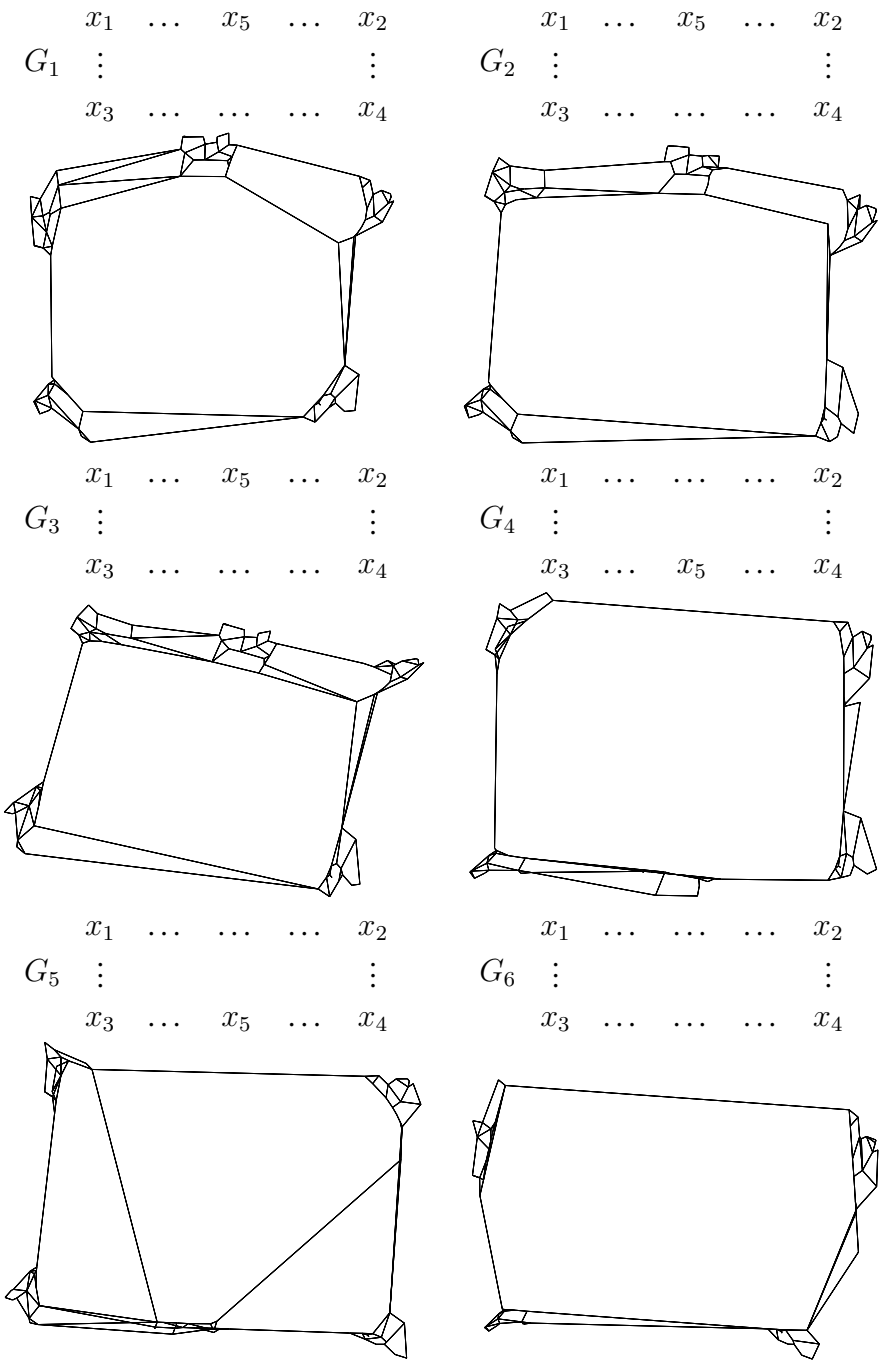
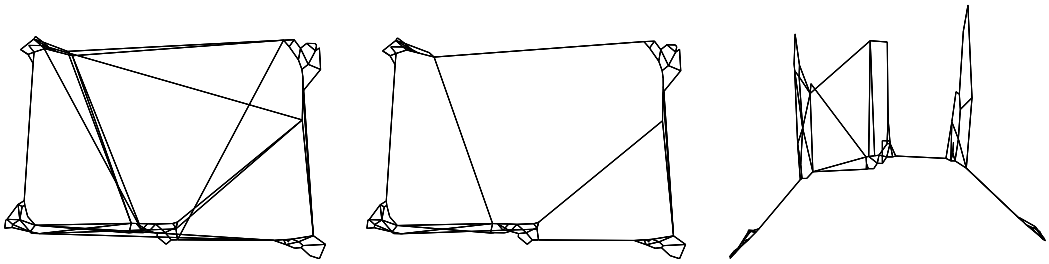
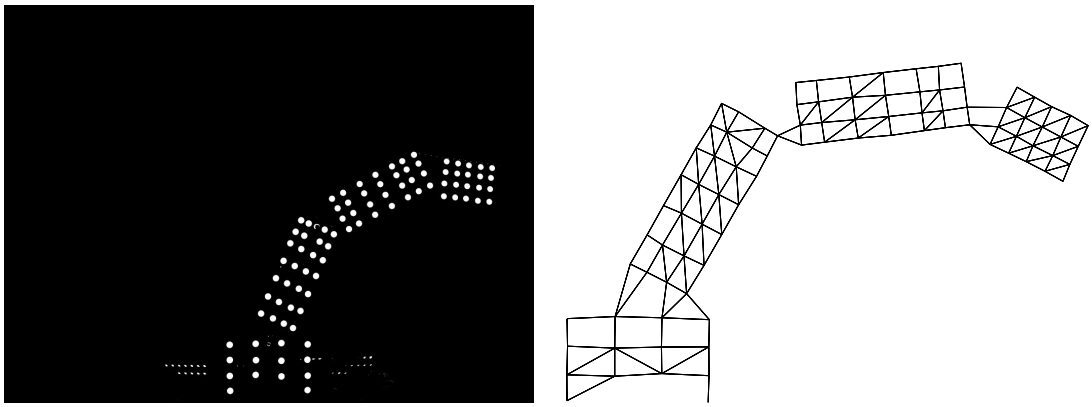


Figure 5.8: Graphs generated using the synthetic data set drawn using MDS embedding.



(a) (b) (c)

Figure 5.9: Various thresholds of the connections between subgraphs for a generated graph.



(a) (b)

Figure 5.10: The motion capture process; the object is augmented with white circles and a simple thresholding step is applied to the image (a). The center of each white dot is found and converted to a coordinate. The Gabriel graph of the coordinate set is computed (b).

5.3.2 Real-world Data

To show the utility of our approach on real world data we use images of articulated objects. These objects have multiple joints about which their various parts can move and are therefore an ideal choice for our approach. Our approach will partition the full object into its various parts and then model the way the parts are connected.

We acquire point sets from the real world images through a motion capture preprocessing step. Although we could use a corner/edge detection algorithm to find the features that identify the parts, we choose to use the more robust process of motion capture. Each moving part is augmented with a number of small white circles that will be found in the preprocessing stage and converted into points. Figure 5.10 shows an example of the motion capture process.

Since the points are fixed on the object in question it is very simple and quick

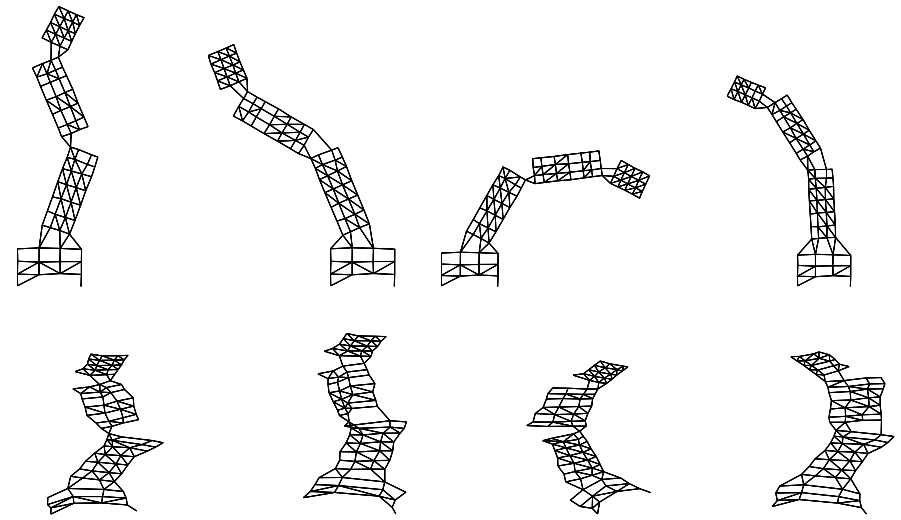


Figure 5.11: Four of the eighteen graphs used for the real-world sample set: the top row shows the Gabriel graphs computed from each sample image and the bottom row shows the optimization based drawing of the graph above it.

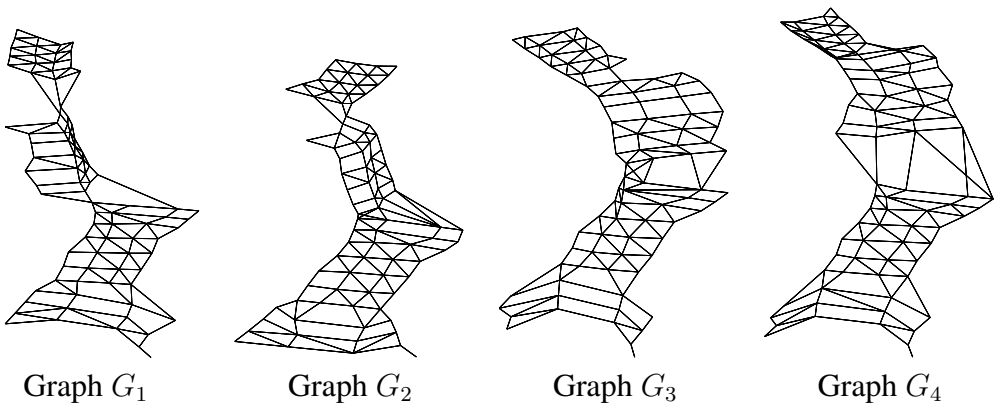


Figure 5.12: Graphs generated from the articulated object data set drawn using the optimization based approach.

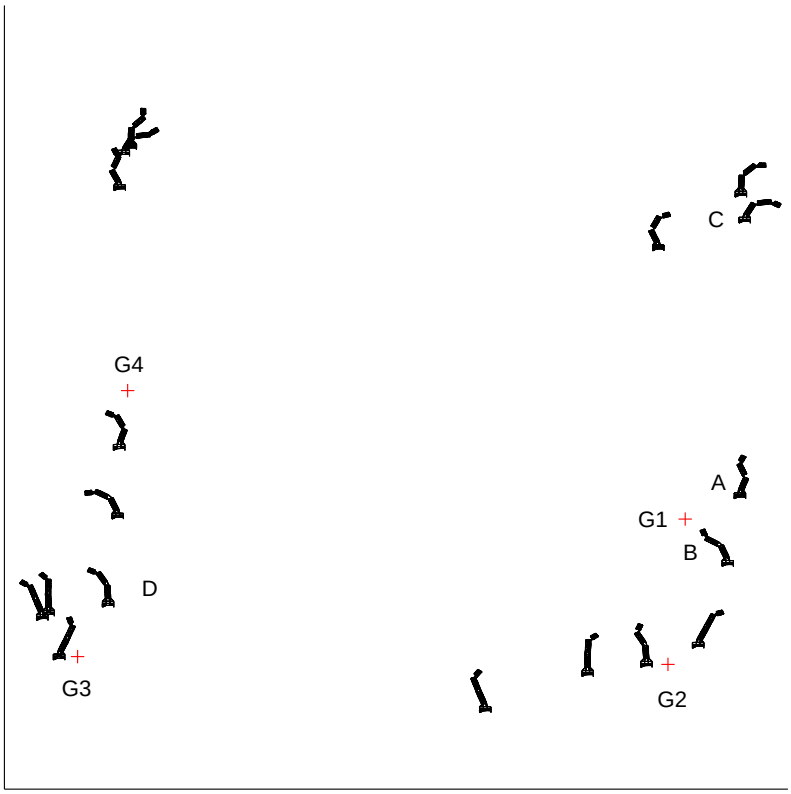


Figure 5.13: A PCA projection of the graphs in the sample set. Four generated graphs are also shown. Sample graphs are indicated by a picture of the Gabriel input graph. Graphs shown in figure 5.11 and figure 5.12 are labeled correspondingly on the plot.

to label the points by hand. Not only does this mean no alignment needs to be performed on this data but we can also partition the graphs based on our knowledge of which point is in which subgraph.

Due to the more rigid graph structures present we employ the optimization drawing method described in the introduction of this section. The data set for this experiment consists of 18 images of an object in different articulations. The Gabriel graphs constructed for 4 of these images are shown on the top row of figure 5.11. The bottom row shows the optimization based drawing of the graph above it. Figure 5.12 shows graphs that were generated using this data set. The graphs do not appear in the sample set of graphs (i.e. they truly are new graphs) however they are similar to some graphs in the sample set.

Figure 5.13 shows a PCA projection of the graphs in the sample set and the generated graphs in figure 5.12. Each sample graph is depicted by its Gabriel graph and the generated graphs are marked with crosses. The graphs shown in figure 5.11 and 5.12 are labeled correspondingly in the PCA plot. Similar graphs are grouped quite tightly as is the case of the graphs near C on the plot. The variation caused by the bottom articulation of the model does not affect the graph as strongly as the upper two articulations. This is why the graphs grouped around C have varying bottom articulations but the top two articulations are the same. Similar trends can be observed for other groups of graphs in the sample set although the groups near B and D are not as easily seen.

5.4 Conclusions

In this chapter we have shown how a parts-based generative model for graph structure may be constructed. This model is fully generative in the sense that we may sample from the model to generate new examples. The key idea underpinning the approach is that the sample graphs can be well understood in terms of the substructures they contain. Given an algorithm to produce these substructures, we can view each sample graph as a set of subgraphs and connections between subgraphs. With the decomposition of the sample graphs to hand our algorithm can commence.

The first step is to find subgraphs that represent similar substructures; this is accomplished through a clustering step. From this we obtain a reference graph for

each cluster and we can use this to solve correspondence problems on both the subgraphs and the connections between subgraphs. More importantly, this clustering step allows us to view the connections between subgraphs as connections between clusters which is essential for building the model.

To construct the model we fit three distributions; one over the clusters present in each sample graph, a second over the subgraphs present in each cluster and a third over the connections between clusters. The first is modeled using a Gibbs sampler, the second is modeled using a simple frequency selection method and for the third we transform the connection matrices for each sample graph into a long vector and then construct a normal distribution over this vector space.

Finally, to generate new examples from the model we may sample from the three distributions defined above and then combine the components produced to construct a new graph.

To show the utility of our approach we evaluated it on both synthetically generated data and real-world data. Both sets of data take the form of point sets since this allows us to easily visualize the generated graphs. The synthetic data is constructed by producing a set of points and then constructing a Gabriel graph over the point set. We showed that a new graph could be generated from this data set that combined aspects of the sample graphs. The real-world data set was constructed from images of articulated objects. Images were taken of an object in a number of positions and transformed into Gabriel graphs through a motion capture step. With the distributions defined over the data, we were able to sample new graphs from the model and visualize their structure using a custom optimization-based drawing step. Furthermore, when the sample and generated graphs are plotted using a PCA projection we show strong correlations between location on the plot and graph structure.

For the approach to work the subgraphs representing substructures of the sample graphs must be reasonably structurally similar or the clustering stage will fail to capture the variation in the set and over partition the data. The same is true for the connections between substructures, for the model of connections between subgraphs to produce meaningful results the substructures from a particular cluster should all connect in a similar way. Therefore, this approach is most effective on data types that lend themselves to a partitioning process, namely chemical structures, point sets of articulated objects and point sets of scenes.

5.4.1 Future Work

One way to extend this work would be by applying the approach to a more extensive data set. It is difficult to suggest such a data set however, due to the problems of visualizing the generated graphs. Graphs of scenes are an interesting domain as they fit the decomposition problem very well, however it is not clear how such generated graphs could be visualized. We performed some experiments with images depicting arrangements of objects but encountered some problems in constructing sample graphs that were robust to variation in the corner extraction process. Using very simple objects we did achieve some success but even then the only tool we had to display the results was to draw the graphs using an MDS embedding which failed to convey the complex structure of the graphs.

This work could also benefit from a more careful examination of the method of generating connections between clusters. When considering graphs with no weights on the edges we are modeling discrete data with a continuous distribution. A better model could be to associate a probability with the existence of each entry in the connection matrix between a pair of clusters. This could be modeled using a Bernoulli distribution as is described in Torsello's work [102].

The existence of multiple subgraphs from the same cluster appearing in a single sample graph significantly complicates the model. Currently, the multiple connection matrices between a cluster pair are merged so they may all be stored in the same space in the long vectors. If a generated graph requires multiple samples of a connection matrix between a cluster pair, then multiple long vectors must be sampled and these ignore the co-occurrence of connections. It is not clear that this is the best way to handle the situation. For example, a better approach might be to ignore the additional connections or to allow room in the long vectors to store all connections between a pair of clusters.

Neither of these two approaches would be ideal because of our underlying assumption that connections between a cluster pair should always be similar. However, this is rarely true in real world data. For example, multiple occurrences of two objects in a scene may be linked very differently in each sample. In images from the COIL data set where parts of the object remain invariant, the connections between them do not. So the solution to this problem is to remove this underlying assumption. This means extending the model to handle different types of connections be-

tween each cluster pair. One approach could be to identify different connections of a cluster pair through a second clustering step and then construct a mixture model that accounts for the different modes of the distribution of the connections. This would result in a model that was more complex but also had more representational power.

5.5 Symbols

$\mathcal{S}$	A set of graphs.
S_k	A graph from the set $\mathcal{S}$.
$\mathbf{S}_k$	An adjacency matrix representation of a graph S_k. The padded matrix is $\check{\mathbf{S}}_k$ and the aligned matrix is $\mathring{\mathbf{S}}_k$.
$\mathcal{F}$	The set of all subgraphs of the graphs in $\mathcal{S}$.
$\mathcal{F}_k$	The set of subgraphs of graph S_k.
F_{ki}	Subgraph i of graph S_k.
$\mathbf{F}_{ki}$	An adjacency matrix representation of subgraph F_{ki}. The padded matrix is $\check{\mathbf{F}}_{ki}$ and the aligned matrix is $\mathring{\mathbf{F}}_{ki}$.
$\mathbf{C}_{kij}$	A matrix storing the cut connections between subgraph F_{ki} and F_{kj}. The padded matrix is $\check{\mathbf{C}}_{kij}$ and the aligned matrix is $\mathring{\mathbf{C}}_{kij}$.
n	The size of the largest subgraph in $\mathcal{F}$.
m	The size of a cluster configuration.
ψ	A function that maps a subgraph to a cluster.
R_c	The reference subgraph for cluster c.
$\mathcal{X}_c$	The set of subgraphs belonging to cluster c.
$\mathbf{M}(F_{ki}, F_{lj})$	The matching matrix that aligns F_{lj} to F_{ki}.
$\mathbf{M}'(F_{ki})$	The matching matrix that aligns F_{ki} to its cluster reference.
$\mathbf{s}_k$	A vector storing the connection matrices for a sample graph S_k.
ξ	A function that maps a cluster pair to an index range in a vector.
z_{ab}	The largest number of connection matrices between a cluster pair a, b displayed in a sample graph S_k.
$\mathbf{\Lambda}, \mathbf{\Phi}$	The set of eigenvalues and eigenvectors resulting from an eigendecomposition of the covariance matrix $\mathbf{\Sigma}$.
$\mathbf{b}$	A parameter vector of a multivariate normal distribution.
$\mathbf{U}$	The matrix of cluster configurations for use with the Gibbs Sampler.
s, s_a, s_b	States in the Gibbs Sampler.

$h(.,.)$	Hamming distance.
H_s	The sum of Hamming distances from state s to all cluster configurations in $\mathbf{U}$.
P_s	The probability of choosing state s in the Gibbs Sampler.
G_k	A generated graph.
$\mathbf{G}_k$	A generated adjacency matrix.
$\mathbf{g}_k$	A generated vector of connection matrices.
$\mathbf{C}_g$	A generated connection matrix.
$\tilde{\mathbf{C}}_g$	A thresholded generated connection matrix.

Chapter 6

Generative Models for Chemical Structures

6.1 Introduction

The field of chemoinformatics[69, 51] is a relatively new discipline that studies the application of computational methods to chemistry. The scope of the field is wide ranging from providing databases of molecules that may be searched by molecular structure, to the simulation of molecular interactions which can help to predict how two molecules will react. From the beginning medicinal chemists have been using tools from this domain to help them design more effective drugs and better understand their properties. A combination of a better understanding of biochemistry and advances in chemoinformatics has moved the process of drug discovery away from refinements of natural products and serendipitious discoveries and towards rational drug design.

The recent advances in mapping the human genome have opened up a whole series of new drug targets[32, 35]. It is estimated[29] that as many as 25 000 different proteins will be identified and that many of these will be suitable for binding drug-like molecules to and therefore represent new drug targets. This explosion in the number of new targets calls for new ways to proceed with the process of drug discovery and exploration of chemical space. One such method that is rapidly being adopted is that of fragment based drug discovery which seeks to simplify the drug discovery process by focusing on small molecules termed fragments and combina-

tions of these fragments.

In this section we will detail a method of generating chemical structures that offers medicinal chemists another way to explore the chemical space near a set of chemically similar molecules. This type of exploration of the chemical space often occurs in drug discovery. Given a set of drugs that perform well against a target, we will show how molecules can be generated that have similar structural properties to the initial set. Given that the effectiveness of a drug is very dependent on its structure, we hope that the molecules we generate will also be effective against the target.

In the remainder of this introduction we will first offer an explanation of the drug discovery process and where this approach would fit in. Secondly, we will discuss various approaches to the problem of generating chemical structure and finally, we present our contribution.

6.1.1 Overview of Drug Discovery

Today, drug discovery[33, 67] often begins by trying to identify the biological mechanism that gives rise to a disease or infection. For example, one can find the metabolic pathways that cause the problematic state to occur and either inhibit or aid one of the links in this pathway. This often takes the form of creating a molecule, or a *ligand*, that is designed to interact with a specific *receptor*.

The receptor will generally be a small volume of space between the folds of a protein that is involved in the metabolic pathway. This volume, termed the *active site*, is the space in which the ligand will interact with the protein. The first model of this interaction was proposed by Emil Fischer in 1894 and was termed the *lock and key* model. In this model the ligand is the key and the lock is the active site in the protein. The ligand must be specific enough to only interact with the intended receptor and not interfere with other receptors. In other words, the main property of a molecule in determining its affinity for an active site is its structure. Although this model has been superseded with the *induced fit model* the basics remain the same.

For a medicinal chemist to search for molecules that would be suitable to use as ligands for a receptor, information about the structure of the receptor is required. This structure can be found by performing *X-ray crystallography* on the receptor. From this an inverse of the structure can be found and it is this volume that a ligand

must fit into. A medicinal chemist would of course take many additional factors into the design of a suitable ligand however, for the purposes of this work, we are most interested in the structural constraints.

Ligands that successfully dock with the active site can be found by several means. *High throughput screening*[53, 76] (HTS) involves testing thousands of different ligands against the receptor in controlled laboratory situations. An *assay* is developed such that when a potential ligand is applied to it the effect that ligand has on the receptor can be measured. This process is usually automated by laboratory robotics and a pharmaceutical company with a large library of drug-like molecules can screen a large number of molecules for activity against a receptor in little time.

Another possibility is to perform this screening virtually using a computer model of the target receptor and a database of drug-like molecules. Methods to perform these types of virtual screens were discussed in sections 2.2.2 and 2.2.4 of the literature review. Although these approaches can process molecules faster than their real-world counterpart, they cannot take into account all the factors that real-world experiments can. Generally the two procedures are used in tandem to perform an initial very fast screen *in silico* and then a more comprehensive screen *in vitro*. Once molecules that display high activity against a target have been found they are counter screened against a set of reference targets to check for the specificity of the molecule for the particular active site.

With several potential ligands identified (known as *leads*), the process of refinement and elaboration begins. This is known as *lead hopping* and is highly iterative. Ligands are constantly refined to improve their affinity for the receptor and their pharmacological properties[1]. Each "hop" generally involves modifying or replacing one of the ligand's *functional groups*. Functional groups are small groups of atoms within a molecule that are responsible for how it reacts. However, it is not clear how to perform this enumeration of chemical structures in an efficient way. This is due to the fact that, for a given group of structurally similar molecules, there exists many more molecules that have similar structural properties. It is the question of how to explore this area of *chemical space*[83, 71] in a meaningful way and extract ligands that have high affinity for the active site in question that our research should help answer.

[1]For example, the ADME (absorption, distribution, metabolism and excretion) method for determining the suitability of drugs for humans

Fragment Based Approaches

Traditionally, molecules screened through HTS are of approximately the same size as the drugs they may eventually become. The idea behind *fragment based approaches* [89] is the screening and use of small molecules termed *fragments*. This has a number of beneficial effects which we describe below.

First, if a fragment is found to be active against a target then it provides a good starting foundation for designing a drug. This is due to an upper limit on the weight of a molecule for it to be effective as an orally administered drug. Typically any molecule that is found to be active against a target will need to be altered to be made suitable for use as a drug, and generally these alterations involve increasing the molecular weight. Therefore, if the molecular weight of a drug candidate is near this upper limit there is less scope for refinement than there would be if the drug candidate had a lower molecular weight. In other words, it is a fragment's high binding affinity to low molecular weight ratio that is of great importance.

Second, if two fragments are found to have high affinity for a target but are structurally different then it may be that the active site has a complex geometry allowing two fragments to bind simultaneously. Since there is only a limited amount of space in the active site, we want to exploit that space as best as possible. This means finding molecules that occupy that space effectively, however, with large molecules there is a low chance of finding an exact fit. On the other hand, small fragments have a better chance of exploiting the geometric properties of the active site. If we discover two different regions of binding then this can be used by *scaffolding* two fragments together into a larger fragment that binds more successfully.

Third, the use of fragment-based screening allows chemical space to be explored in a far more efficient manner than traditional methods. Erlanson and Jahnke (Chapter 1 [57]) provide a simple example illustrating this. Given two sets of compounds, each containing 1000 fragments, we can use a linker to construct the set of all binary combinations. This set would contain 1000000 fragments which would be difficult to synthesis and screen. On the other hand, if we select only the 5 best fragments from each set then we have $1000 + 1000 + (5 \times 5)$ fragments to synthesis and screen which is far more manageable and we have almost covered the same chemical diversity (for a specific target).

Due to these reasons, fragment based drug discovery has received much atten-

tion and has already been successfully applied to a number of drug developments. By making use of fragments in our approach we might also be able to capitalize on some of the benefits they provide.

6.1.2 Discussion of Possible Approaches

In the *mixing spectral representations for graphs* and *vectorial generative models for graphs* chapters we have observed the problem of a generated graph not being quite correct with respect to the graph representation used. To recover a discrete graph from the vectorial representations used we must make use of a "thresholding step". This problem is even more pronounced when we consider graphs with a high order set of constraints governing what it means for a graph to be "correct".

In our case we will be using graphs representing chemical structure and for these to be valid they must fulfill a number of constraints that are derived from the laws of chemistry. These include constraints on which atoms can bond to others and the way in which this may be performed, permissible 3D conformations derived from the rotation of bonds and electrostatic properties to name but a few. However, we will be taking a simplified view of these constraints and will only consider chemical correctness to entail that a simple valence model is satisfied. To facilitate describing a generated graph before the "thresholding step" has taken place we will adopt the term quasi-graph or quasi-molecule to mean an instance of a representation that almost represents the chosen structure but does not meet all of the requirements.[2]

We will now detail some approaches to the problem of generating graphs that satisfy a set of chemical constraints.

One approach is to extend our thresholding step from the previous work to allow us to map a generated quasi-molecule to the nearest correct molecule by, for example, a series of edit steps. However, it would be very difficult to define such a series of edit steps. Another alternative to mapping a quasi-molecule onto a correct molecule would be to represent the molecules in a vector space. Then the distances between two molecules could be found directly. However, ensuring that the generated molecules and a selection of molecules that we could map onto reside in the same vector space is a challenging problem. Furthermore, we would also require

[2]For instance, if we have a graph with non-discrete edges but our choice of representation requires discrete edges, then we could term such a graph a quasi-graph.

an enumeration of the molecules near the generated quasi-molecules in that area of chemical space and this is an expensive operation

If we modify the way we construct the model such that it is impossible to sample incorrect chemical structure then the approach, at least in principle, becomes simpler. However, it is unclear how a distribution could be constructed in this fashion. One possibility is to use a Gibbs Sampler to move probabilistically through the space of chemical structures. Each step in the Gibbs sampler would move from one valid chemical structure to another therefore ensuring that quasi-molecules cannot be generated. However, the difficultly here is in encoding a complex relational structure into a state and ensuring that only states representing correct chemical structure can be generated.

Since molecules are highly constrained graphs they can be described in an elegant string based representation. Such strings are known as *SMILES* strings and were discussed in the literature review in section 2.2.1. Since a well-formed SMILES string can only represent valid chemical structure the problem becomes how to move between well-formed SMILES strings. One possibility would be to define edit operations that could be applied to specific substrings inside the SMILES strings. SMARTS patterns (which are basically extended regular expressions) could be used to identify substrings that could be substituted with other substrings such that the SMILES string as a whole still represented a valid chemical structure. However, assigning probabilities to these edit operations so that we could move through chemical space in a meaningful way would be difficult.

Since molecules are highly decomposable, as discussed in the overview of fragment based approaches, the parts based approach detailed in chapter 5 of this thesis could be applicable. The segmentation step would decompose a molecule into its functional groups and these would form the subgraphs that are used in the method. If constraints were placed on how the set of clusters were selected and which connections were possible in the connection matrices then it would be possible to only generate valid chemical structure. However, encoding these constraints in the model would be difficult.

Another possibility is to generate new molecules directly by using a reaction based model[68, 6]. A pool of fragments representing the constituent parts of the input set are computed and then new molecules can be created by reacting these

together. If we compute the probability that a certain reactant appears in the input set (or gives rise to a molecule that appears in the input set) then we can define a distribution over the set of reactants. Using these as the input to a reaction simulator would be a way to generate new molecules probabilistically. This method could be used exclusively or could be part of the process of a more complex method.

For example, this could be a way to generate a set of valid chemical structures that are similar to those in the input set. Then when a quasi-molecule is produced it could be mapped onto a molecule generated by a reaction based approach. One of the main advantages of this method is that molecules generated in this way could be more easily synthesized in the real world than those constructed without any regard for the synthesis method. On the other hand, domain knowledge is required to specify the reactants and the reactions.

6.1.3 Contribution

The approach we select is the first described briefly in the previous section; we work in a vector space where points in this space represent both valid and invalid chemical structures. After embedding the sample set in this vector space, we construct a model that is capable of producing vectors describing almost, but not quite correct chemical structures. Where we produce invalid chemical structures we map these to valid chemical structures using a projection step. To obtain a set of valid chemical structures suitable for use as the range of our projection function we enumerate part of the chemical space in a controlled way. We will term the set of molecules we begin with the *input set*, the set of molecules we enumerate the *projection set* and the set of molecules we generate the *generated set*.

Therefore our approach consists of the following main steps:

- Construct the projection set.
- Sample from a distribution on the input set to generate new quasi-molecules.
- Map the generated quasi-molecules onto the molecules in the projection set to obtain the final set of valid generated molecules.

The key idea is that although the set of projection molecules do not represent a sample from the input distribution, they are similar enough such that if we se-

lect a subset intelligently then those graphs in the subset are acceptable as samples. On the other hand, the set of generated quasi-molecules we create are drawn from the true distribution of input molecules. We therefore find the subset of projection molecules that are suitable as samples by projecting a true sample from the input distribution (a generated quasi-molecule) onto the most similar "correct" molecule from the projection set. In this way we can approximate sampling the input distribution without the possibility of generating a graph describing a chemically incorrect molecule.

We have chosen this method since we feel it has the most promise and fits in best with the other topics pursued in this thesis. The approach is an amalgamation of our earlier work on vectorial generative models for graphs and the initial step from the parts based approach. In addition we introduce our method of enumerating the chemical space around a small group of molecules and significantly extend our previous thresholding step from a simple function with binary output to a complex projection between quasi-molecules and valid molecules.

Furthermore, we believe this process will be of use to medicinal chemists looking to explore chemical space for more potential drug candidates. After a suitably large number of ligands have been identified that have all been shown to bind well to a target receptor, these can be used to populate the input set of our approach. We can then use our model to generate new molecules which are structurally similar to the input set. Although we do not expect them to display all the pharmacological properties those in the input set did, we do expect them to be similar in structure and therefore perform well at interacting with the target receptor.

Of course, these interactions take place in 3D while our approach only considers an abstract 2D structural representation. Therefore, the final stage of our approach is to compute a 3D conformation for each generated 2D structure. Therefore, there will be situations where a molecule which appears promising when represented as a graph, is not so successful when represented in 3D.

However, the set of possible 3D structures that can be generated from a 2D representation is heavily constrained by the laws of chemistry so this problem is not as significant as it initially seems. Indeed, there are many examples in the literature where 2D approaches successfully generate useful 3D structures[52, 31, 28]. For the datasets we employ we have found that there are typically 50-150 low energy

3D conformations possible for each generated structure.

6.1.4 Chapter Overview

We begin by detailing our method which is given in section 6.2. Next we provide comprehensive results for two data sets in section 6.3. One data set targets the COX2 protein and the second targets the EGFR protein. Finally, in section 6.4 we give our concluding remarks and outline some future work that could be pursued in this line of research. A table of the symbols used in this chapter is given in section 6.5 on page 194.

6.2 Method

We will begin this section by giving an overview of our method, commencing with the first phase which consists of the construction of the projection set. Next, we will discuss the second phase which is the alignment of the input set and the projection set. The last phase is the construction of the model of the input set, which is a *Gaussian Mixture Model* (GMM), and finally mapping samples from this model to the graphs of the projection set. Please consult figure 6.1 for a diagrammatic overview of the method.

At first glance it may seem an overwhelming task to enumerate the chemical space near the molecules of the input set, but we can approximate an enumeration of chemical space using statistical methods. In fact, if we adopt a fragment based approach, we already have much of the information about the building blocks that will allow us to enumerate molecules; the fragments of the input molecules consisting of functional groups and carbon scaffolds. We will construct a model that allows us to generate new correct chemical structures, according to our simplified model of chemistry, which are composed of the fragments present in the input molecules. We will discuss this step in detail in section 6.2.3. The methods used in this step are similar to those in the initial step of the parts-based generative models approach.

We then turn our attention to the problem of performing alignment and computing the similarity of molecules. To model the distribution of the input molecules we will require an alignment of only the molecules in the input set. This will allow us to

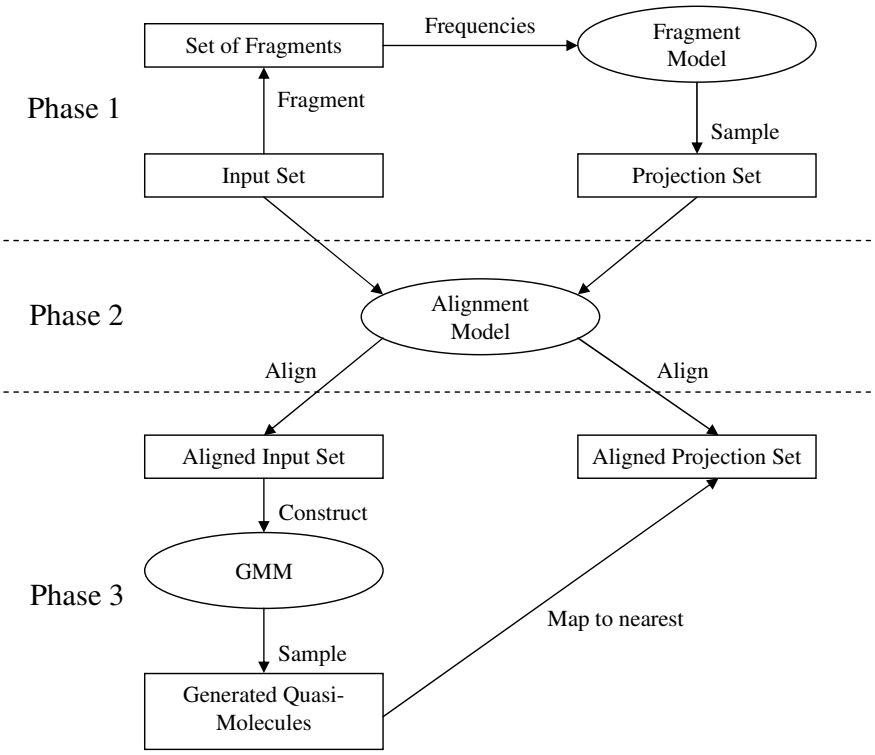


Figure 6.1: An overview of our method for generating chemical structure.

construct a distribution over a vectorial representation of each molecule. Therefore, generated quasi-molecules will be of a vectorial form.

On the other hand, to map generated quasi-molecules to the projection molecules we will require a similarity measure. There are a number of ways to accomplish this efficiently such as computing features like fingerprints (see section 2.2.1). However, this similarity measure must work with the vectorial form of the generated quasi-molecules. Therefore, we require that the projection set and the generated quasi-molecules will reside in the same space. The best solution to accomplish this is to perform a global alignment over the input and projection sets. This will allow the similarity of two molecules to be assessed directly in the vector space and also provide us with a visualization method for all three sets.

Therefore, we will require a step that aligns not only the small number of molecules in the input set but also the large number of molecules in the projection set. Due to the size of projection set this step must be as efficient as possible while still producing high quality alignments. We solve this problem by performing a hierarchial alignment on the input graphs and then "slotting in" each projection

graph to the position in the alignment tree of the input graph that it is closest to. The benefit of this is that a full pairwise alignment only needs to occur between the graphs of the input set. This process, as well as other approaches considered, are described in section 6.2.4.

Lastly, we discuss how we construct a Gaussian mixture model over the input molecules and use this to sample new quasi-molecules. To estimate the parameters of the GMM accurately we actually work in a reduced version of the vector space and therefore new samples are produced in this space. However, we are then able to move back to a common space to compute the mappings between the generated quasi-molecules and the projection molecules. These steps are described in section 6.2.5.

We have spoken informally about the input set and the projection set in the introduction, we will formalize these descriptions both here and in the following sections. Our approach commences from a set of molecules. We construct a graph describing the chemical structure for each molecule in the set. We then collect these graphs into the set $\mathcal{S}$. We call this the *input set* and it is the distribution of the graphs in this set that we will sample to generate new chemical structures. We will begin the discussion of our method with a description of how we represent graphs of chemical structure (section 6.2.1). Following this we will explain our simplified model of chemistry in section 6.2.2. Finally, beginning in section 6.2.3, we give a detailed description of our approach.

6.2.1 Representation of Molecules

Beginning from a set of input graphs $\mathcal{S}$ we construct an adjacency matrix $\mathbf{S}_k$ for each graph $S_k \in \mathcal{S}$. In each graph, the vertex set represents the atoms of the molecule and the edge set represents the bonds between atoms. The weight functions $w_{\mathcal{V}}$ and $w_{\mathcal{E}}$ assign a weight to every atom and bond respectively.

Although we could assign a weight to an atom based on its atomic number this would result in a large number of essentially arbitrary weights. This choice of weight function was used by Wilson [121] however, as the work involved classification and not generation, the weight function could be tuned to improve the performance of the classifier. There is no obvious tuning method that would work with our generative approach.

Instead, we make use of the fact that the properties of an atom are largely dependent on its outer electron shell configuration, or in other words, the number of valence electrons it possesses. Non-metal elements are divided, by valence configuration, into eight groups in the periodic table that give indications of how they will behave. We exploit this for determining the weights to assign to atoms in our matrix representation. We consider two atoms to be similar if they have similar valence configurations.

We begin by assigning atoms from the nobel gas family a value of 1.0. Next we set atoms with seven valence electrons to 0.8, six valence electrons to 0.7, and so on down to atoms with one valence electron being assigned a weight of 0.2

Wilson [121] also encoded the end points of a bond into the weight function for edges. For example, the bond H-O would have a different weight to H-N. However, in our method this approach introduces unnecessary information to the graph representation, and therefore a simpler solution will suffice. Edge weights are assigned by bond strength: single bonds are assigned a weight of 0.5 and double bonds are assigned a weight of 1.0.

6.2.2 Implicit Hydrogen Model

As mentioned earlier, graphs describing chemical structure are constrained by the laws of chemistry. To reduce the complexity of the approach we adopt a very simple model of chemistry which we will describe in this section.

Using our method, it is only possible to generate chemical structures that are present in the projection set. Therefore, the sole point of failure to generate correct chemical structures lies in the construction of the projection set. We have discussed earlier, in the outline, that we will be using a fragmentation approach to segment the input molecules and then reassemble them into new molecules. This process is highly applicable to our evaluation domain as it explores the possible structural configurations of the functional groups present in the input molecules. Furthermore, it does not require an experienced chemist to oversee the construction of the projection set, as would be necessary with a reaction based method. However, the approach can result in the construction of invalid chemical structures which can occur in two ways. We describe these next.

The first type of problem occurs when considering the bond created by joining

two fragments. It is quite likely that the atom to atom bond that connects the two fragments will differ to the one that the fragments were previously involved in. Therefore, it is unlikely that the valency requirements of the fragment complex will be satisfied. Furthermore, the bond could involve charges on the connecting atoms or possibly be a double or triple bond.

The second problem occurs as a property of a fully assembled molecule. Since fragments are joined with no regard for the 3D structure of the molecule, when the 3D structure is considered for docking it is possible that fragments will overlap. This will clearly result in an invalid chemical structure.

The first problem is solved to a degree by the model outlined below. On the other hand, solving the second problem is more difficult. One approach would be to generate 3D configurations of the molecule throughout the stages of its construction, at each stage checking that a valid 3D conformation can still be generated. However, as our generative model is 2D in nature we ignore this 3D constraint and simply record the molecule as a failure if no valid 3D conformation can be found[3]. We should note however, that molecules containing invalid 3D structure usually fall outside the distribution of input molecules and are therefore rarely selected as generated molecules.

To solve the problem of constructing correct bonds between two fragments we make use of the following observations.

- Since we are dealing with small drug-like organic molecules which very rarely contain metals, almost all bonds should be covalent. Furthermore, organic compounds only consist of carbon atoms, hydrogen atoms and functional groups.

- Our procedure for segmenting a molecule will not break aromatic structures, double bonds or functional groups apart. The majority of bonds it breaks are between two of the following atoms: carbon, oxygen and nitrogen.

From the first point we may assume that only covalent bonds will be present in the input set and when joining two fragments, we only require the creation of covalent bonds. Furthermore, from the second point we may assume only single

[3]Note that finding the 3D conformation of a generated molecule is not part of our generative model, we consider it a postprocessing step.

covalent bonds will be required and we will never join directly into the middle of an aromatic structure or a functional group.

To simplify the joining process we use an implicit hydrogen model where charge information must be included on the atoms or disregarded. This is in contrast to an explicit hydrogen model, where all hydrogen atoms are included in the model and can therefore be used to determine charge information on atoms.

In an implicit hydrogen model, the valence requirements on an atom can be satisfied by the addition of an appropriate number of bonds to hydrogen atoms, therefore resulting in correct chemical structure as least in terms of a simple valence model. Furthermore, this removes the problems of joining two fragments when one of the connecting atoms is an ion. Therefore, in our model, all hydrogen atoms are made implicit from the start and charge information is removed. Our method commences and new chemical structures are generated. When a 3D conformation of a generated molecule is required for docking, the implicit hydrogen model is used to augment the molecule with hydrogen atoms where needed to make the molecule complete in terms of our simple valence model.

In addition to simplifying the joining procedure, there are many advantages to choosing an implicit hydrogen model. Firstly, since the molecules consist of less atoms the resulting model is less complex. Secondly, the alignment procedure will give better results as the core structures which must be aligned are not influenced by hydrogen atoms. Third, the removal of charge information does not significantly distort the structure of a molecule, which is what we are primarily concerned with. Finally, correct valence configurations and resulting charge information can always be added to interesting molecules by a chemist if our model does not produce the correct configuration. Molecules containing ions are actually quite rare in the data sets we employ, none of the molecules in the COX2 data set contain ions and only 34% of molecules in the EGFR data set contain charge information. As a result we often generate correct chemical structure by only using our simple model.

6.2.3 Constructing the Set of Projection Molecules

The aim of this part of the process is to construct the set of projection molecules which we will project generated quasi-molecules onto. We term this set of molecules the *projection set* with the symbol $\mathcal{P}$ since these graphs will form the range of the

projection function that maps a generated quasi-molecule to a valid molecule. The first step is to segment each graph in $\mathcal{S}$ into its constituent *fragments*: functional groups, carbon scaffolds and aromatic rings. We then construct a model describing the probability with which each fragment appears in the input graphs. This is similar to a combination of the "clusters within each sample graph" and "frequency of subgraphs within a cluster" distributions constructed in the parts-based generative models but essentially a much simpler model. Finally, we use this model to construct the graphs in the projection set by repeatedly combining fragments identified in the first step with probabilities computed in the second step.

Segmenting the Input Molecules

Graph segmentation is a well known topic in pattern recognition and computer vision and there are many algorithms that perform well at the task. Perhaps the best known algorithm is Shi & Malik's Normalized Cut [100]. However, for the purposes of segmenting chemical structures, a more specialized method is required that considers the chemistry of the molecule as well as the structure of the graph.

As discussed earlier, drug discovery often proceeds by making small changes to drug-like molecules by altering their functional groups, in an effort to make the molecule bind more tightly to an active site or improve other desirable properties. Since we will be building the set of projection molecules from the fragments identified in this step we would like the way we segment the molecules to reflect the way a molecule may be improved by the method just described. In practice this means that while segmenting molecules we want to produce clear functional groups with no extra or missing atoms. A molecule cannot be completely decomposed into functional groups and after they have been removed we are often left with one or more carbon scaffolds. Since we want to assemble full, useful molecules in the projection set we include the scaffolds in our fragment model.

The tool *Chomp* from the *OEChem Toolkit* [5] provides some of the functionality we require. It takes a molecule file and a set of SMARTS rules that specify which bonds should be broken and returns a set of fragments. The fragment set it returns is simply a list of molecules with atoms on the other side of broken connections replaced by stars. We use $\mathcal{B}_k$ to describe the set of fragments that are returned by segmenting input graph S_k, .

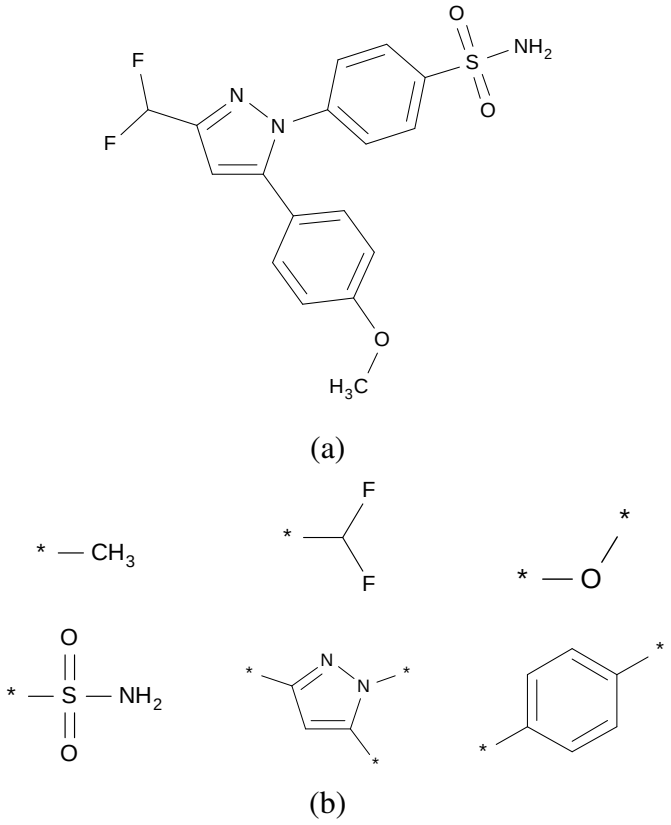


Figure 6.2: The fragments resulting from running Chomp on the molecule (a) are shown in (b).

Figure 6.2 shows the result of applying Chomp to a molecule. As can be seen in the figure, Chomp takes multiplicity into account and so, if two identical fragments are identified, only one copy is returned. This is a problem for our application since we require information on the frequency with which each fragment occurs. Given a set of correspondences between the atoms in the fragments and the atoms in the molecule we could use this to compute the frequencies indirectly but unfortunately Chomp does not provide this facility.

To compute the frequencies of fragments that occur more than once in a molecule we use a custom program built from the OEChem Toolkit. The program makes use of the `OESubSearch` class from the toolkit which provides exact substructure search capabilities for molecules. We perform a substructure search for all the fragments identified by Chomp and record the correspondences. Since smaller fragments can sometimes be found within larger fragments we cannot simply infer the frequencies from the correspondences. Instead we must match the functional groups to the atoms in the molecule in order of the largest to the smallest and record which

atoms in the molecule have been used in this stage. We then match the carbon only fragments without using any atoms that have already been marked as participating in a match. Again as smaller carbon only fragments can be subgraphs of larger carbon only fragments we must match in the order of largest to smallest. When all the atoms in the molecule have been matched to atoms in the fragments we can then compute the fragment frequencies using the matches. This procedure results in a function ζ that maps a fragment $B_{ki} \in \mathcal{B}_k$ to an integer n that specifies how many times that fragment occurs in input molecule S_k.

$$\zeta : B_{ki} \rightarrow n \tag{6.1}$$

Building a Fragment Set

At this stage we know the fragments in each input molecule and the frequencies with which they occur but only within that molecule. To construct our model we require the frequency of fragments occurring across the whole set of input molecules. We accomplish this by matching the fragments identified in all input molecules to find the set of unique fragments contained in the input molecules. Let us begin by defining a set $\mathcal{B}$ that consists of all fragments found in the input graphs.

$$\mathcal{B} = \bigcup_{k=1}^{|\mathcal{S}|} \mathcal{B}_k \tag{6.2}$$

We then compute a set $\mathcal{F}$ of unique fragments as follows:

$$\mathcal{F} = \text{unique}(\mathcal{B}) \tag{6.3}$$

where the unique(.) operator returns the set of unique fragments present in the argument. Therefore, the set $\mathcal{F}$ consists of a single occurrence of every fragment found in the input molecules and each $F_i \in \mathcal{F}$ represents a unique fragment. However, we must retain the number of occurrences of each unique fragment to build our statistical model. This is accomplished as such: for each fragment $F_i \in \mathcal{F}$ the set $\mathcal{F}_{ij}$ contains all instances of the fragment F_i in $\mathcal{B}$ where it occurs j or more times:

$$\mathcal{F}_{ij} = \{\forall X \in \mathcal{B}, X = F_i, \zeta(X) \geq j\} \tag{6.4}$$

We can then compute a distribution $P_i(j)$ that gives the probability of a fragment F_i occurring j times in a single input molecule. It is defined as such:

$$P_i(j) = \frac{|\mathcal{F}_{ij}|}{|\mathcal{S}|} \tag{6.5}$$

To compute the set of unique fragments $\mathcal{F}$ in equation 6.3 we require a method of determining whether two graphs are isomorphic. We could use a general graph matching algorithm such as Gold & Rangarajan [48] but, since we are dealing with chemical structures and we are interested only in exact matches, we can use a more application specific method. We turn to the OEChem Toolkit again and make use of the `OEExactGraphMatch` function to compare two fragments. This function is very fast since it compares molecules by their canonical smiles strings. The work involved only requires constructing the canonical smiles strings for the two molecules to be matched and then a string comparison.

Constructing the Projection Molecules

To construct a new molecule from the projection set $P_k \in \mathcal{P}$ we must first decide which fragments it should be composed from. We find this by using the model we constructed in the last section to take a sample of the distribution of fragments present the input graphs. The result of this process is a *fragment list* describing the fragments that should appear in our new molecule and with what multiplicity.

For every fragment $F_i \in \mathcal{F}$ and every multiplicity j we sample from $P_i(j)$ to determine if that fragment with that specific multiplicity will occur in our new molecule. In practice this means generating a random uniform number in the interval $[0, 1)$; if it is less than the value of $P_i(j)$ we add that fragment and multiplicity to our fragment list. If we find that a fragment F_i should occur with multiplicity j and k then we disregard the sample with the lower multiplicity. Note that the fragment list only serves as a suggestion since it may not be possible to include all fragments in the list due to a lack of available connection points.

To produce a new molecule from the fragment list we begin with a molecule with no atoms in it i.e. an empty graph. We then repeatedly join a fragment from the fragment list to the incomplete new molecule. When joining the first fragment the new incomplete molecule simply becomes equal to that fragment. After the

first join however we must identify a connection atom on the incomplete molecule and then join the fragment to that connection atom. Clearly we do not want to run out of connection points until we have joined all atoms in the fragment list but this is not always possible. However, to ease the problem we proceed by joining the fragments in order of the number of connection atoms. We start with the fragments with the most connection atoms and finish with the terminal fragments (fragments with only one connection atom). At each join if there is more than one connection atom available then the one used to join a new fragment is selected at random. This joining process continues until we either run out of fragments in the fragment list or run out of connection atoms in the new molecule to join fragments to. In the second case the new molecule can be declared complete. However, in the first case the new molecule will still have connection atoms present. Due to our simple valence model described earlier, we can disregard these unused connection atoms and still produce correct structure. This is because our implicit hydrogen model will correct any valency problems resulting from unused connection points on fragments through the addition of hydrogen atoms.

Finally, we perform a filter of the new projection molecules to make sure they exhibit properties similar to those in the input set. This can either be a coarse threshold on the number of atoms a molecule should have or a more sophisticated test of the pharmacological properties of the molecules by using a tool like *filter* from the OEChem Toolkit.

6.2.4 Aligning the Molecules

We explained earlier that the best solution to the problem of alignment and computing similarities is to require that our vectorial descriptions of the input molecules $\mathcal{S}$, projection molecules $\mathcal{P}$ and generated quasi-molecules all eventually reside in the same vectorial space. In practice this means that we must align all molecules from the input and projection sets.

In the following sections we describe our approaches to solving this alignment problem for large sets of molecules and describe some of the difficulties that arise when working with these types of graphs. Before proceeding we note that we pad all adjacency matrices so they are the size of the largest graph in $\mathcal{S} \cup \mathcal{P}$.

Aligning to the Largest Molecule & Associated Problems

Our initial solution to this problem was to identify the largest molecule in the set of input and projection molecules and then use this as a reference molecule to align all other molecules. We used the graph matching algorithm of Gold and Rangarajan [48] on the padded adjacency matrices to perform this alignment.

We found that when all the molecules were approximately the same size as the reference graph then this was an adequate solution. However, if the molecules differed greatly in the number of atoms then this solution had significant issues. The problem arises when the reference molecule contains two smaller almost identical molecules as subgraphs. If these molecules occupy approximately the same subgraph then there is no problem as they will be mapped to approximately the same part of the vectorial space. However, consider the case when the two subgraphs lie at opposite ends of the molecule. Then each smaller molecule will be aligned to a different part of the reference graph and in turn be mapped to a different part of vectorial space.

While this is correct within the context of an individual alignment, it is problematic when constructing a statistical model over the vectors as the the same component of two input vectors would correspond to different structures. This problem is a direct result of there being no ordering over the vertices of a graph and is especially problematic for chemical structures because larger molecules are frequently composed of smaller molecules. The result of this is a poor vectorial representation of the molecules.

The key problem here is that we have chosen a bad representative for the set, but the only other representatives we could choose would be smaller. If we were to choose a smaller representative then information would be lost when we aligned larger molecules to this small representative molecule.

Clearly it is not possible to align the molecules in the two sets using a single representative due to the large diversity in the projection set (and to a lesser extent the input set). Therefore, in the following sections we consider solutions with multiple representatives.

Partial Hierarchical Alignment Based on a Secondary Distance Measure

In this section we consider a solution to the alignment problem using a partial hierarchical alignment. We select a small set of reference graphs from the input and projection sets. To obtain this set of reference graphs we cluster the molecules in the input and projection sets. From each cluster we select a reference graph that has the minimum sum of distances to all other graphs in the cluster. We then align the remaining molecules in each cluster to their cluster reference graph to obtain a global alignment.

The chain of reference graphs is pre-aligned from largest to smallest. We denote a sequence of reference graphs as $R_1, R_2, ..., R_n$ where the indices indicate the ordering of graphs based on the number of vertices. R_1 is the largest graph, R_2 is the second largest graph, etc.

We align the reference set in a chain starting with the second largest. If we let $\mathbf{M}(R_i, R_j)$ be the permutation matrix that aligns R_j to R_i and $\check{\mathbf{R}}_i$ be the padded adjacency matrix for R_i then the second largest graph is aligned to the largest as follows.

$$\mathring{\mathbf{R}}_2 = \mathbf{M}(R_1, R_2)\check{\mathbf{R}}_2\mathbf{M}(R_1, R_2)^T \tag{6.6}$$

This results in the aligned adjacency matrix $\mathring{\mathbf{R}}_2$. We then proceed to align the third largest graph using the result of the previous alignment, $\mathring{R}_2$.

$$\mathring{\mathbf{R}}_3 = \mathbf{M}(\mathring{R}_2, R_3)\check{\mathbf{R}}_3\mathbf{M}(\mathring{R}_2, R_3)^T \tag{6.7}$$

This process continues until the whole reference set has been aligned.

This method requires a set of clusters of the input and projection molecules to be computed, and to perform clustering we require pair-wise distances between all molecules. Once we have the distances we can use Normalized Cut to compute the clusters. Ideally we would like these distances to be the same as those computed by the algorithm we use for graph alignment. However, computing the pair-wise distances for a large set of graphs using a full graph alignment algorithm is a very computationally expensive procedure. For this reason we must employ a secondary distance measure that is much faster to calculate than a full graph alignment and will provide the set of pair-wise distances we require.

The *fingerprint* of a molecule is a feature based representation that allows the similarity of two molecules to be assessed. A full description of molecular fingerprints is given in section 2.2.1. Both the computation of fingerprints and distances between fingerprints can be performed quickly allowing us to obtain a full set of pairwise distances efficiently.

In the previous section discussing the construction of the projection molecules we described the use of fragments in the process. Each of these fragments are exactly the kind of structural pattern that a bit in a fingerprint might represent. Given that we already have complete knowledge of which fragments are present in the input and projection molecules we constructed a fingerprint representation of each molecule in $\mathcal{S}$ and $\mathcal{P}$ using this information. It turned out that these fingerprints were a poor representation of structure. While they captured the fragments present in each molecule perfectly, they contained little information on how the fragments were connected. As a result there was a large discrepancy between the distances reported by the fingerprint method and the full graph alignment method. As such using this distance measure resulted in the production of a poor set of clusters.

Next we considered a more general fingerprint method that has better representational power. In this method we do not compute the fragments a fingerprint identifies ourselves, instead we use a general set of fragments that are known to give good feature descriptors of molecules. The tool we employ is called *Babel* from the *Open Babel Toolbox* which is a program to convert between different chemical structure file formats. One of the formats it supports are *Daylight style fingerprints* [1]. Using this fingerprint representation we obtained distance measures much closer to those produced by the graph alignment algorithm. However, there is still enough error present to produce poor clusters and representative molecules.

The reason for the difference in distance measure is due to how each approach views structure in the graph. In the fingerprint method the paths present in the graph are of large importance while in the graph alignment method of Gold and Rangarajan the edit distance can be considered more important. To see where this results in a large difference consider a large molecule consisting of a long chain of atoms. If a bond connecting two atoms near the middle of the molecule is removed then the set of paths present in the molecule are significantly changed, resulting in a significant change in the bits of the fingerprint. However, in terms of graph edit

distance this only corresponds to the removal of one vertex and results in a smaller change.

Although a secondary, cheaper to calculate, distance measure appeared to be a good solution we could not find one that displayed approximately the same behavior as the full graph alignment distance measure and this discrepancy resulted in a choice of poor clusters and reference graphs.

Hierarchial Alignment of the Input Set

We mentioned earlier that finding the set of representatives through computing the pair-wise distances using a full graph alignment algorithm was infeasible. This is because the number of distance computations required grows as a square of the total number of graphs involved in the alignment: $O((|\mathcal{S}| + |\mathcal{P}|)^2)$. However, if we only allow the graphs in the input set to be representatives then the problem is alleviated to some degree. In other words we reduce the total amount of work that adding another projection graph incurs since we must only compute the distances between the new projection graph and all molecules in the input set. The complexity of the approach then reduces to $O(|\mathcal{S}|^2 + |\mathcal{S}||\mathcal{P}|)$. We restrict only graphs in the input set to be representatives since we expect them to be better choices for representing typical graph structure of the distribution than the projection graphs.

The alignment procedure using this approach proceeds as follows. We must first align the graphs of the input set which we accomplish through a pairwise hierarchial alignment. We then compute the distance to each input graph for each projection graph and align each projection graph to the input graph at minimum distance.

To compute the hierarchical alignment of the input graphs we must compute the distances between all pairs of graphs in the input set. We obtain this pair-wise distance matrix by aligning each pair of input graphs in both directions and then averaging the distances, thus the distance matrix is symmetric. We record the distance between input graph S_i and S_j in the distance matrix $D(S_i, S_j)$. We construct the hierarchial alignment tree by computing the set of graphs that will be present at each level of the tree, starting from the deepest level first: d. Using the matrix $\mathbf{D}$ we approximately solve the *stable roommates problem*[56] which involves finding the set of pairings such that the total sum of the distances of the pairings is minimized. In each pairing the smaller graph will be aligned to the larger one and then the larger

one moves up to the next level of the tree (at depth $d-1$). At depth $d-1$ of the tree we have approximately $\frac{|\mathcal{S}|}{2}$ graphs and we remake the distance matrix $\mathbf{D}$ to include only graphs present at this depth. Again we approximately solve the stable roommates problem on $\mathbf{D}$, align the smaller graph of each pair to the larger and move the larger graph up to the next level of the tree (depth $d-2$). This process repeats until we are left with one graph that is the root of the tree, which by definition must be the largest graph in the input set.

With the tree constructed we can then apply all the alignments to the padded adjacency matrix representations for each graph. We describe the alignment for the input graphs first. We commence from the leaf representing the input graph: if it is the larger of the pair then we need not perform any alignment at this stage and we may move to the next level. If it is the smaller of the pair then we must align it to the larger before moving up to the next level. At the next level we repeat the procedure, if it is the larger then we move up with no alignment and if it is the smaller then we align it to the larger and then move up. This continues until we reach the root. We denote the sequence of alignments required for input graph S_i as $\omega(S_i)$.

Applying the alignment to the projection graphs is very similar. We require the computation of a new distance matrix $\mathbf{D}'$ of size $|\mathcal{S}|$ by $|\mathcal{P}|$ where the distance between input graph $S_i \in \mathcal{S}$ and projection graph $P_k \in \mathcal{P}$ is stored in $D'(S_i, P_k)$. We can then find the input graph S_i closest to projection graph P_k as follows:

$$i = \underset{j}{\operatorname{argmin}}\ D'(S_j, P_k) \tag{6.8}$$

We denote the matrix $\mathbf{M}(S_i, P_k)$ as the permutation matrix that aligns P_k to S_i. We align the projection graph to the closest input graph as calculated above resulting in the following aligned adjacency matrix $\mathring{\mathbf{P}}_k$ for graph P_k:

$$\mathring{\mathbf{P}}_k = \mathbf{M}(S_i, P_k)\check{\mathbf{P}}_k\mathbf{M}(S_i, P_k)^T \tag{6.9}$$

To finalize the alignment we must apply the sequence of alignments $\omega(S_i)$ to the adjacency matrix computed in equation 6.9.

We illustrate this procedure with an example tree. Figure 6.3 shows a small hierarchial alignment tree consisting of five graphs, four input graphs $\mathcal{S} = \{S_1, ..., S_4\}$ and one projection graph $\mathcal{P} = \{P_1\}$. The larger graph is always the left child in this

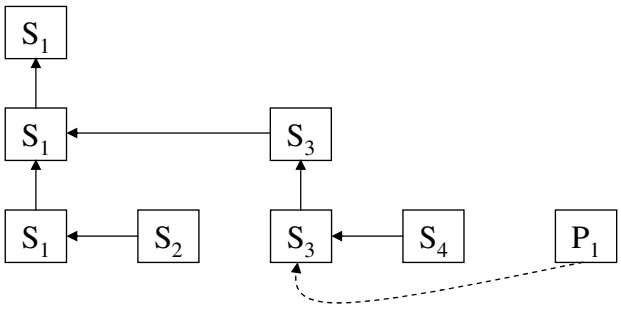


Figure 6.3: An example pairwise hierarchial alignment tree.

example. To align S_4 we must first align it to S_3 resulting in the adjacency matrix

$$\mathbf{M}(S_3, S_4)\check{\mathbf{S}}_4\mathbf{M}(S_3, S_4)^T$$

We then move up to the next level and note that S_3 is not the larger of the pair so we must apply the alignment of S_3 to S_1 resulting in the adjacency matrix

$$\mathbf{M}(S_1, S_3)\mathbf{M}(S_3, S_4)\check{\mathbf{S}}_4\mathbf{M}(S_3, S_4)^T\mathbf{M}(S_1, S_3)^T$$

.

We are now at the root of the tree so no further alignment is necessary. This sequence of alignments is given by $\omega(S_4)$. To align the projection graph P_1 we must first align it to its the input graph at minimum distance which is S_3 resulting in the adjacency matrix

$$\mathbf{M}(S_3, P_1)\check{\mathbf{P}}_1\mathbf{M}(S_3, P_1)^T$$

We then apply the sequence of alignments $\omega(S_3)$ which are as follows: S_3 is the larger graph of the pair so we move up the tree. Next we find S_3 isn't the larger graph of the pair so we align to S_1 resulting in the following adjacency matrix:

$$\mathbf{M}(S_1, S_3)\mathbf{M}(S_3, P_1)\check{\mathbf{P}}_1\mathbf{M}(S_3, P_1)^T\mathbf{M}(S_1, S_3)^T$$

We are now at the root so the process is complete. A full pairwise hierarchial clustering tree for the input molecules from the COX2 data set is shown in figure 6.4. Again the left child is always the larger of the two. Where alignment occurs the distance between the two graphs is written above the middle of the line.

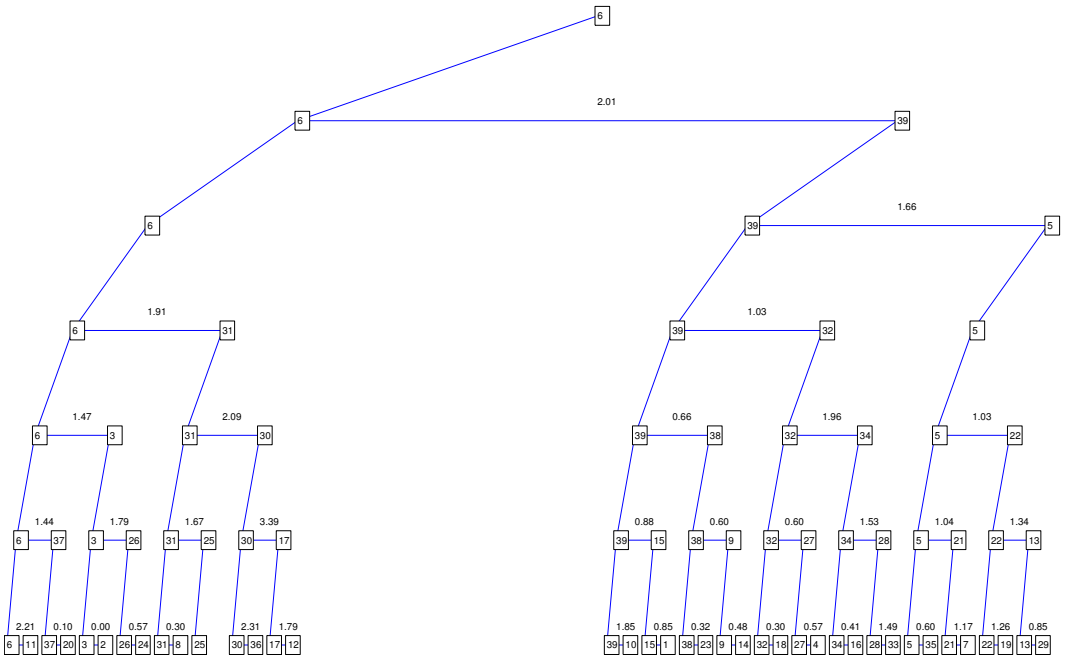


Figure 6.4: The pairwise hierarchial alignment tree from the input molecules of the COX2 data set.

6.2.5 Generating New Molecules

With the projection set now constructed and graphs from both $\mathcal{S}$ and $\mathcal{P}$ aligned we proceed to produce a set $\mathcal{G}$ of generated graphs. To model the variation in the input graphs we fit a Gaussian mixture model over the aligned vectorized adjacency matrices of the input graphs. However, as these vectors reside in a very high dimensional space it is difficult to estimate the parameters of the GMM directly in this space. To solve this problem we reduce the dimensionality of the vectors describing the input graphs by projecting them into a space determined by performing PCA on the input set. In this reduced space we still capture the majority of variation in the input set and can also successfully estimate the GMM parameters. From this we can generate new vectors in this reduced space describing new quasi-graphs.

However, to complete the generation step we require a mapping from each generated quasi-graph to the graph from the projection set that it is nearest to. In other words we need to map the generated vectors to the real molecules in the projection set. At the moment the projection set and the generated graphs reside in different spaces so it is impossible to define such a distance measure and compute a mapping. There are three possible solutions to this problem.

Firstly, we could use the previously calculated PCA projection to project the

projection set into the same space as the generated vectors. The problem with this approach is that the PCA projection was constructed from only the input set and therefore does not represent the variation in the projection set effectively. If the input set and projection set are very similar then this would be an acceptable solution however, we expect the projection set to contain new structures and hence it is unlikely that this will be the case.

A second solution would be to use the inverse of the PCA projection to recover the generated vectors back into the original space. Then the distances between the generated vectors and graphs in the projection set could be computed directly. However, there are two problems with this. First, since we are discarding some components from the projection to work in the reduced space, then recovering the generated vectors using this subset of components would introduce errors. Second, this results in a distance measure defined in a very high dimensionality space and when noise is present this measure will produce poor matches.

The third solution is an extension of the second one. We can compute a second PCA projection based on the input and projection sets and use this to reduce the dimensionality of the original space. The generated vectors are recovered to the original space (as in the second solution) and then projected, along with the vectors from the projection set, using the new PCA transform into a second reduced vector space. The distances can then be calculated in this second reduced vector space.

We choose the third solution to solve this problem since we expect it to introduce the least error and be the most robust to different data sets. If the projection set contains significantly different structures to the input set then the first method will not work. The second solution suffers from a distance measure defined in a high dimensional space. Although the third solution is not ideal in that it inherits the problem of the second by introducing error in the recovery of the generated vectors, we still feel that it is best suited for the task. Furthermore, as GMM estimation methods become more powerful the first PCA projection can consist of more components and therefore less error will be introduced when recovering the generated vectors.

To clarify the process we will now describe the sequence of operations required in more detail. Please refer to figure 6.5 for a diagrammatic representation of the method that will be referred to through out this description. The process is split up

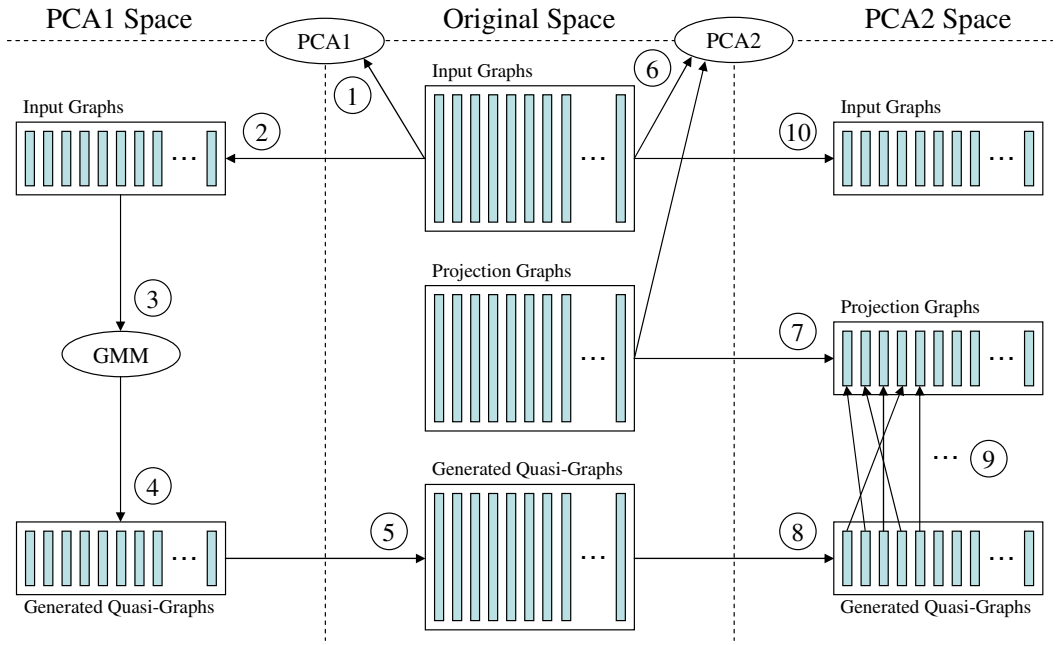


Figure 6.5: A diagram showing the process of generating new molecules.

into three stages: fitting the GMM to the input data, sampling from the GMM to create new quasi-graphs and finally mapping the quasi-graphs onto the graphs from the projection set.

Fitting the GMM to the Input Data

We commence from the aligned padded adjacency matrix $\mathring{\mathbf{S}}_k$ describing input graph $S_k \in \mathcal{S}$. We stack each adjacency matrix $\mathring{\mathbf{S}}_k$ into a long vector $\mathbf{s}_k$[4] using $\mathrm{vec}(\mathring{\mathbf{S}}_k)$ and perform PCA on this set of vectors (① in figure 6.5). This results in a projection matrix $\mathbf{\Phi}_{\mathcal{S}}$ of eigenvectors and a mean vector $\boldsymbol{\mu}_{\mathcal{S}}$. We select the top t_1 components of the projection matrix resulting in a matrix $\tilde{\mathbf{\Phi}}_{\mathcal{S}}$ which is $\mathbf{\Phi}_{\mathcal{S}}$ truncated after t_1 columns. The value chosen for t_1 is dependent on the amount of sample data available.

We can now project the vectors describing the input graphs into the space given by the first PCA projection (PCA1 - ②):

[4]Please note that prime symbols on vectors represent which vector space they reside in as shown in figure 6.5; a vector with no prime symbol ($\mathbf{a}$) means the original space, a vector with one prime ($\mathbf{a}'$) is in PCA1 space and a vector with two primes ($\mathbf{a}''$) is in PCA2 space.

$$\mathbf{s}_k = \mathrm{vec}(\mathring{\mathbf{S}}_k) \tag{6.10}$$

$$\mathbf{s}'_k = \tilde{\mathbf{\Phi}}_{\mathcal{S}}^T(\mathbf{s}_k - \boldsymbol{\mu}_{\mathcal{S}}) \tag{6.11}$$

We are now ready to fit a GMM to the data described by the set of vectors $\{\mathbf{s}'_1, \mathbf{s}'_2, ..., \mathbf{s}'_{|\mathcal{S}|}\}$. We use the algorithm proposed by Verbeek et al [109] which calculates the parameters of the GMM using a greedy EM based approach (③)[5]. The algorithm is also capable of estimating the number of components in the mixture model. The output of the algorithm is k multivariate normal distributions. A single multivariate normal distribution is defined as such:

$$p(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}\sqrt{|\boldsymbol{\Sigma}|}} \exp[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})] \tag{6.12}$$

with parameters mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. The definitions of these are given in equations 4.4 and 4.5 respectively. Using $p(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ we can define a k component normal mixture distribution:

$$p_k(\mathbf{x}) = \sum_{i=1}^{k} p(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) w_i \tag{6.13}$$

The mixing weight w_i is the probability that a sample from the GMM would be drawn from component i. As such the sum of the mixing weights must equal one:

$$\sum_{i=1}^{k} w_i = 1$$

for $i \in \{1, ..., k\} : w_i \geq 0$. Therefore, a component i of the GMM is represented by the triple $(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, w_i)$ which denote the mean, covariance and weight respectively.

In order to sample from component i of the GMM we require the set of eigenvalues and eigenvectors from covariance matrix $\boldsymbol{\Sigma}_i$, $\boldsymbol{\Lambda}_i$ and $\boldsymbol{\Phi}_i$ respectively. The eigenvectors describe the principle components of variance in the subset of vectors that make up this component of the GMM and the eigenvalues describe the variance of each principle component.

[5]The external reviewer, Dr. Peter Hall, suggested that the GMM estimation algorithm of Figueiredo & Jain [44] would produce better results.

Sampling from the GMM

To sample from the GMM with distribution $p_k(\mathbf{x})$ we must first choose which component will be sampled. This is accomplished by generating a uniform random number a in the range $(0, 1]$ and then using a function υ to map this to a component i.

$$\upsilon : a \rightarrow \begin{cases} 1 & \text{if } a \in (0, w_1] \\ i & \text{if } a \in (\sum_{j=1}^{i-1} w_j, \sum_{j=1}^{i} w_j] \end{cases} \tag{6.14}$$

To sample from component i we first generate a parameter vector $\mathbf{b}$. This is produced by sampling from a number of 1D normal distributions with zero mean and variance determined by the diagonal values in $\mathbf{\Lambda}_i$. The parameter vector is computed as such:

$$\mathbf{b}(j) \sim \mathcal{N}(0, \mathbf{\Lambda}_i(j, j)) \tag{6.15}$$

We can then project the parameter vector $\mathbf{b}$ on the components of the distribution $\mathbf{\Phi}_i$ and add the mean $\boldsymbol{\mu}_i$ to compute a vectorial representation of a new quasi-graph $\mathbf{g}'_k$ (④).

$$\mathbf{g}'_k = \boldsymbol{\mu}_i + \mathbf{\Phi}_i \mathbf{b} \tag{6.16}$$

Mapping the Samples to the Projection Set

To map a generated quasi-graphs to the graph from the projection set which is closest we require representations of both graphs to reside in the same vector space. As discussed in the beginning of this section we accomplish this by using the inverse PCA transform to recover representations of the quasi-generated graphs to the original vector space. A new PCA projection is computed from the input and projection sets of graphs thus capturing the structural variations contained in the two sets[6]. The recovered representations of the quasi-generated graphs are projected into this new space and distances between graphs may be directly computed. This allows us to

[6]Although technically we only need to compute the PCA transform from the vectors of the projection set, we include the input set to enhance the quality of the projection. After all, the generated quasi-graphs are sampled from the input set so this helps by including information on the variations in the generated graphs.

compute the mappings from generated quasi-graphs to graphs in the projection set.

We begin this phase by recovering the generated quasi-graphs using the inverse PCA transform (⑤):

$$\mathbf{g}_k = \tilde{\mathbf{\Phi}}_{\mathcal{S}} \mathbf{g}'_k + \boldsymbol{\mu}_{\mathcal{S}} \tag{6.17}$$

We now compute a new PCA space that is more suitable for performing distance calculations than the original space. To construct this new space we compute a PCA transform on the vectors describing the graphs in the input and projection sets (PCA2 - ⑥). The vectors of the input set are computed as described in equation 6.10 and the projection graphs are vectorized as follows:

$$\mathbf{p}_k = \mathrm{vec}(\mathring{\mathbf{P}}_k) \tag{6.18}$$

The result of the new PCA transform is a mean vector $\boldsymbol{\mu}_{\mathcal{S}\cup\mathcal{P}}$ and a projection matrix $\mathbf{\Phi}_{\mathcal{S}\cup\mathcal{P}}$. We select the t_2 largest components of variance in the projection and truncate the projection matrix to contain only this information; $\tilde{\mathbf{\Phi}}_{\mathcal{S}\cup\mathcal{P}}$ is the result of truncating $\mathbf{\Phi}_{\mathcal{S}\cup\mathcal{P}}$ after t_2 columns. We can now project both the projection graphs and the generated quasi-graphs into this new space as follows (⑦ and ⑧).

$$\begin{aligned} \mathbf{p}''_k &= \tilde{\mathbf{\Phi}}^T_{\mathcal{S}\cup\mathcal{P}}(\mathbf{p}_k - \boldsymbol{\mu}_{\mathcal{S}\cup\mathcal{P}}) \\ \mathbf{g}''_k &= \tilde{\mathbf{\Phi}}^T_{\mathcal{S}\cup\mathcal{P}}(\mathbf{g}_k - \boldsymbol{\mu}_{\mathcal{S}\cup\mathcal{P}}) \end{aligned}$$

We are now in a position to compute the nearest projection graph to each quasi-generated graph. We define a function ρ that maps a generated quasi-graph to the graph from the projection set at minimum Euclidean distance.

$$\rho : \mathbf{g}''_k \rightarrow P_j \text{ where } j = \underset{i}{\mathrm{argmin}}\ \mathrm{dist}(\mathbf{p}''_i, \mathbf{g}''_k) \tag{6.19}$$

Therefore, the final multiset of generated molecules is given by $G_k = \rho(\mathbf{g}''_k) \in \mathcal{G}$. A multiset is used to allow repetitions of projection molecules since it is likely that repeated molecules will indicate statistical interesting molecules to explore.

Finally, we note that to aid visualizing all three sets, we can also project the input graphs into PCA2 space (⑩).

$$\mathbf{s}''_k = \tilde{\Phi}^T_{\mathcal{S}\cup\mathcal{P}}(\mathbf{s}_k - \boldsymbol{\mu}_{\mathcal{S}\cup\mathcal{P}}) \tag{6.20}$$

6.3 Experimental Results

We make use of a number of experimental methods to evaluate our process of generating chemical structure. We will begin with a visual inspection of the generated molecules and see how they relate in structure to the input set of molecules. Next we visualize the various distributions constructed throughout the process by using PCA. Although this is a significant simplification of a very high dimensional data set, it is nevertheless very useful for comparing distributions. We can also visualize mappings between sets of graphs by overlaying them on the PCA plot. This will be useful for visualizing the mappings between the generated quasi-graphs and the projection graphs. The mappings can also be visualized using histograms. Using these results we will assess the performance of the hierarchial alignment step, the applicability of the projection molecules as targets for the mapping of the generated quasi-molecules, the suitability of fitting a Gaussian mixture model and the quality of the final generated molecules. We can further assess the quality of the generated molecules by applying them to our evaluation domain which is the task of docking molecules to active sites in proteins.

By using a set of molecules that dock with high affinity to the selected active site as input to our method, we hope to generate molecules that also dock with high affinity. Although we do not expect to replicate all of the pharmacological properties that the molecules in the input set exhibit, we do expect that some of the generated molecules will perform well in the docking test. As discussed in the introduction, this is because the docking process is primarily a test of structure and we are generating structurally similar molecules to those in the input set.

This section begins with a brief description of the evaluation domain. For a full description of docking we refer the reader to section 2.2.4 of the literature review. A discussion of the programs available for performing docking is also given in the literature review, although we will describe the program we use here in detail. After an overview of docking we will evaluate data sets suitable for use with our method. Then, in the final two sections, we give detailed results for the selected data sets.

6.3.1 Evaluation Domain

Docking is a subfield of molecular modeling with the aim of finding the best possible pose of a ligand in the active site of a receptor. To do this the docking program must calculate a set of features that determine whether the ligand and the active site are complementary. For example, the molecular surface of the ligand and the surface of the active site are considered. When the ligand is bound to the receptor they form a complex, and the affinity with which the two molecules bind may be measured and described using a scoring function.

The problem is complicated by the fact that most ligands are steroisomers. Molecules which are steroisomers may exist in a number of different 3D structures, and most small, drug-like molecules have this property. Therefore, the docking program must consider all possible poses of all possible conformations of a ligand to truly explore the variety of complexes that may be formed.

Docking with Fred

Fred (Fast Rigid Exhaustive Docking) is a tool from the OEChem Toolkit which provides robust, industry-standard docking between ligands and proteins that can be performed in an automated manner. It exhaustively examines all possible poses of all conformations of a ligand using geometric features such as shape complementarity and features based on pharmacological properties. For the optimal docking of each ligand it computes a score based on one of its in-built scoring functions. The average time for docking a ligand is in the region of a few seconds making it suitable for performing virtual screening on large databases of molecules.

There are a few features of Fred that make it an especially good choice for a docking program.

- Fred has a "very effective method for determining the shape of an active site (that works well even on very shallow/open binding sites)"[3]. This is useful to us since although we know the approximate region an active site occupies, it is still up to Fred to find the exact active site and calculate its shape.

- Fred uses a multiple consensus scoring method. This provides protection against any bias a scoring function may give to a particular active site or ligand.

- Fred uses an exhaustive, non-stochastic method for determining poses. This property is important for producing consistent, repeatable results.

- Fred provides a utility, FredTool, that can be used to setup an OEChem specific file that describes the active site. This tool allows generation of the active site either by hand or from an already docked ligand.

Using FredTool on a pre-docked ligand produces two volumes describing an inner and an outer contour. See figure 6.7 for a 3D image of the protein and the inner and outer contours describing the active site. The inner contour must contain a significant portion of the docked ligand while the position of the ligand in the outer contour is less significant. However, any successful docking must not exceed the boarder of the outer contour. We make use of the *chemgauss3* scoring function to score ligand-protein complexes.

However, by itself Fred is unsuitable for docking the molecules we produce. This is because we only have 2D descriptions of the molecules after generation and we clearly require 3D descriptions to proceed with docking. To compute the full range of 3D conformations for each molecule we generate we make use of the program *Omega* from the OEChem Toolkit. Omega can generate a full set of conformations for a drug-like molecule in a couple of seconds.

Data Sets

Given that we would like to evaluate our generated molecules in a real-world docking scenario, we require the following properties from any data set we choose.

- A protein that will function as the receptor.

- A bounding box indicating the active site on the protein. Alternatively, an already docked ligand can provide the bounding box information.

- A reasonably large selection of ligands that are all known to bind with high affinity for the active site.

To create the type of data set described above we could use the following approach. We must first choose a protein with a function that is well studied in the literature and, as a result, has plenty of information regarding the docking affinity of

a number of ligands. The structure of the protein can be obtained from the Protein Data Bank (PDB) [9] however the active site must be identified by hand from the literature. Next, a set of ligands that are reported to bind with a high affinity for the target must be collected. This step involves searching the literature to assemble the set or using the results of real-world high-throughput screening against the active site of the receptor. A file describing each molecule can be downloaded from the ZINC database [10] which is free and also allows sets of molecules to be retrieved with ease.

The process above can be considerably simplified by making use of a number of publicly available data sets. Generally these data sets are constructed with the idea of testing docking programs in mind. Leach et al [70] give a review paper describing the state of the art of docking technology including data sets available. The first type of data sets are those that include a docked ligand with the protein file thus facilitating setting the volume the active site occupies. PDBbind [111] is one such data set. It contains a very large number of protein-ligand complexes however, only one ligand per protein is given therefore making collection of the remainder of the data set quite difficult. Another data set, Binding MOAD [54], has the same problem.

The second type of data set available fulfills all the requirements we listed above. The Directory of Useful Decoys (DUD) [55] describes a total of 2950 active ligands across a total of 40 target receptors. Not only does this suit our purposes perfectly but they also include, for each active molecule, 36 decoy molecules with similar physical properties but dissimilar structure. If we require any tests against a random set of molecules we can use the set of decoys instead. If we still obtain good results then we have a stronger result with the decoys than with a random selection since they are specifically chosen to complicate matters.

From the DUD data set we select two subsets, each consisting of a protein, a docked ligand and a set of active ligands. The two subsets we selected contain data on the *Cyclooxygenase-2* (COX2) protein and the *epidermal growth factor receptor* (EGFR) protein. Pharmacological inhibition of the COX protein can suppress pain and inflammation and is how most non-steroid based anti-inflammatory drugs function. Mutations involving the EGFR protein can result in it being constantly active which can lead to certain types of cancer. Therefore, methods of inhibiting

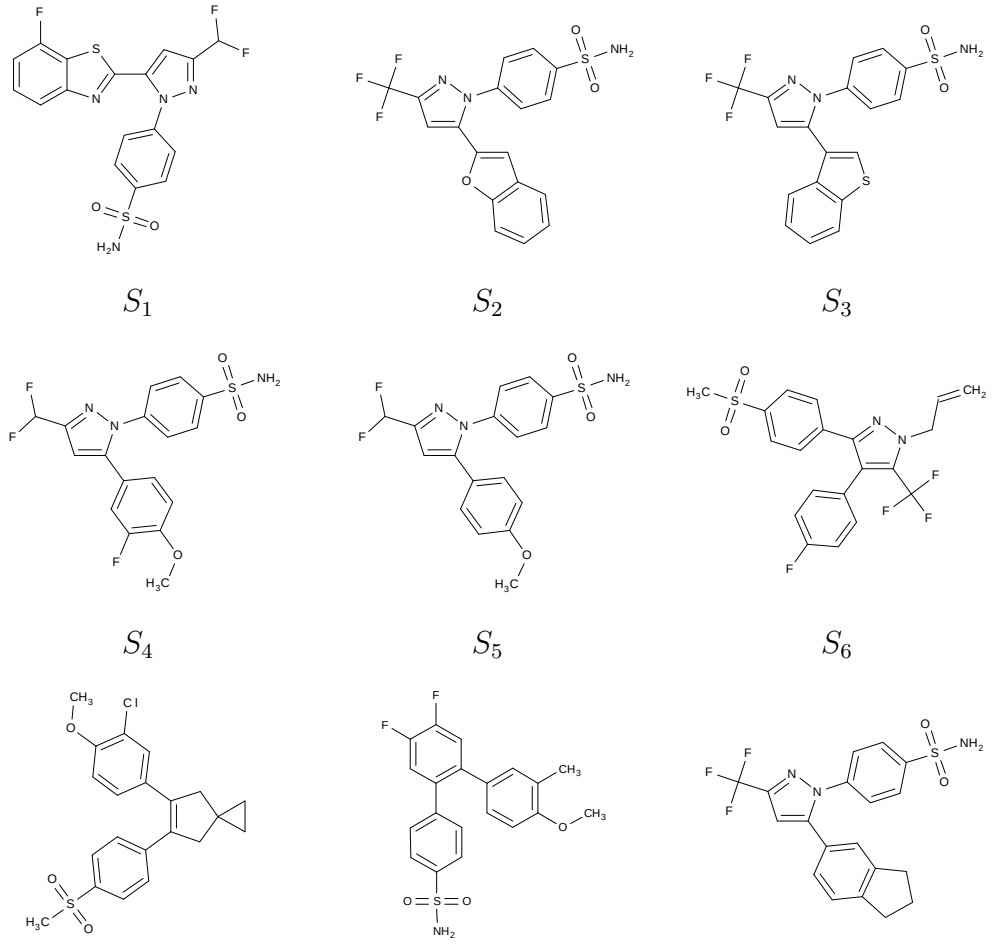
the function of the EGFR protein can be used to regulate its activation.

We form our data sets as follows. From the set of active ligands we remove any duplicates that may be present (the duplicates would manifest themselves as conformers of another active ligand and, since we are not using any 3D information in our process, they are superfluous). We then use FredTool to setup the active site using the docked ligand so there is no need to compute the active site by hand. Next, we preprocess the ligands to remove all hydrogen atoms which results in the model discussed in section 6.2.2 (the hydrogen atoms become implicit). We then perform a preliminary docking of the set of active ligands and select the top performing ligands. For the COX2 data set we select the 39 top scoring ligands and for the EGFR data set we select the top 110 ligands. These sets of ligands will form the input to our model.

6.3.2 Results from the COX2 Data Set

We begin our experimental results from the COX2 data set by giving examples of the molecules contained in it. Figure 6.6 shows 9 different molecules from the data set. Molecule S_1 has the highest binding affinity for the COX2 active site, molecule S_2 has second highest affinity, etc. Although it is not immediately obvious from the 2D depictions they all have a reasonably similar molecular surface. Figure 6.7 shows the active site of interest for the COX2 protein and figure 6.8 shows the molecule from the input set with highest affinity docked in the active site. It is for this site that we will aim to generate molecules that bind with high affinity. The parameters for the results in this section are as follows: the projection set consists of 500 molecules and we sample 200 vectors from the GMM representing the generated quasi-molecules.

Figure 6.9 shows the top 6 scoring molecules we have generated from this data set. Scoring in this case is a measure of a binding affinity computed by the docking procedure described in the previous section. Note the similarities of structure observed between the input molecules shown in figure 6.6 and the molecules shown in this figure. All the molecules contain a triangle shaped structure that is orientated to fit in the active site as in figure 6.8. This is clearly seen in graphs $G_1, ..., G_5$. In graph G_6 this triangle is more stretched. Also note that the molecules are correct according to our simple valence model described in section 6.2.2.



S_1 S_2 S_3

S_4 S_5 S_6

S_7 S_8 S_9

Figure 6.6: Nine molecules from the COX2 data set. Molecule S_1 has the highest binding affinity for the COX2 active site, molecule S_2 has second highest affinity, etc.

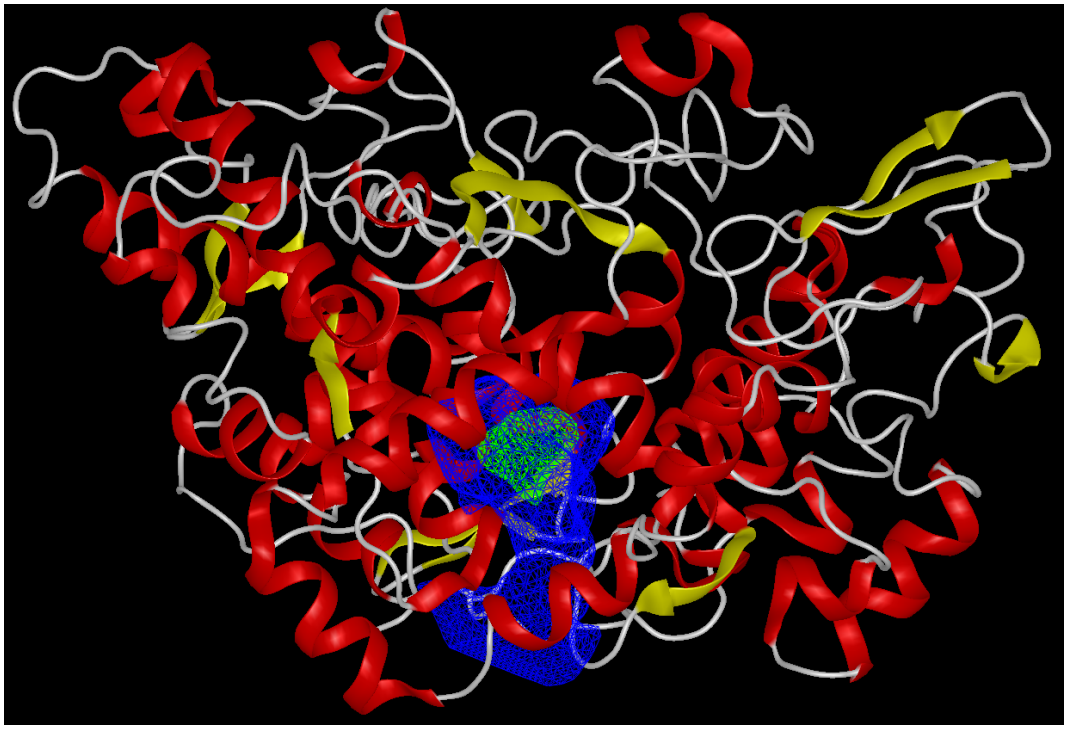


Figure 6.7: The active site of interest in the COX2 protein. The outer contour is shown as a blue mesh and the inner contour is shown as a green mesh.

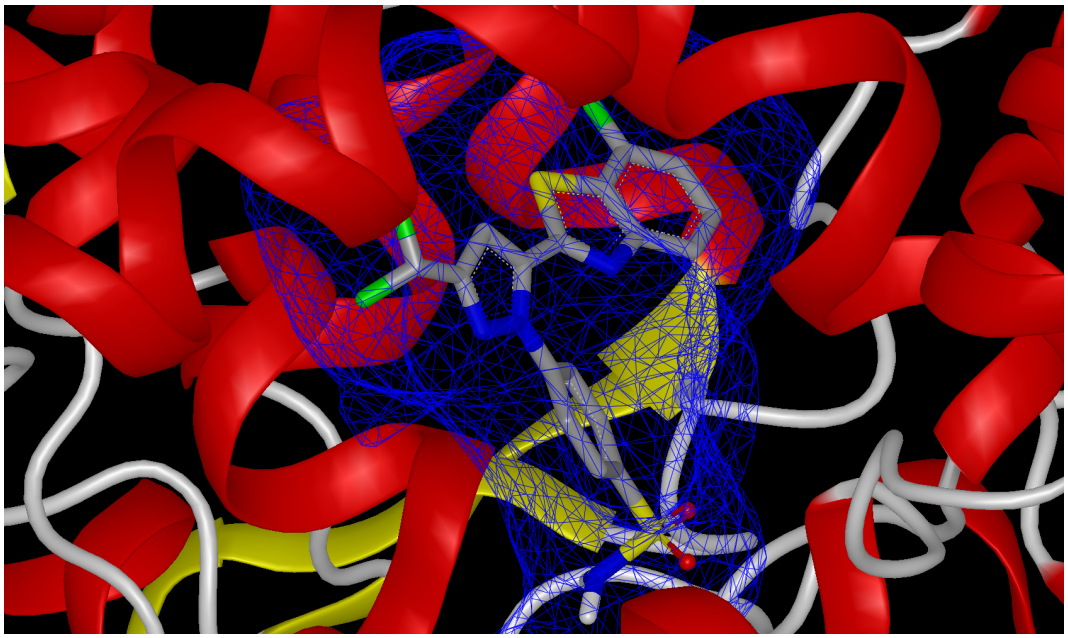


Figure 6.8: The active site of interest in the COX2 protein. Molecule S_1 from figure 6.6 (the molecule from the input set with highest binding affinity) is shown in its optimal pose as computed by Fred.

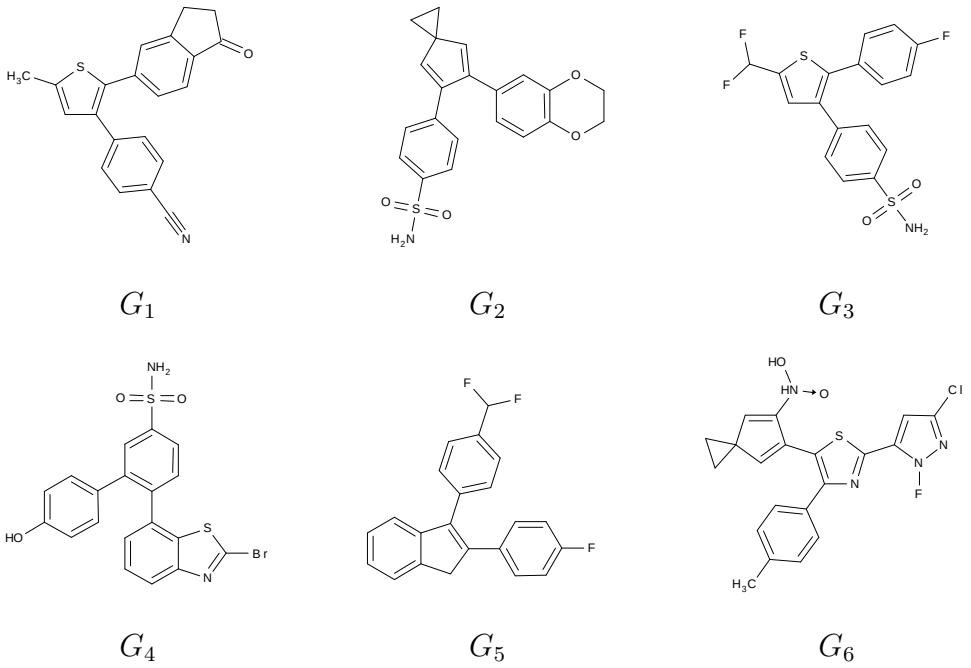


Figure 6.9: Six molecules we have generated using our approach from the COX2 data set. Molecule G_1 has the highest binding affinity from the generated set, G_2 the second highest binding affinity, etc.

We now show PCA plots allowing us to view the various distributions constructed throughout the process. We will begin from the PCA projection shown in figure 6.10. This shows the first PCA space that is calculated in the generation procedure i.e. the space the GMM is constructed in and labeled PCA1 in figure 6.5. This projection shows the input molecules as red plus signs and the generated vectors from the GMM as crosses. For each component computed by the GMM we draw ellipses showing the axis of principle variance, one eclipse at the standard deviation and a second at two times the standard deviation. The means are marked in the center of the ellipses. In this case the GMM estimation algorithm has chosen to fit two normal distributions to the data.

For this data set we have set $t_1 = 7$ which means we use the seven largest components of the projection to transform data. Despite the fact that we are only visualizing two of these seven dimensions the distributions that have been fitted look reasonable. Vectors have been sampled spanning the whole space of the input molecules and in areas where the input molecules are clustered so are the generated vectors.

The remainder of the PCA projections shown in this section are of the second

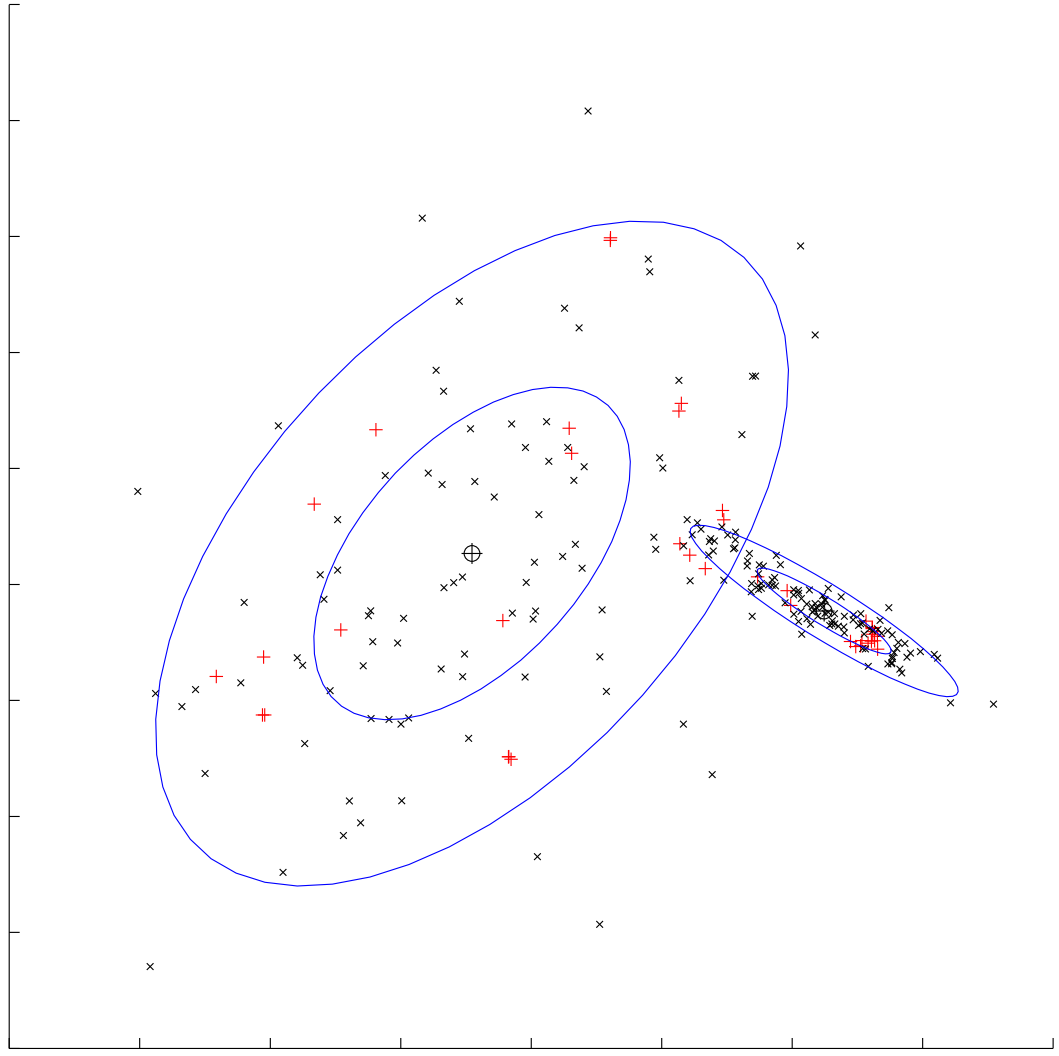


Figure 6.10: A PCA projection of the input set (projected into PCA1 space). The input set is marked with plus signs and the generated quasi-molecules are marked with crosses. Also shown are ellipses indicating the axis of principle variance of the normal distributions determined by the GMM algorithm.

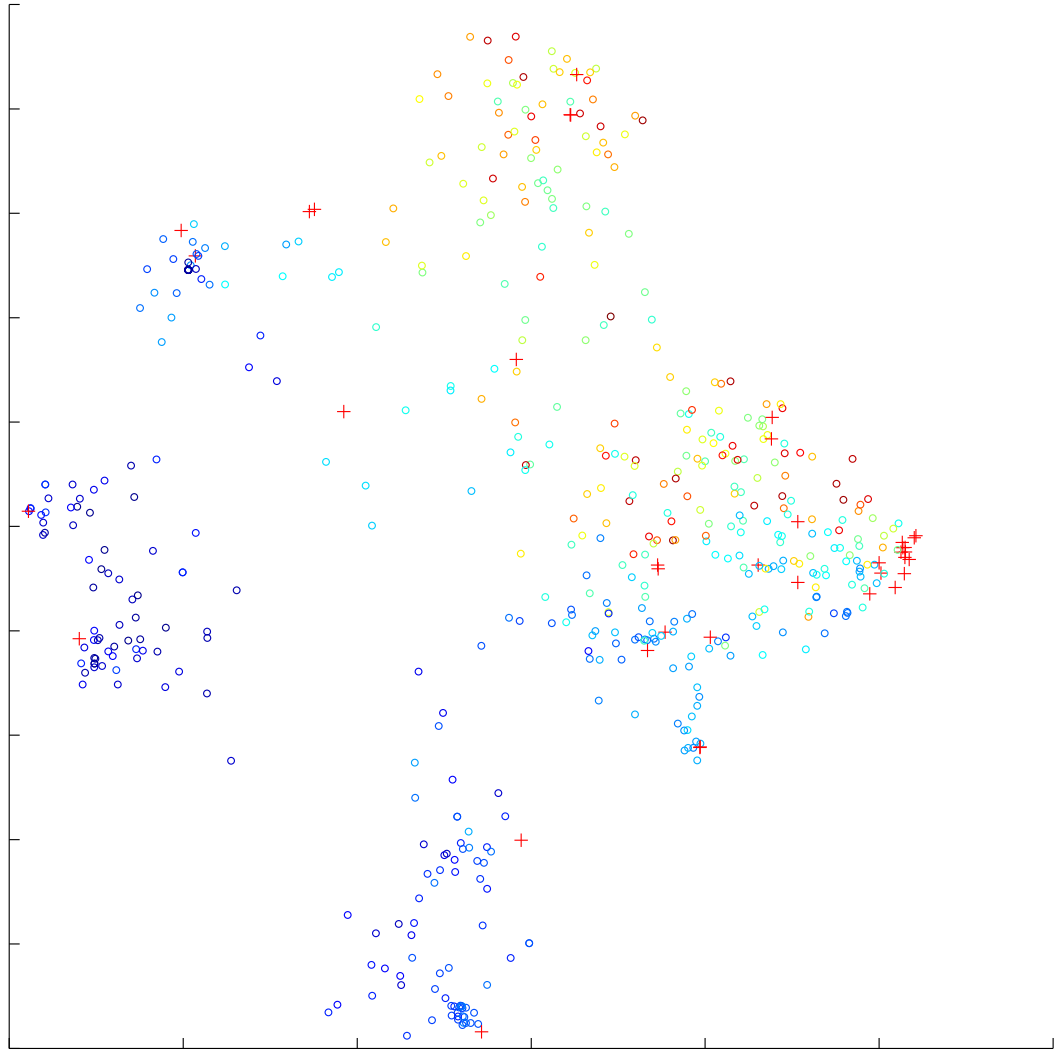


Figure 6.11: A PCA projection of the input set (marked with plus signs) and the projection set (marked with circles). The weight of each projection graph is given by the colour of the circle representing it, blue indicates low weight and red indicates high weight

PCA transform (labeled PCA2 space in figure 6.5). The parameter t_2 was set to seven again so we use the seven largest components of the projection to transform data. We begin by showing just the molecules from the input and projection sets (figure 6.11). This projection shows the input molecules as red crosses and projection molecules as colored circles. The color of each circle indicates the molecular weight of the projection molecule, with blue being molecules of low molecular weight and red indicating molecules of high molecular weight. The molecular weight of molecules is approximately increasing in one direction over the plot which is expected as molecules with similar weight will align well to each other (see section 6.2.4 on hierarchial alignment). The hierarchial alignments clearly have a large impact on the distances between molecules in the resulting vector space and this can be observed in the PCA plot. Notice the large cluster of input graphs in the middle right of the plot. This has arisen since the distances between alignments of these molecules are all relatively low values. The input graphs contained in the right middle cluster are the majority of the graphs contained in the subtree at depth 2 of graph 39 in figure 6.4.

We can now begin to evaluate one of the main goals of this research: are the set of projection molecules close enough to the input distribution to effectively allow generated graphs to be mapped to them? In other words do we have diversity in the projection graphs and are these diverse regions near clusters of input graphs? By a visual inspection of the PCA plot in figure 6.11 we can see that this requirement has been suitably fulfilled. Although there is a slight lack of projection graphs near the middle right cluster, the remainder of the space is well covered by projection molecules. Another important aspect is that the projection molecules we constructed in the process are different to the input molecules, i.e. we are not just duplicating the input set. To verify this further we used the distances computed by the full graph alignment algorithm in the hierarchical alignment process to check whether any of the projection graphs had zero distance to the input graphs. We found no distances of zero, meaning that no projection graphs appear in the input set.

We can assess the effectiveness of the projection set further by considering our alignment step. Ideally we would like approximately the same number of projection graphs to align to each input graph which would indicate that we have constructed

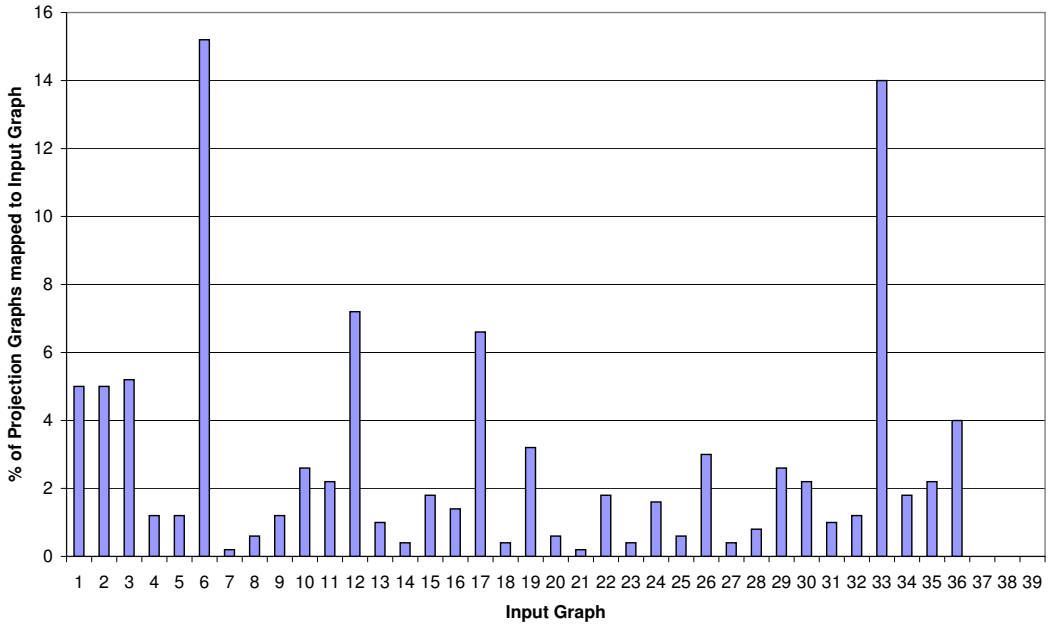


Figure 6.12: A histogram showing the percentage of projection graphs that align to a specific input graph in the hierarchical alignment step.

projection graphs that are close in distance to the whole range of input graphs. The histogram in figure 6.12 shows the percentage of projection graphs that map to an input graph. Obviously this method is open to error in the following situation: if two input graphs are very near each other then the majority of projection graphs in that area might map to only one input graph. This would indicate the other input graph is not well represented in the projection set, which would be an incorrect conclusion. However, this is a minor issue and the results from the histogram show that mappings are relatively equal across the input graphs.

The exceptions are the large discrepancies of input graph 6 and 33 which can be explained by the projection graphs that are significantly smaller and significantly larger than those in the input distribution. Since input graph 6 is the largest of the input distribution we would expect that all projection graphs that are larger than graph 6 would be aligned to it. The same applies to graph 33 which is one of the smallest in the input set.

We now consider the generated graphs both in the distribution they form and the mappings from the quasi-generated graphs to the projection graphs. We begin by introducing a PCA plot of the generated quasi-graphs projected into PCA2 space (the space constructed from the input and projection sets). This is shown in figure

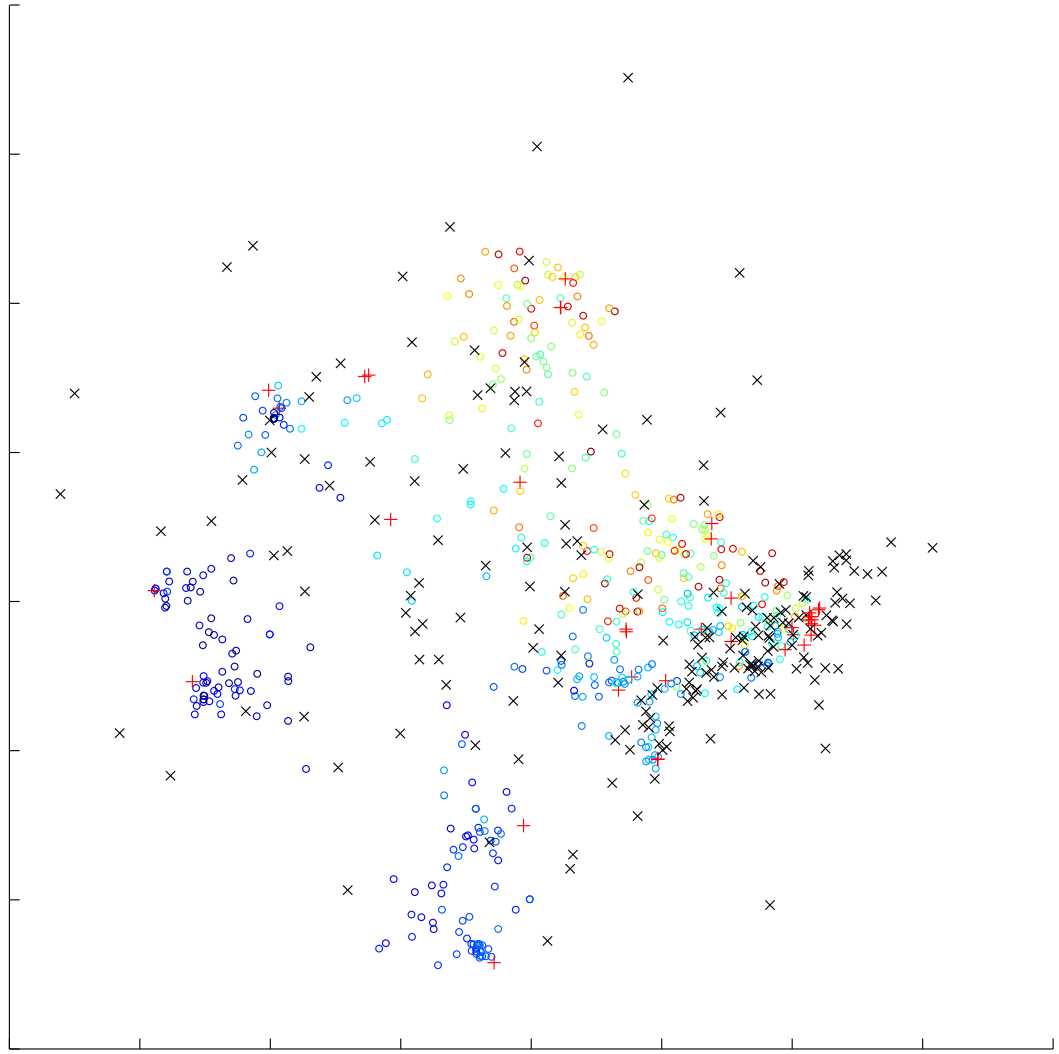


Figure 6.13: A PCA projection of the input set (marked with plus signs), the projection set (marked with circles) and the quasi-generated set (marked with crosses).

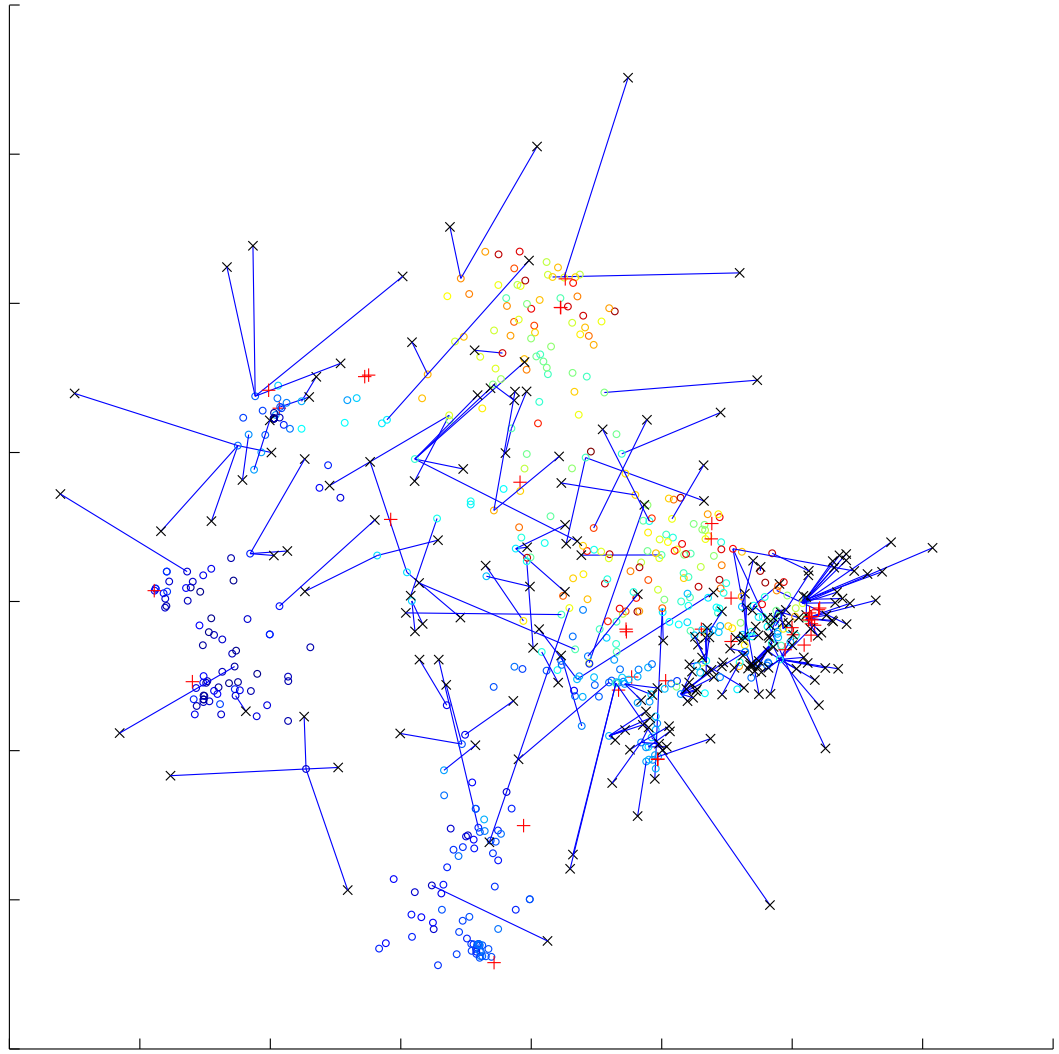


Figure 6.14: A PCA projection with the same data as figure 6.13. Matches between generated quasi-molecules and molecules from the projection set are shown.

6.13 where the markers are as before except crosses have been introduced to indicate generated quasi-graphs. In figure 6.14 we indicate the mappings from generated quasi-graphs to projection graphs by drawing a line between their two markers on the PCA plot.

We will now assess whether the generated quasi-molecules lie in the correct parts of the space after their recovery into the original space and then the re-projection into PCA2 space (see figure 6.5). Although we can only consider the first two components of the distribution in figure 6.13 we can see that the generated quasi-molecules still cover the distribution of input graphs well, especially near the cluster on the right side of the plot. The remainder of the space is sparsely populated with input graphs and correspondingly the generated quasi-molecules also sparsely cover this space. The mixture of normal distributions seems an appropriate way to model this data.

We will now assess the quality of the mappings. We would expect a high quality mapping to map each quasi-generated molecule to a projection molecule a short distance away. Furthermore, we would not expect many quasi-generated molecules to map to the same projection molecules. This would indicate a lack of diversity of projection molecules in a space where there was a very high probability of producing samples from the input distribution. We can observe from the PCA plot with mappings overlayed (figure 6.14) that where the samples from the GMM are appropriate, our mappings also seem to be appropriate. However, when outliers from the input distribution are mapped to a projection molecule the mapping is significantly worse. However, this only occurs in a 20-30 out of the 200 generated quasi-molecules.

In some areas there are not enough projection molecules and we end up with many generated quasi-molecules mapping to a single projection molecule. This occurs in the cluster of input graphs on the right of the plot. As stated before this indicates that there is not enough diversity in this area of chemical space. However, this is not a significant problem as it can be resolved by populating the projection set with more molecules. Due to the relatively fast hierarchical alignment step this is not a very costly procedure.

Now we have identified the set $\mathcal{G}$ which consists of the unique subset of projection molecules that the quasi-generated molecules map to, we can visualize the

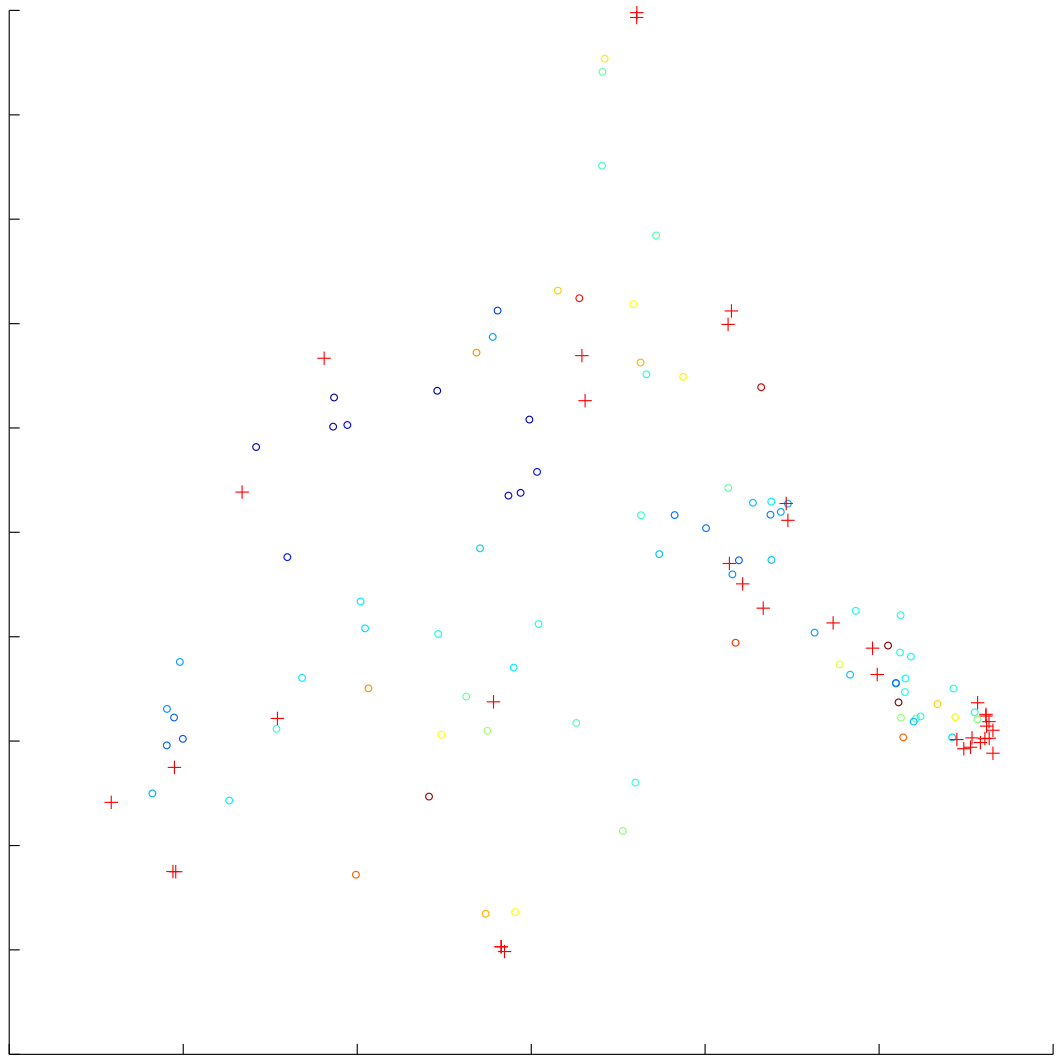


Figure 6.15: A PCA projection showing the input set (plus signs) and the subset of the projection set (circles) that comprise the true generated set.

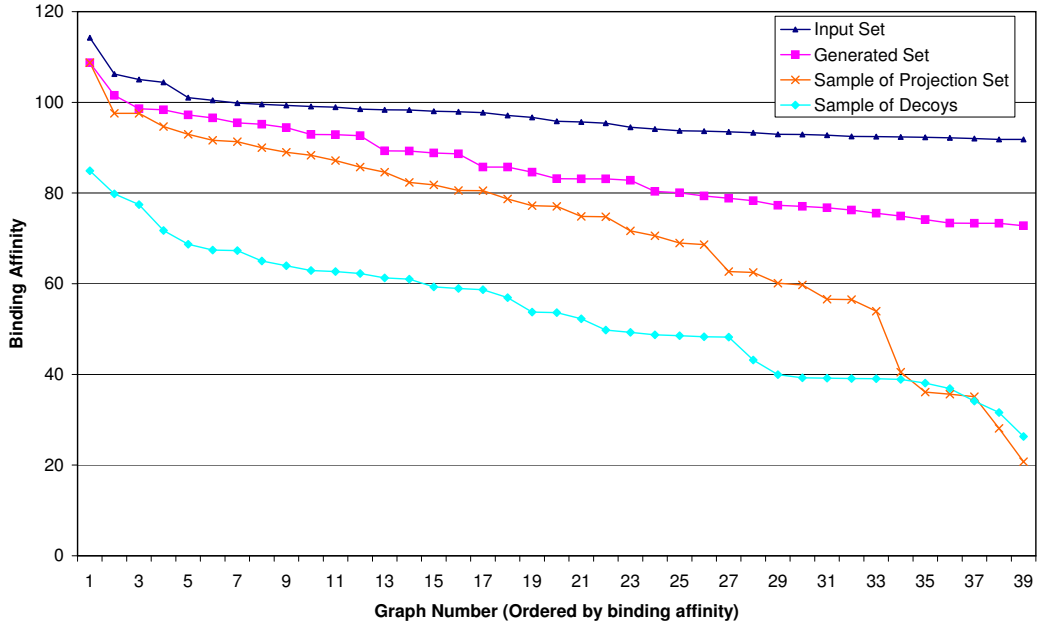


Figure 6.16: The docking scores of the molecules in the input and generated sets sorted by score. In addition a random sample of molecules from the projection and decoy sets are included for comparison.

distribution in PCA1 space again. This is shown in figure 6.15. When viewed this way we can see that our generated set is distinct to the input set and diverse in terms of structure. We have also generated graphs spanning the whole space of the input set. There is a disappointing lack of generated graphs about the right cluster of input graphs but as discussed above this could be improved by populating the projection set with more molecules.

We now describe the results of applying this set of generated graphs to the evaluation domain, that is, docking them to the identified active site on the COX2 protein. We begin by constructing all the conformers for every molecule in the generated set by using Omega. The average number of conformers for a molecule from the generated set is 50, with some molecules having as many conformers as 150. This is a good result as there is plenty of conformational space for Fred to explore to find a high scoring docking pose. On the other hand, there is not too much conformational flexibility as to render the 2D description of the molecules unimportant.

The six top scoring molecules are shown in figure 6.9 at the start of the results section. Figure 6.16 shows the docking score for four sets of 39 molecules sorted by score (we limit each set to 39 since this is the number of molecules in the input

set). Unsurprisingly, the highest scoring set is the input set. The second highest scoring set is the set of generated molecules.

To provide some comparison to the scores of the generated set we include the scores of two additional sets. The first is a random sample of 39 molecules from the set of decoys supplied with the DUD data set for the COX2 protein (see section 6.3.1). This shows, as we would expect, that choosing a random sample of chemical structures results in a set of molecules that do not dock well with the active site. Because these molecules are chosen from the decoy data set it provides a stronger result than just sampling 39 molecules at random from chemical space.

The second data set is a sample of 39 molecules from the projection set. With this set we are showing that taking the projection set by itself is not enough to produce a set of molecules that dock with high affinity. In other words, the subset of the projection set that is produced using our method is vastly superior.

In total we generated 88 unique molecules. The highest scoring molecules in the generated set are quite good however, none approach the best scoring molecules in the input set. Despite that the top ten generated molecules are all very successful at docking and this drops off slowly when we consider the remainder of the molecules. All molecules with a score over 80 give a reasonable docking.

In this section we have shown that we can successfully generate a set of molecules that dock with high affinity without having to examine hundreds of thousands of molecules. We conclude this section by showing the docking for the highest scoring generated molecule (graph G_1 in figure 6.9). This is given in figure 6.17. Please refer back to figure 6.8 to see that the generated molecule docks in a similar way to the top scoring input molecule.

6.3.3 Results from the EGFR Data Set

We will present the results from the EGFR data set in the same format as those from the COX2 data set and therefore description of the mechanisms of providing results will be terse in this section. Figure 6.18 shows 9 different molecules from the input set. Molecule S_1 has the highest binding affinity for the EGFR active site, molecule S_2 has second highest affinity, etc. The majority of the molecules in the figure have the same type of structure. That is, a large center functional group with two functional groups extending from either side of the center. Figure 6.19

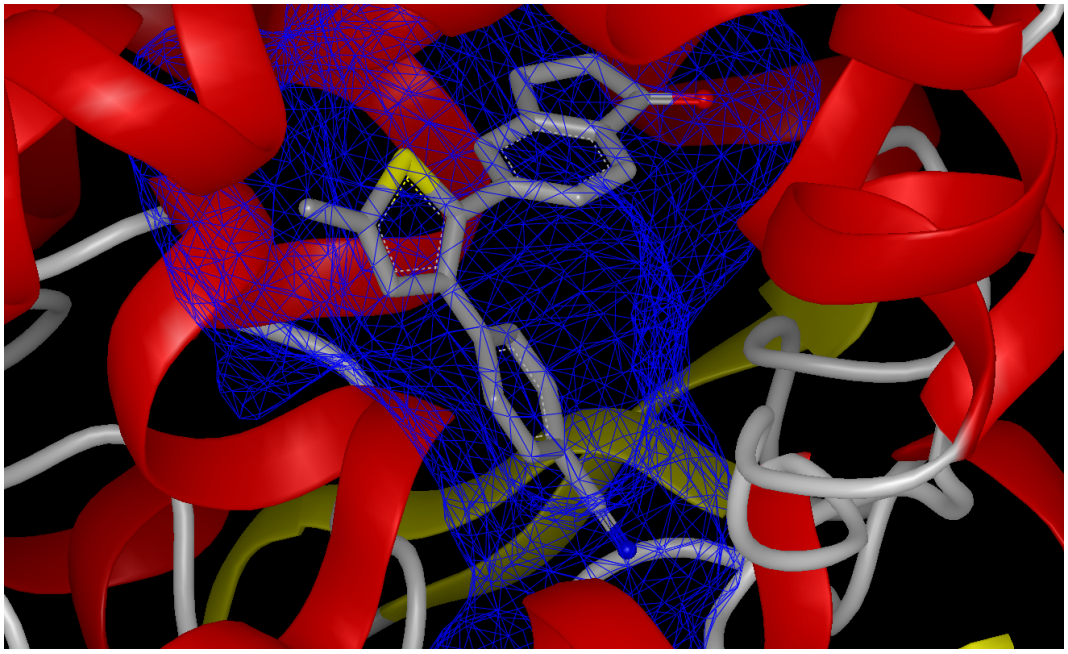


Figure 6.17: The active site of interest in the COX2 protein. Molecule G_1 from figure 6.9 (the molecule from the generated set with highest binding affinity) is shown in its optimal pose as computed by Fred.

shows the active site of interest for the EGFR protein and figure 6.21 shows the molecule from the input set with highest affinity docked in the active site. Figure 6.20 shows an alternate view of the active site. This view shows that this active site differs considerably from the active site on the COX2 protein. This is unsurprising of course, however it's more general shape will allow a larger set of molecules to dock successfully than the more specific shape of the COX2 active site. We will examine this issue more closely when we discuss the docking results.

The parameters for the results in this section are double those for the COX2 data set. This is due to having significantly more input data available - 39 input molecules for the COX2 data set versus 110 for the EGFR data set. The projection set consists of 1000 graphs and we sample 400 vectors from the GMM representing the generated quasi-molecules.

Figure 6.22 shows the top 6 scoring molecules we have generated from this data set. Some of these molecules are very similar to those in the top scoring set of input molecules (figure 6.18). Specifically G_1, G_3, G_4 and G_6 have the same bulky center structure with two smaller functional groups attached. Note that G_6 has been drawn differently due to the optimization based drawing algorithm. Molecule G_2, with

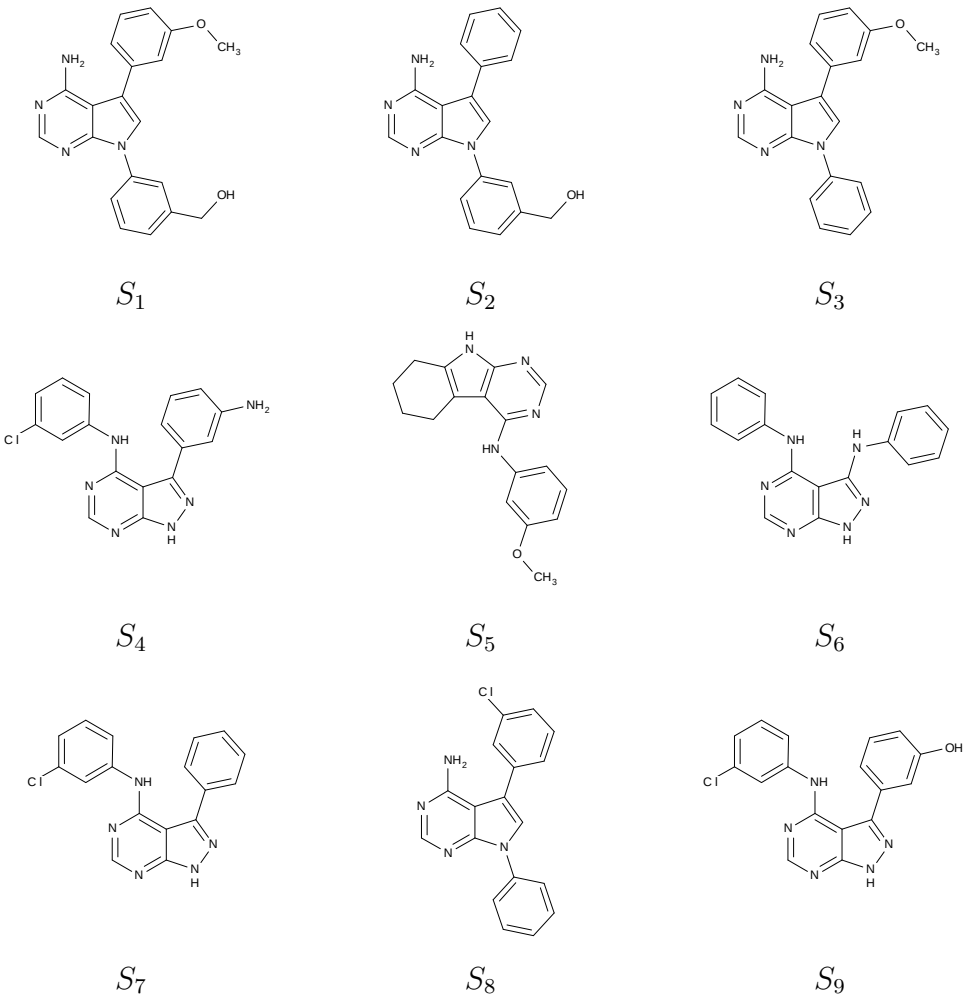


Figure 6.18: Nine molecules from the EGFR data set. Molecule S_1 has the highest binding affinity for the EGFR active site, molecule S_2 has second highest affinity, etc.

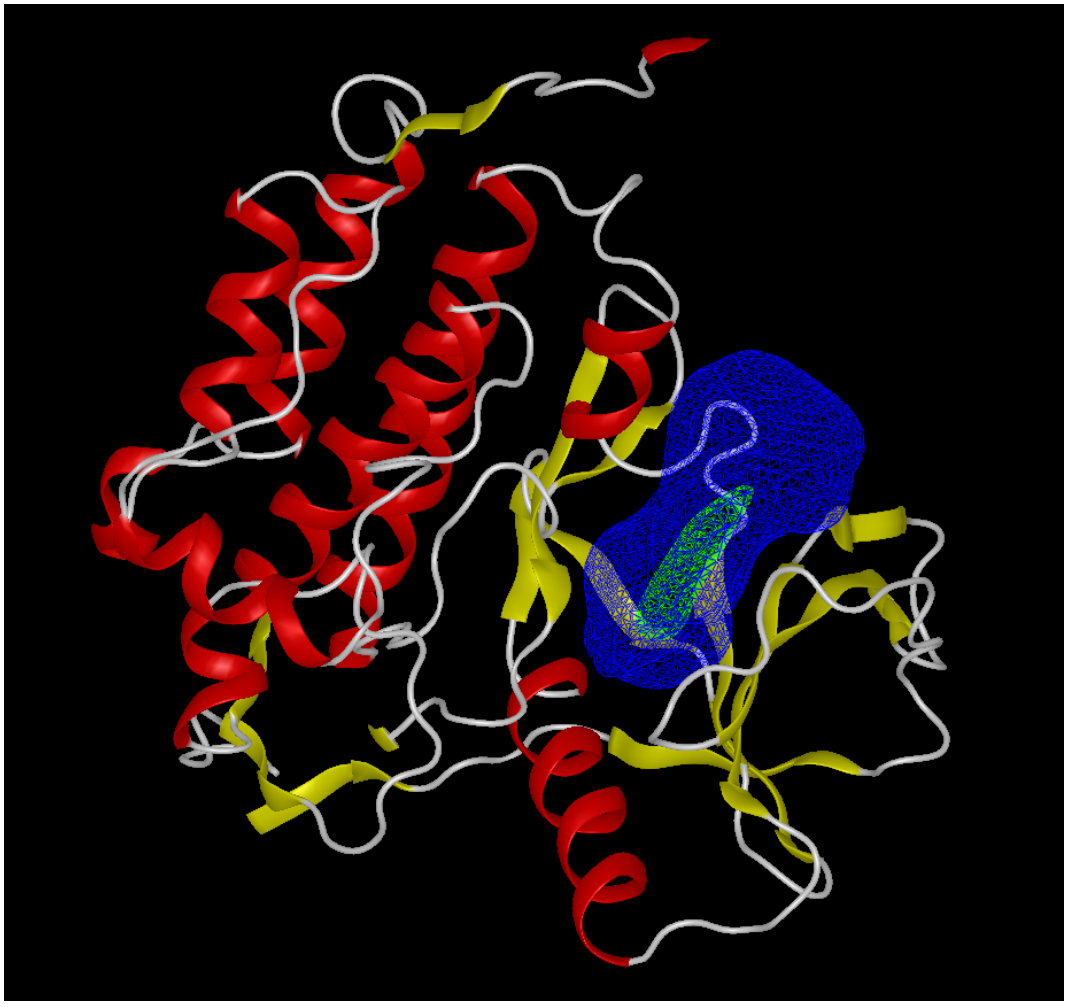


Figure 6.19: The active site of interest in the EGFR protein. The outer contour is shown as a blue mesh and the inner contour is shown as a green mesh.

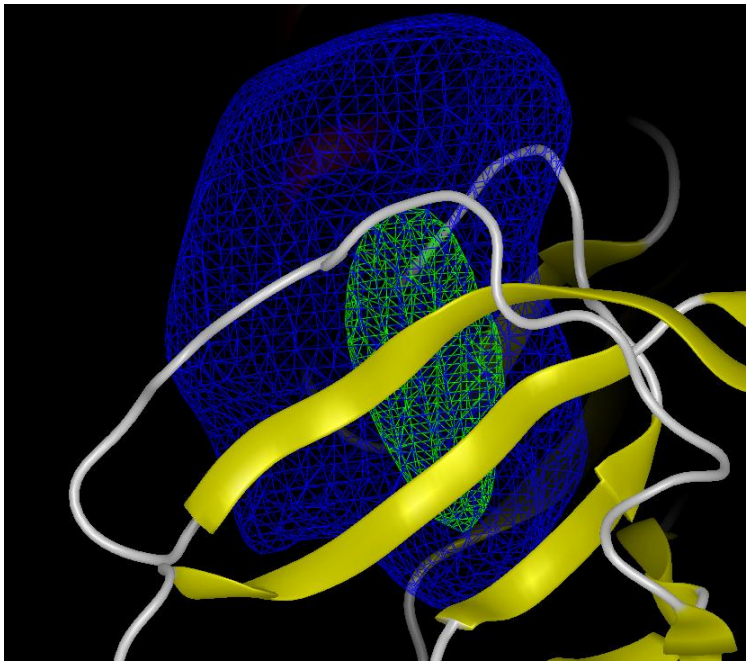


Figure 6.20: An alternate view of the active site of interest in the EGFR protein.

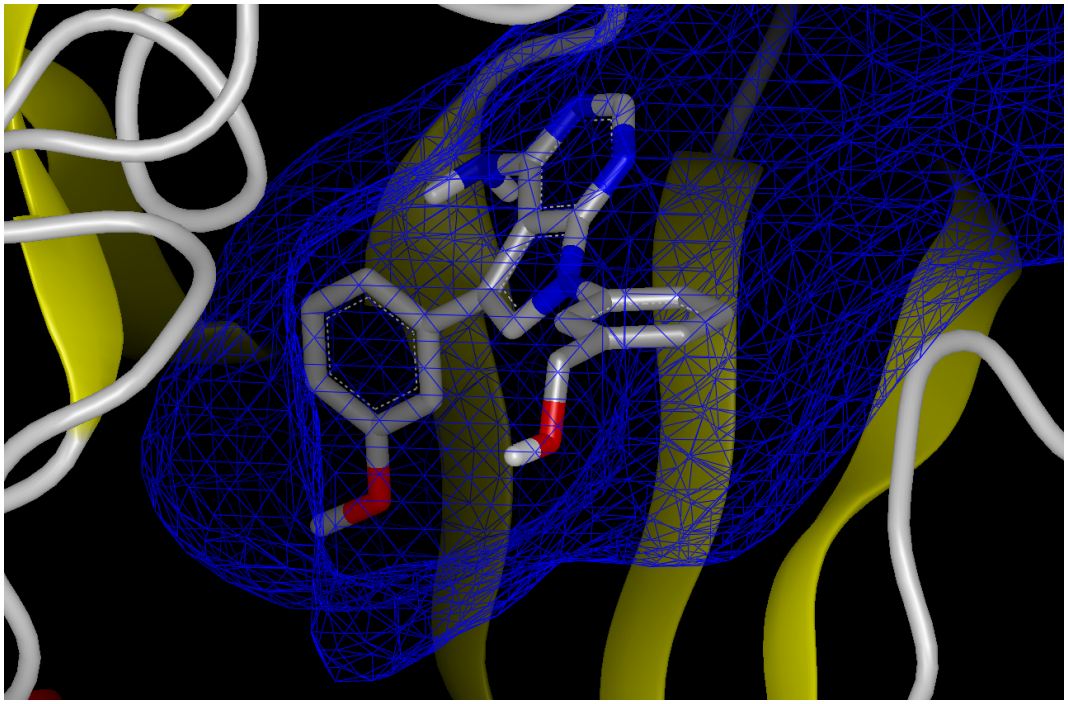
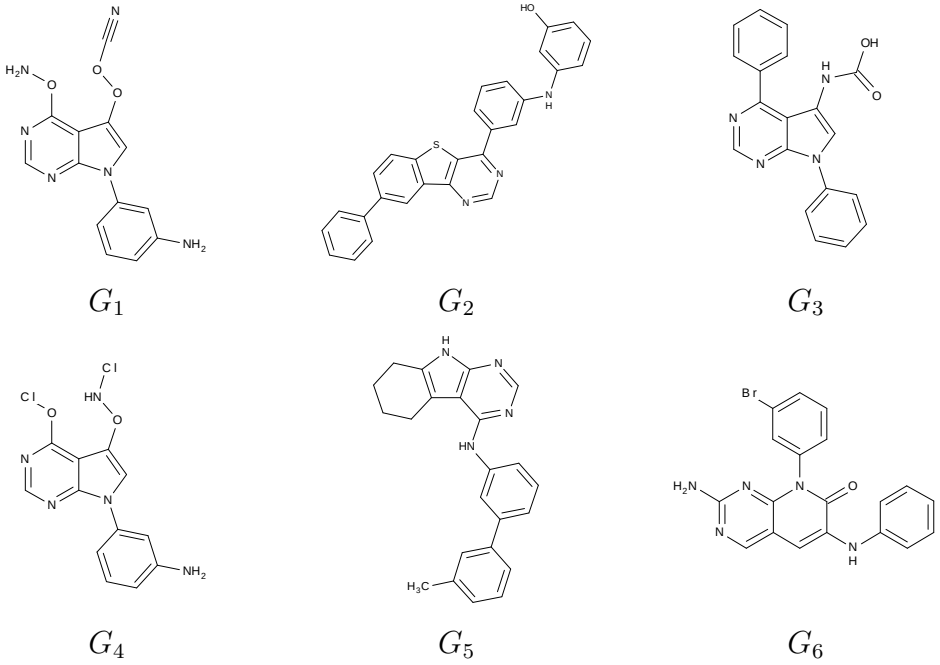


Figure 6.21: The active site of interest in the EGFR protein. Molecule S_1 from figure 6.18 (the molecule from the input set with highest binding affinity) is shown in its optimal pose as computed by Fred.



G_1 G_2 G_3

G_4 G_5 G_6

Figure 6.22: Six molecules we have generated using our approach from the EGFR data set. Molecule G_1 has the highest binding affinity from the generated set, G_2 the second highest binding affinity, etc.

its long but bulky in the middle shape, fits the active site very well. However, it is probably a poor choice for a drug given its large mass. Finally, G_5 is also a good fit for the active site but is only vaguely similar to those in the input set.

We will now show PCA plots beginning with figure 6.23. This figure shows the input molecules and generated quasi-molecules in PCA1 space. Also shown are the components of the GMM. We have chosen to set $t_1 = 12$ for this data set which results in increased accuracy of the recovered generated quasi-molecules. Furthermore, as we have significantly more input data in this data set we have more modes of variation that we would like our reduced space to represent. As a result it is harder for the GMM algorithm to fit mixtures to the data properly so it fits less mixtures to compensate. Although the mixtures chosen by the algorithm are not ideal, they are adequate. The two distributions on the left are good but preferably the distribution on the right should be split into two different distributions. However, as we are only viewing 2 of 12 dimensions there could be unseen factors that are contributing towards this selection.

We now turn to PCA projections showing PCA2 space. We set the parameter determining the number of dimensions in PCA2, t_2, to 10 for this data set (again,

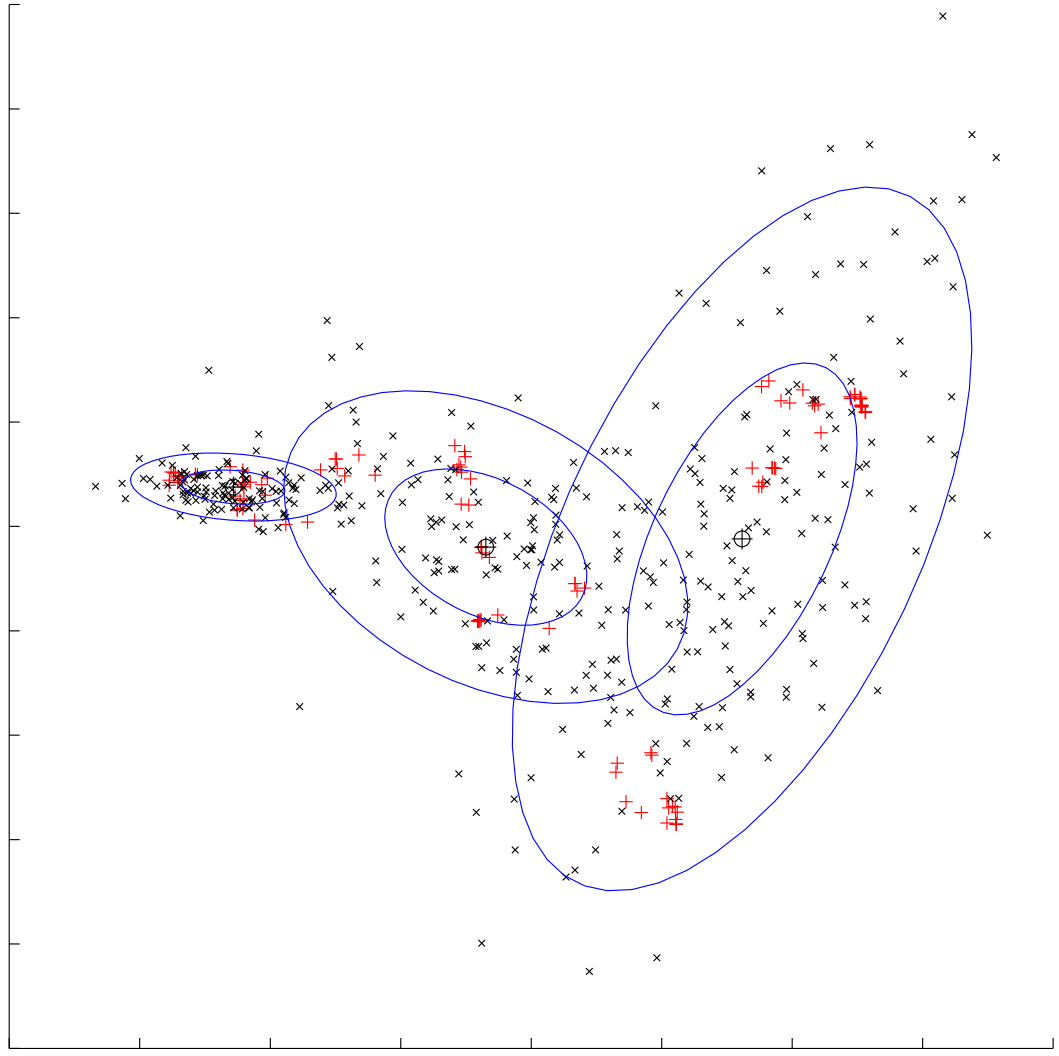


Figure 6.23: A PCA projection of the input set (projected into PCA1 space). The input set is marked with plus signs and the generated quasi-molecules are marked with crosses. Also shown are ellipses indicating the axis of principle variance of the normal distributions determined by the GMM algorithm.

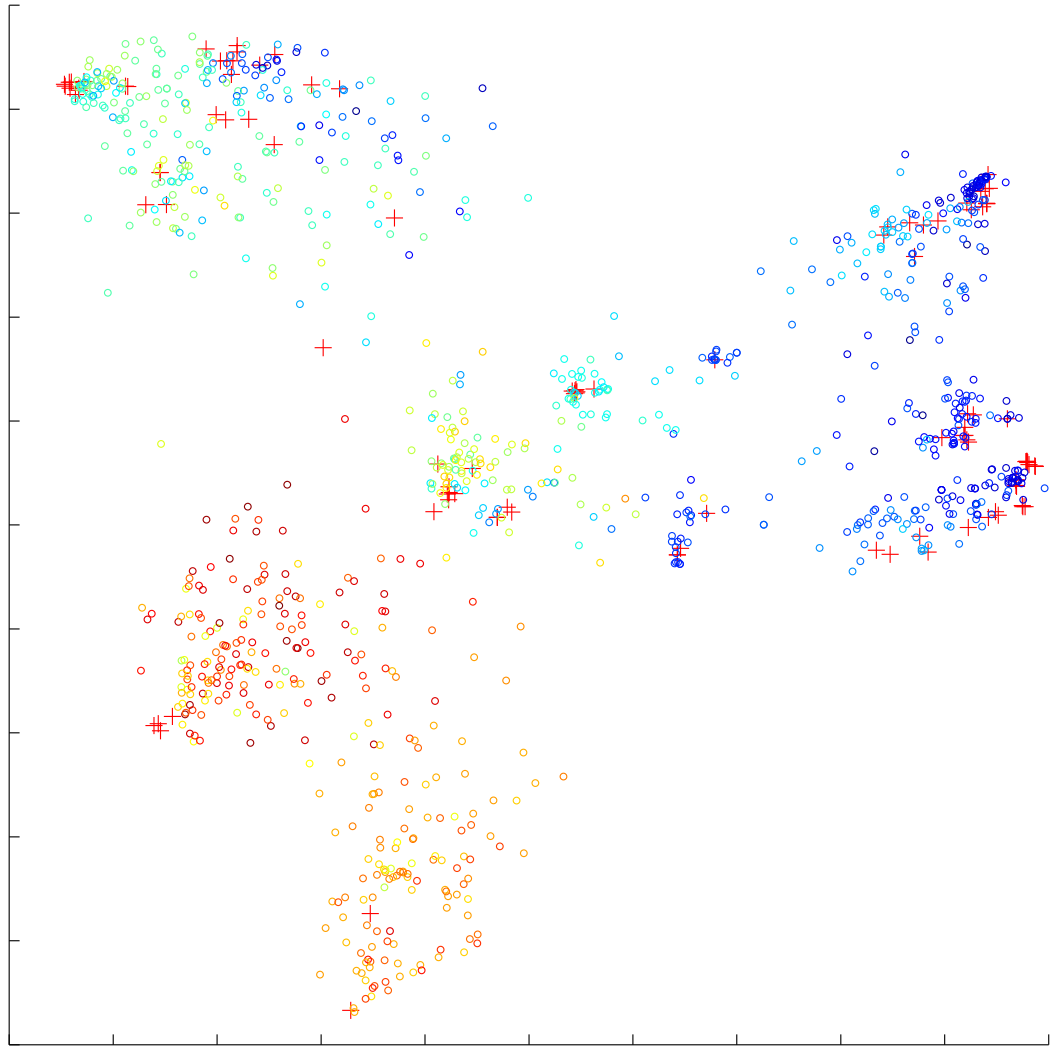


Figure 6.24: A PCA projection of the input set (marked with plus signs) and the projection set (marked with circles). This projection is in PCA2 space.

increased compared to the COX2 data set because of the increased size of the input set).

Figure 6.24 shows the input and projection sets. We have successfully generated projection molecules covering the whole of the input space, and they are also clustered in the regions of that the input molecules are clustered. The only problem here is that we have generated a large number of redundant projection molecules in the bottom left of the plot which reduces the effectiveness of this projection set. We can also see from this figure that the projection set is distinct to the input set, we are not simply replicating the input set in our construction of the projection set. We performed an exhaustive search to check for any projection molecules that appear in the input set and none were found.

In the previous data set the spaces of PCA1 and PCA2 looks quite similar, however in this data set they appear quite different. This is due to the increased size of the projection set. Certain input molecules (specifically those in the bottom left of figure 6.24) were under-represented in the input data and as such when the PCA1 transform was computed the variation they express was also under-represented. However, in the projection set there are a great number of molecules all similar to those outliers. This results in the PCA2 space expressing this variation, and therefore making the two projections appear quite different. However, we argue that this is not a problem since we are only interested in capturing variation in the input set for the model that generates new graphs and this is exactly what basing the GMM in PCA1 space does.

As in the COX2 results we can use a histogram to assess whether the projection graphs are being generated over the whole space of the input graphs. Figure 6.25 shows the percentage of projection graphs that are mapped to a single input graph. Again we see quite uniform results with the exceptions being input graphs 15, 83 and 100 which have significantly more graphs mapped to them. The reason for this is the same as the outliers in the histogram shown in the COX2 data set, input graph 15 is the smallest in the input set and as such many very small projection graphs are mapped to it. Equivalently, 83 and 100 are two of the largest graphs in the input set and graphs in the projection set that are larger than any of those in the input set are mapped to these.

In figure 6.26 we show the generated quasi-molecules in PCA2 space overlayed

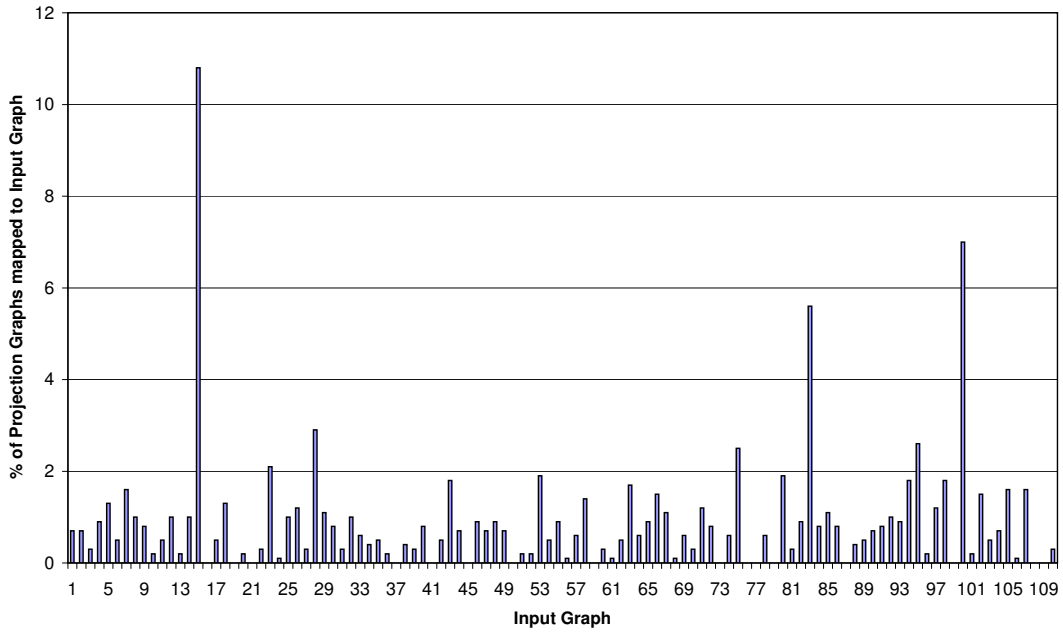


Figure 6.25: A histogram showing the percentage of projection graphs that align to a specific input graph in the hierarchical alignment step.

on top of the input and projection sets. The increased value of t_1 helps in making this recovery and re-projection step successful. The generated quasi-molecules span the space of input molecules well and although there are some outliers, there are not a significant number of them. Specifically, in the void between the top left cluster and the remainder of input graphs there are very few generated quasi-graphs.

In figure 6.27 we show the matches between generated quasi-molecules and the molecules of the projection set. Since these matches are computed in 10 dimensions ($t_2 = 10$) when viewed in 2 dimensions they look a little chaotic.

Finally, we move back to PCA1 space and re-project the subset of the projection set that comprises the generated set into the space of input molecules. Here we see that the distribution of the generated molecules agree with the distribution of input molecules and span the space well. In particular, where clusters of input molecules appear, often clusters of generated molecules do too and where voids lie, there are few generated molecules. In total we produce 199 molecules in the generated set. When Omega is used to compute the 3D conformations of the generated molecules we obtain an average of 65 conformers per generated molecule. This provides Fred with plenty of poses to explore to find the optimal conformer and pose combination.

We now discuss the docking results which are given in 6.29. It is immediately

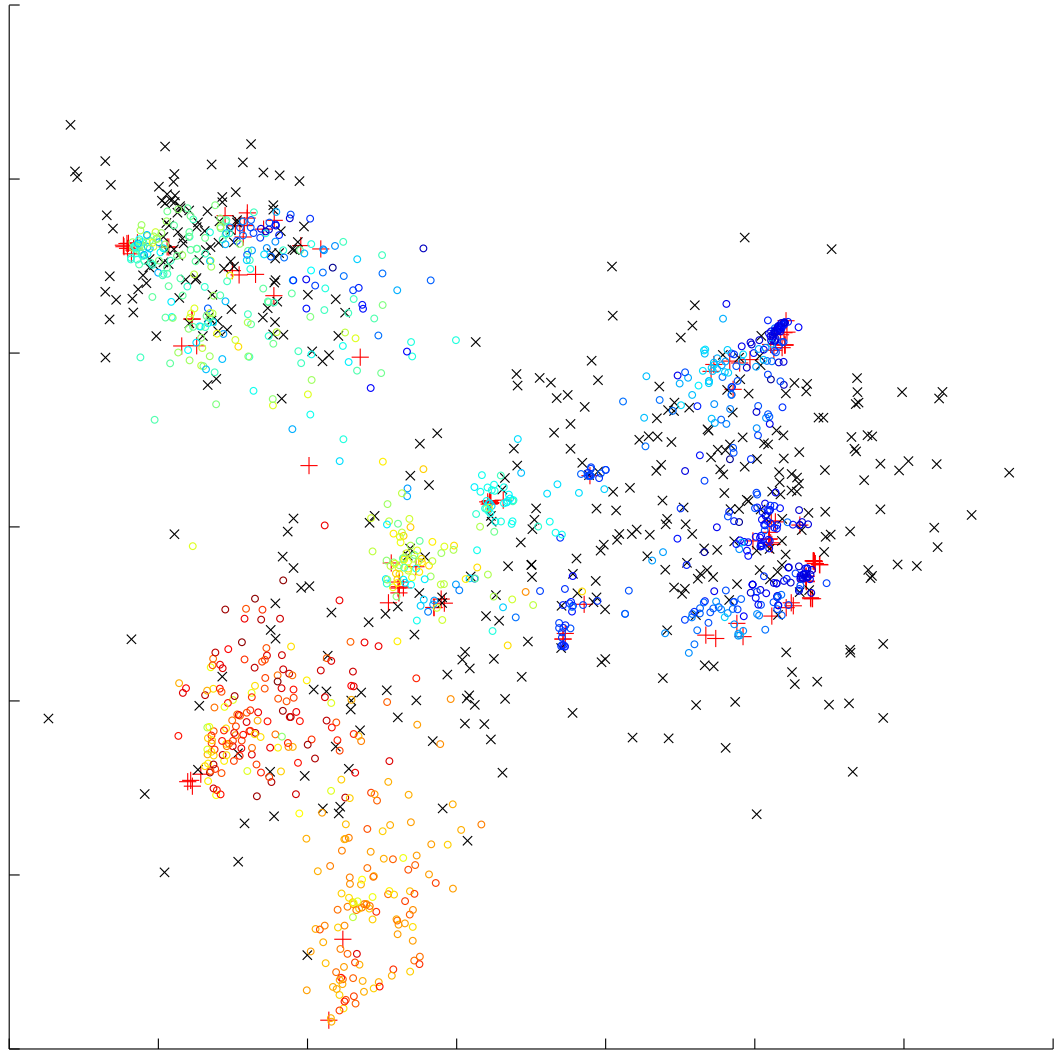


Figure 6.26: A PCA projection of the input set (marked with plus signs), the projection set (marked with circles) and the quasi-generated set (marked with crosses).

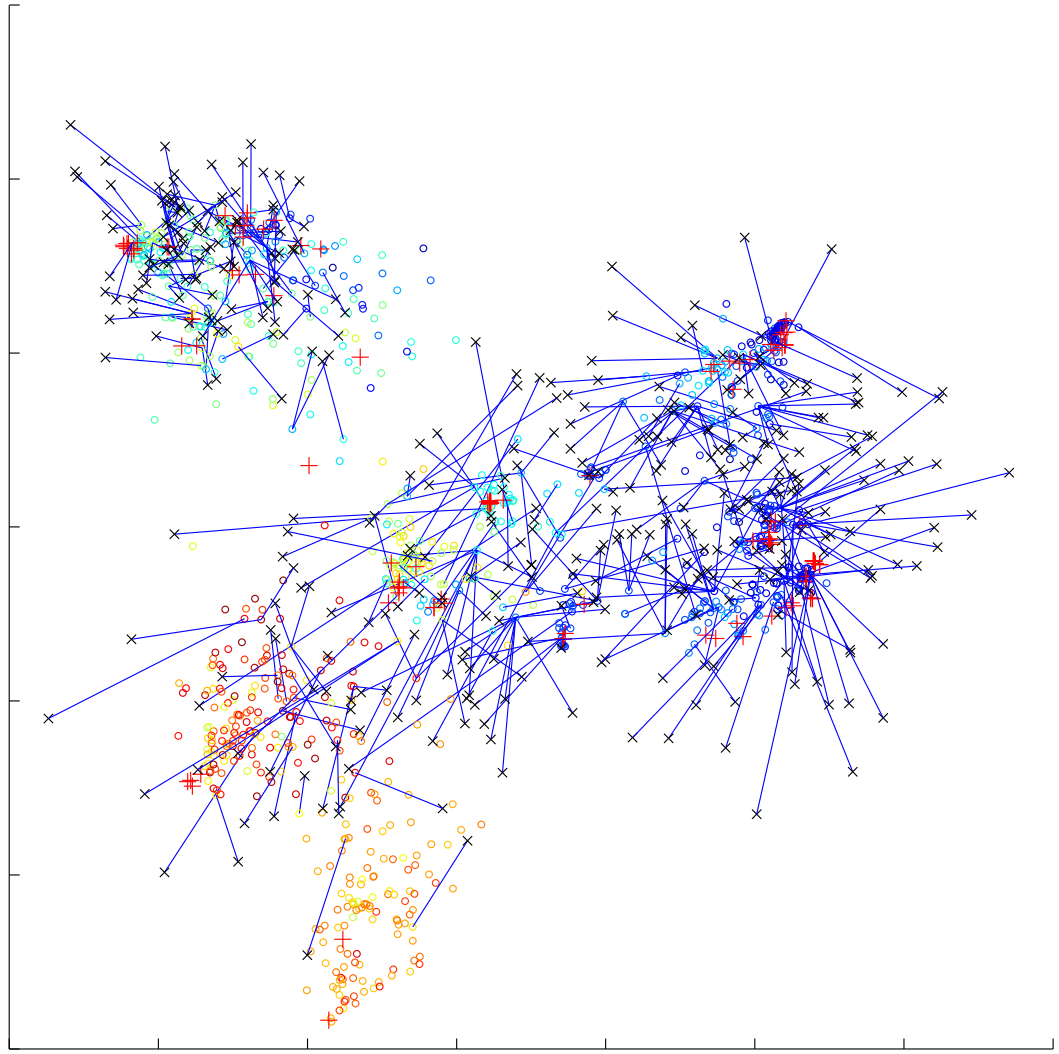


Figure 6.27: A PCA projection with the same data as figure 6.26. Matches between generated quasi-molecules and molecules from the projection set are shown.

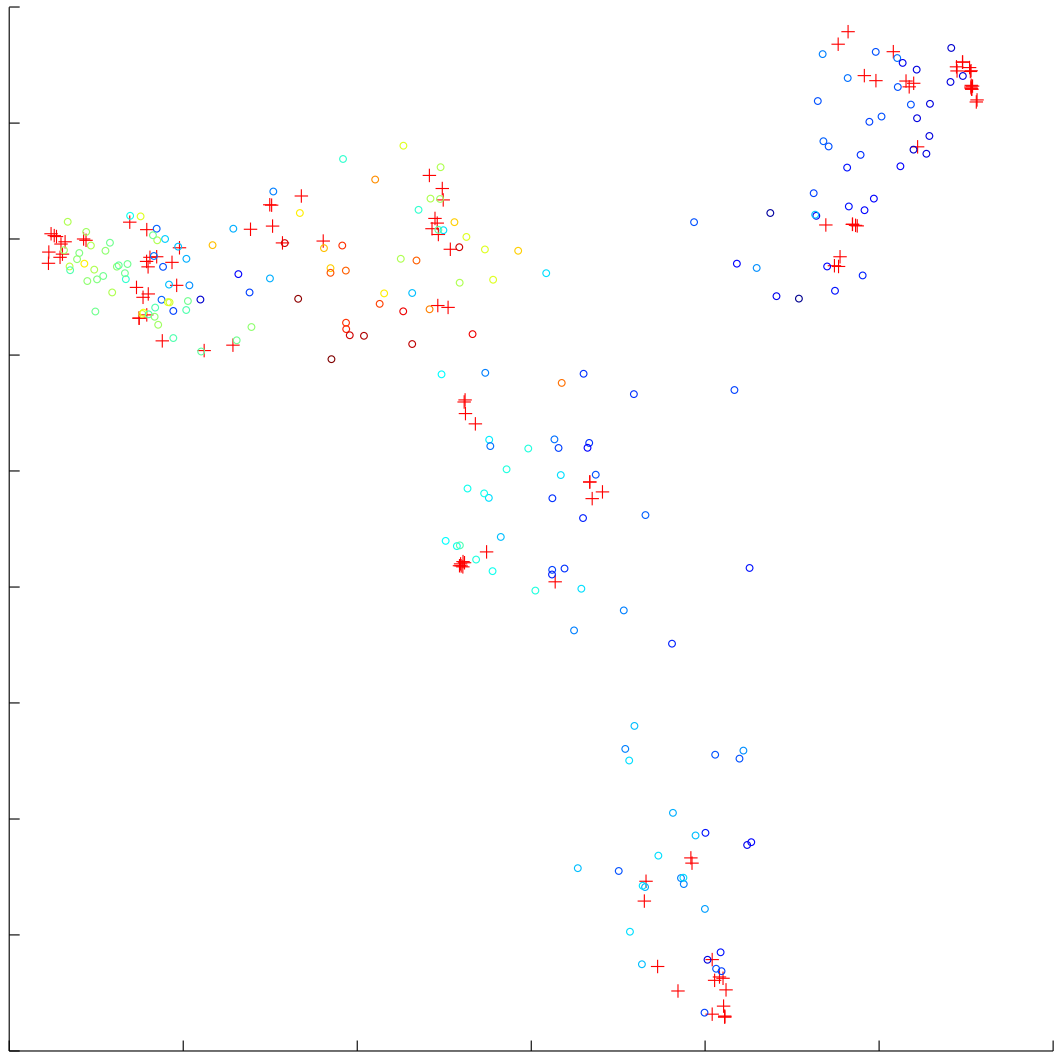


Figure 6.28: A PCA projection showing the input set (plus signs) and the subset of the projection set (circles) that comprise the true generated set.

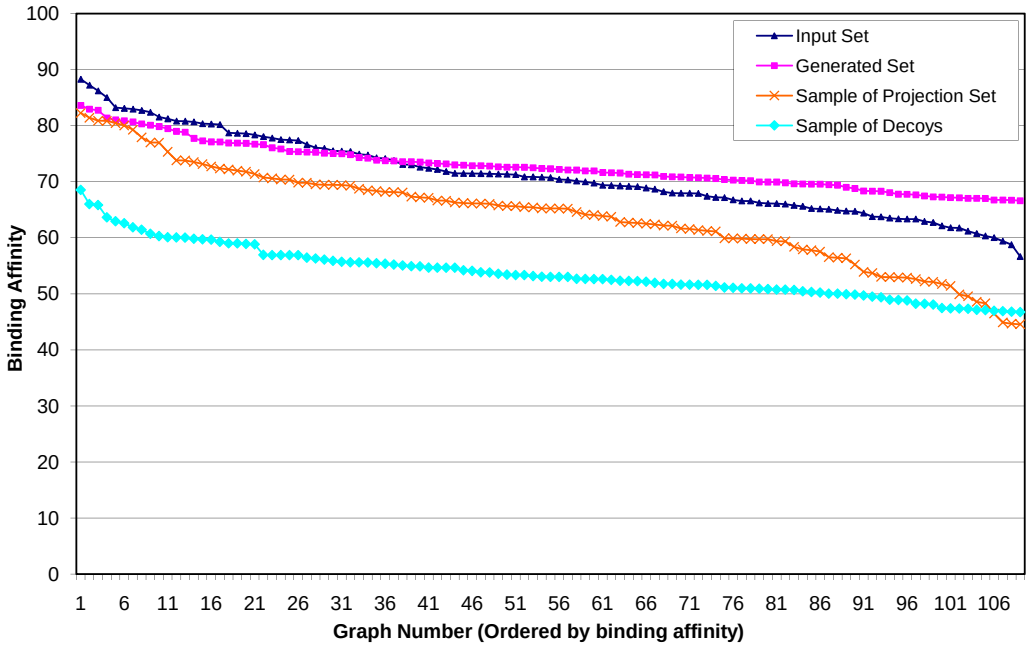


Figure 6.29: The docking scores of the molecules in the input and generated sets sorted by score. In addition a random sample of molecules from the projection and decoy sets are included for comparison.

apparent that the set of generated molecules dock well with this active site. As discussed earlier this is due to the more generic shape of this active site when compared to the COX2 active site. While the generated set appears to perform very well in comparison to the input set it is important to remember that the molecules of the input set are real drugs and have additional constraints imposed such as selectiveness of binding, druglikeness and valid synthesis routes. For example, ligands identified in the lead hopping process that bind with a great number of active sites must be discarded since they will interfere with the function of proteins other than the required target. We postulate that many of the generated molecules would display the same problem, however this is out of the scope of our research.

This view is reinforced when we consider the performance of the sample of decoys compared to the performance of the decoys in the COX2 active site. Recall from the COX2 docking results (6.16) that the performance of the decoys dropped off very quickly. In other words, we identified a few decoys that bound well but the majority performed poorly. Contrast this with the performance of the decoys in the EGFR active site and we see that the decoys for the EGFR protein perform

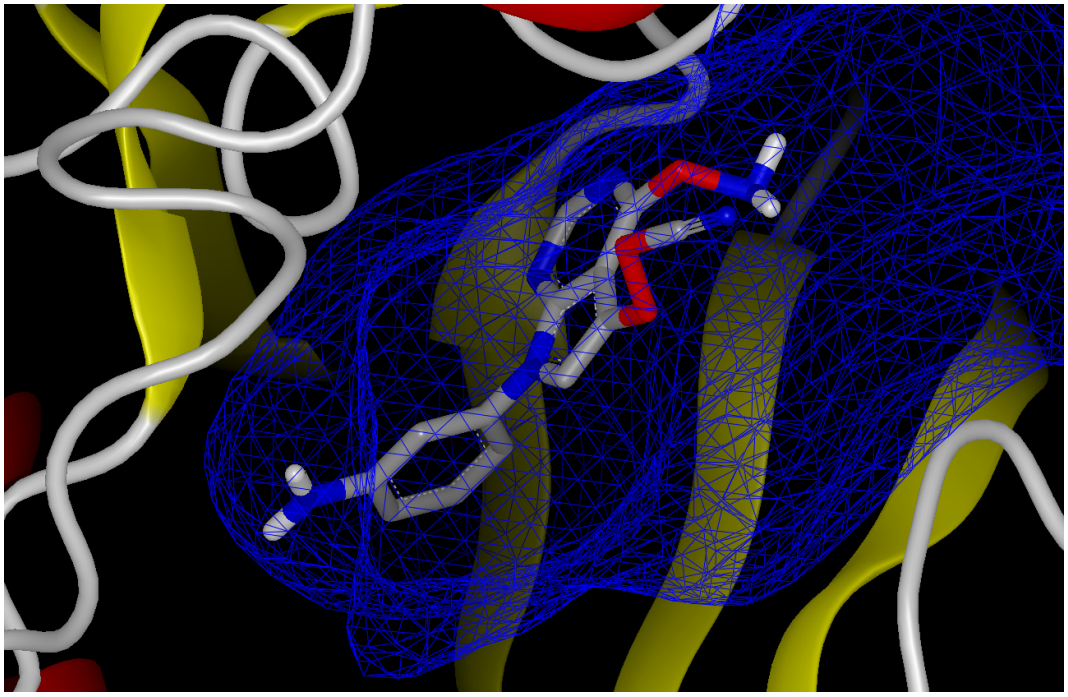


Figure 6.30: The active site of interest in the EGFR protein. Molecule G_1 from figure 6.22 (the molecule from the generated set with highest binding affinity) is shown in its optimal pose as computed by Fred.

much better and do not tail off quickly in the same way the COX2 ones did.[7] This suggests that it is much easier for a random molecule to bind to the EGFR active site than the COX2 active site. The same result is seen when we consider docking a random sample of the projection set.

We conclude this section of results by giving the generated molecule that docks with the highest affinity (G_1 from figure 6.22) in its optimal docking pose in the EGFR active site. This is shown in figure 6.30.

6.4 Conclusion

In this chapter we have demonstrated how a generative model for chemical structure may be constructed. The task of generating chemical structure is significantly harder than that of generating simple and weighted graphs due to the additional set of constraints governing what constitutes a valid chemical structure. The challenges this presents forces us to look at new ways of generating relational data and a wide range of methods were suggested in section 6.1.2 to solve the problem. These

[7]The sets of decoys are different for the two proteins.

range from extensions of previous methods devised in this research to considering a distribution of edit operations on linear descriptors known as SMILES strings, to possibly generating chemical structure in a statistically way by the use of reaction simulation. In the end the method chosen involved both an extension of previous work and the introduction of new ideas to cope with the additional constraints of chemical structures.

The proposed method works by a three stage process. The first step involves the construction of a projection set of valid chemical structures near those of the input set in chemical space. The second step constructs a generative model over the molecules of the input set and allows us to sample quasi-molecules from the distribution. The final step is to map the generated quasi-molecules onto the molecules of the projection set.

This method allows us to approximate sampling chemically correct structures from the distribution of input molecules. The approximation arises from the step involving mapping true samples to molecules from the projection set. While this additional mapping step does introduce a small amount of error into our method we do not believe this to be significant and indeed we have shown via extensive results that the molecules we generate are of high quality for the task they are designed for. Furthermore, this error can be reduced by increasing the number of molecules in the projection set and therefore reducing the average distance of the mapping between a quasi-molecule and a molecule from the projection set.

One of the most challenging aspects of this work was finding a method of performing a global alignment on the graphs from the input and projection sets. This arose as a requirement from the need to compute the similarities between the generated quasi-molecules and the molecules from the projection set. The complications introduced by aligning chemical structures and the large sizes of the sets in question contributed significantly to the complexity of the problem. The solution was to perform a hierarchical alignment between the graphs of the input set and then slot the graphs from the projection set into an appropriate position in the hierarchical alignment tree. This not only resulted in good alignments which are indispensable for constructing our generative model but the complexity increases polynomially only in the size of the input set. In contrast, an addition to the projection set only results in a linear increase in the amount of computation required. This is the main

reason behind the extendability of the approach through increasing the size of the projection set, the additional graphs do not cause the alignment step to become infeasible.

The mechanisms of action in chemical reactions and the field of fragment-based drug design have also played vital roles in this work. The observations by Johnson and Maggiora [59] and Patterson et al [85] that the properties of a molecule are heavily dependant on its structure are of course of vital overall importance. However, the idea that we can construct a useful set of molecules that constitute the projection set takes its ideas from fragment-based drug design. This arises from the observation that molecules in the input set often perform well at their task due to the specific set and arrangement of functional groups they are made from. Therefore, being able to construct more molecules by breaking those from the input set apart and then reassembling them into whole molecules is what makes this approach possible. Of course there are other ways to construct the projection set and some of these are detailed in the future work section. However, we feel the fragment based approach is the most appropriate in this situation.

The idea of a molecule's function being based on its structure leads us to our evaluation domain, that is the task of docking molecules to active sites in proteins. By populating the input set with molecules that perform well at docking with a certain target, and noting that this performance is mostly based on a molecule's structure, we expected that additional structurally similar molecules we generated would also dock successfully to the target. This indeed proved to be the case, at least in the virtual docking environment we used. Molecules were generated for two target sites, one in the EGFR protein and one in the COX2 protein. In both cases we generated molecules that dock with high affinity and this result was reinforced when compared to the docking scores of a random sample of the projection set and a random sample of a decoy set. The EGFR docking scores were especially high due to the rather general shape of the active site in question.

In addition to the docking results, we have shown 2D images of generated molecules, PCA plots of the vector spaces used throughout our process and histograms indicating the distribution of mappings to provide answers to questions that arose during the design of this approach. Some of those are: Is the projection set suitable for mapping generated quasi-molecules to? Does the hierarchical align-

ment step result in vectors that adequately describe the graphs? Is fitting a GMM the correct choice to model the input set with?

In conclusion we note that while the chemical structures we generate perform well on our evaluation domain, there is an incredible amount more that goes into transforming one of these structures into a successful drug. Therefore, we do not claim that any molecules that we generate would be suitable for use as drugs, or even synthesizable in the real world using current techniques. What we do hope however, it that this approach will be useful for exploring the vastness of chemical space in a region where some information about the required structures is already known.

6.4.1 Future Work

There are a number of ways this work can be extended and we list the most relevant of these below.

The hierarchical global alignment step is designed to produce meaningful alignments between molecules even when the projection set is very large in size. However, it is still, by far, the most computationally complex step in the process. There is room for improvement here; in contrast to the method proposed in Chapter 4, we do not directly reconstruct a generated graph from its vectorial description. Instead, we map generated samples to the nearest correct molecule that is available. Therefore, we could use vectorial descriptions of graphs that do not have inverse mappings. In other words, feature vectors may be constructed from graphs but graphs cannot be recovered from the feature vectors. The benefit here is that many of these feature based descriptions are invariant in the ordering of the graph vertices. Therefore, the expensive alignment step could be replaced in favor of a vertex invariant vectorial description[22, 101, 123]. However, it should be noted that feature based representations are not as robust as representations that retain the full set of available information.

So far we have only applied this framework to the domain of chemical structures. However, there is no reason why it cannot be used for any domain that requires constrained graph structure to be generated. The main requirement for extending this approach for a particular type of graphs is that a method of constructing the projection set is available. While we have only used a fragment based approach

for chemical structure, there are many other ways to construct the projection set which we outline later.

A simple type of constrained graphs are planar graphs. The constraint is that they must be drawable in a 2D plane such that edges only intersect at end points. If the generative models described in chapter 4 or 5 were applied to such graphs then it would be possible to generate non-planar graphs. Of course, the constraints on planar graphs are far simpler than chemical structures and generated graphs could be corrected by removing the edges causing the embedding problems. However, if the framework described in this chapter were used then the generation of non-planar graphs would be impossible.

Currently the projection set is constructed using a fragment based approach. While this provides correct molecules within a very simple model of chemistry, to enhance the usefulness of this approach in the real-world better ways of constructing the projection set are required. We outline some of these below:

- The simplest way to improve the molecules contained in the projection set would be to perform a more advanced filtering step. Since we know that we are searching for drug-like molecules we can apply various measures that have been developed that specify how "drug-like" a molecule is. One example of such a measure is Lipinski's Rule of Five [72] which places upper bounds on four molecular criteria. Recently, interest has grown in the area of *lead-likeness*. This concept differs from drug-likeness since drugs require no further alterations where as lead molecules will likely be refined several times. The main result of this is that the molecular weight of a lead will increase and therefore criteria's determining lead-likeness generally specify a lower upper bound on molecule weight. An investigation of leads vs drugs is given by Oprea [84].

- Another possibility would be to use reaction transform networks to generate new molecules. In these systems, a medicinal chemist specifies a set of reactants and rules by which they may be reacted. The possible reactions are then applied to the reactants thus generating a set of products. The problem with this approach is that it requires extensive training and domain knowledge to set up the correct reactions. However, by producing the projection set

in a manner that resembles real-world reactions, the chance of being able to synthesize the molecules is much higher.

- Another method of populating the projection set would be to extract a subset of molecules from a large database. By computing a crude set of similarities between the elements of a large chemical database (such as ZINC [10]) and the molecules of the input set, the molecules in the database could be ranked by similarity to those in the input set. The top x molecules from this list could then be used to populate the projection set.

Our final suggestion for future work on this topic is to implement a more accurate method of estimating the parameters of the GMM. Currently we obtain sufficient results when working on data with up to 12 dimensions. However, the error introduced by moving between PCA spaces when generating new molecules (see section 6.2.5) could be reduced if the GMM were based in a higher dimensional space.

6.5 Symbols

$\mathcal{S}, S_k, \mathbf{S}_k, \mathring{\mathbf{S}}_k, \mathbf{s}_k, \mathbf{s}'_k, \mathbf{s}''_k$	The set of input molecules is termed $\mathcal{S}$. A graph drawn from this set is S_k. An adjacency matrix representation of this graph is given by $\mathbf{S}_k$ and an aligned version of this matrix is $\mathring{\mathbf{S}}_k$. This graph can be represented by a long vector $\mathbf{s}_k$. When in PCA1 space this becomes $\mathbf{s}'_k$ and in PCA2 space $\mathbf{s}''_k$.
$\mathcal{P}, P_k, \mathbf{P}_k, \mathring{\mathbf{P}}_k, \mathbf{p}_k, \mathbf{p}'_k, \mathbf{p}''_k$	The set of projection molecules is termed $\mathcal{P}$. A graph drawn from this set is P_k. An adjacency matrix representation of this graph is given by $\mathbf{P}_k$ and an aligned version of this matrix is $\mathring{\mathbf{P}}_k$. This graph can be represented by a long vector $\mathbf{p}_k$. When in PCA1 space this becomes $\mathbf{p}'_k$ and in PCA2 space $\mathbf{p}''_k$.
$\mathcal{G}, G_k, \mathbf{g}_k, \mathbf{g}'_k, \mathbf{g}''_k$	The set of generated molecules is termed $\mathcal{G}$. A graph drawn from this set is G_k. This graph can be represented by a long vector $\mathbf{p}_k$. When in PCA1 space this becomes $\mathbf{p}'_k$ and in PCA2 space $\mathbf{p}''_k$.
$B_{ki}, \mathcal{B}, \mathcal{B}_k$	Fragment i from molecule S_k is termed F_{ki}. The set $\mathcal{B}_k$ contains all the fragments found in graph S_k. The set $\mathcal{B}$ contains all fragments i.e. $\cup_{k=1}^{\lvert\mathcal{S}\rvert}\mathcal{B}_k$.
ζ	A function that maps a fragment B_{ki} to the number of times it occurs in occurs in input molecule S_k.
$F_i, \mathcal{F}, \mathcal{F}_{ij}$	The set of unique fragments in $\mathcal{B}$ is given by $\mathcal{F}$. For each unique fragment F_i, $\mathcal{F}_{ij}$ gives the occurrences of the fragment in $\mathcal{B}$ where it occurs with multiplicity greater than or equal to j.
$P_i(j)$	Gives the probability of fragment F_i occurring with multiplicity j.
$R_i, \mathbf{R}_i, \mathring{\mathbf{R}}_i$	Reference graph i is given by R_i with adjacency matrix $\mathbf{R}_i$ and aligned adjacency matrix $\mathring{\mathbf{R}}_i$.
$\mathbf{M}(Y, X)$	The matching matrix that aligns graph X to Y.
$\mathbf{D}, \mathbf{D}'$	Distance matrices.
$w(S_i)$	The alignment sequence that is applied to input graph S_i.

$\boldsymbol{\mu}_{\mathcal{S}}$, $\boldsymbol{\Phi}_{\mathcal{S}}$, $\tilde{\boldsymbol{\Phi}}_{\mathcal{S}}$, t_1	The ingredients of the PCA1 projection: $\boldsymbol{\mu}_{\mathcal{S}}$ is the mean vector, $\boldsymbol{\Phi}_{\mathcal{S}}$ gives the principle components and $\tilde{\boldsymbol{\Phi}}_{\mathcal{S}}$ are the top t_1 principle components.
$\boldsymbol{\mu}_{\mathcal{S}\cup\mathcal{P}}$, $\boldsymbol{\Phi}_{\mathcal{S}\cup\mathcal{P}}$, $\tilde{\boldsymbol{\Phi}}_{\mathcal{S}\cup\mathcal{P}}$, t_2	The ingredients of the PCA2 projection: $\boldsymbol{\mu}_{\mathcal{S}\cup\mathcal{P}}$ is the mean vector, $\boldsymbol{\Phi}_{\mathcal{S}\cup\mathcal{P}}$ gives the principle components and $\tilde{\boldsymbol{\Phi}}_{\mathcal{S}\cup\mathcal{P}}$ are the top t_2 principle components.
$\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, w_i, $\boldsymbol{\Lambda}_i$, $\boldsymbol{\Phi}_i$	The ingredients of component i of the GMM: $\boldsymbol{\mu}_i$ is the mean vector, $\boldsymbol{\Sigma}_i$ is the covariance matrix, w_i is the weight of component$i$, $\boldsymbol{\Lambda}_i$ and $\boldsymbol{\Phi}_i$ are the eigenvalues and eigenvectors of the covariance matrix.
v	A function for selecting which component of the GMM should be sampled.
$\mathbf{b}$	A parameter vector of a multivariate normal distribution.
ρ	A function that maps a generated quasi-graph onto the nearest graph from the projection set.

Chapter 7

Conclusion

In this chapter we will begin by giving a summary of our work and the most important conclusions that we have drawn. Secondly, we will describe the future work that we feel would be most valuable to pursue.

7.1 Summary of Contribution

We have developed and evaluated methods for constructing generative models of relational graphs. In each case we insist that our models are fully generative in the sense that new examples may be sampled from the distribution. We provide a summary of our contribution for each chapter in this thesis.

7.1.1 Mixing Spectral Representations of Graphs

We began by investigating the use of the spectral representation to mix graph structure, which was detailed in chapter 3. While this approach is not a generative model, it paves the way for the generative models defined on the spectral decomposition which are described in chapter 4. The use of the spectral representation is an interesting one for mixing graph structure due to the separation of scales of connectivity within the graph. By selecting the eigenmodes of sample graphs with different proportions, we can combine different levels of structure from different graphs into a single graph. However, moving from the spectral domain back to the graph domain is not a trivial process due to the spectral representation of a mixed graph not displaying the required properties. To solve this we apply a thresholding step to the

elements of the recovered matrix representation.

Using this approach, we performed pair-wise mixing between graphs from two different sets in an effort to generate new graphs that displayed some properties of both sets. By computing the edit distance between all pairs of graphs we were able to perform an MDS embedding to visualize the results. This result showed that we had successfully generated mixed graphs that spanned the space between the two sets.

An approach described by Ferrer et al [37] uses the spectral representation to construct a median graph by performing sequential updates. By mixing all graphs from the set in the same proportion, we can directly compute a spectral representation of a median graph. Using our recovery technique, a graph can be found from this spectral representation. In another paper by Ferrer et al [38] they have compared their approach of computing the median through incremental updates to our approach of directly computing the median through mixing. They report that both methods perform similarly, with the exception of high noise in which their approach performs better. The evaluation domain they use is that of line drawings.

7.1.2 Vectorial Generative Models for Graphs

In chapter 4 we introduced our first generative model. Building on the models of Luo et al [75] and Xiao & Hancock [126], we defined a number of different vector spaces describing graphs on which a generative model could be based. This basic procedure holds for all models; after an initial alignment step, in which all graphs in the sample set are aligned to the largest, we construct a vectorial representation for each graph. We fit a normal distribution over the vector set by performing PCA on the vectorial representations. This identifies the principle modes of variation in the vectors and hence also in the structure of the graphs. Using standard techniques we can fit a normal distribution to the vectors and sample from it to generate new examples. However, due to the sampling process we will not always sample vectors that display the required properties to reconstruct into graphs. For example, when using the spectral representation the eigenvector set may not be orthogonal. Nevertheless, we define a recovery step for all our approaches that allows us to reconstruct graphs from the generated vectorial descriptions.

We defined simple models such as those based directly on the adjacency matrix

(the model proposed in [75]) and on the Laplacian. By moving to the spectral representation we constructed models that better account for the structural variations present in a graph set. These models consisted of a single vector describing the spectral representation (similar to that described in [126]) and a dual-vector representation where the eigenvalues and eigenvectors are modeled separately. We found this necessary, because due to the combining of eigenvalues and eigenvectors into a single vector, the eigenvalue distribution becomes distorted. As mentioned before, the eigenvector sets that are generated using this model do not display all of the required properties to be a valid spectral decomposition. While it is possible to correct this to some extent, by using an orthogonalization step for example, we sought methods that would generate only correct matrices.

The solution lies in the notion that the vector space of valid eigenvector matrices resides on a manifold. By using the exponential map we were able to define transformations to and from this manifold. Using this we defined a model in which each eigenvector matrix is projected into a tangent space on which a normal distribution may be formed. Eigenvector matrices generated using this method display all the required properties and therefore the reconstruction step is simple. We extended our use of manifolds to Laplacian matrices. By modeling our Laplacian matrices on the manifold of positive definite matrices we enforce the generation of valid Laplacians.

We performed an extensive set of experiments on all the models to assess the following points: the classification accuracy, the distributions they form and the distributions of generated graphs, the compactness of the models and the appropriateness of fitting a normal distribution. By using three different data sets which range in the type of noise present from random to structural, we were able to observe some overall trends.

We found that simple models such as those based on the adjacency and Laplacian dealt very well with the more random noise but not so well with structural noise. This is due to the relative simplicity of the models. On the other hand, the model based on the dual-vector spectral representation performed poorly on the random data set but very well on the structural data sets. This was consistent with our expectations as the spectral representation is more capable when used with highly structural data. The single vector spectral representation performed quite poorly throughout the experiments largely due to the eigenvalue magnitude problems men-

tioned earlier. Finally, while the addition of the orthogonal map to the dual spectral model should have provided tangible benefits, this did not seem to translate through into our experimental results. On the other hand, it did simplify the recovery of generated matrix representations.

7.1.3 Parts Based Generative Models for Graphs

In chapter 5 we introduced our second generative model for graph structure. In this model, instead of representing the graphs as whole entities, we construct our model using a decomposition based approach. If the data in question is applicable to a segmentation process then this results in a more accurate generative model. Furthermore, the decomposition step reduces the computational complexity of the approach making it applicable to very large graphs.

The approach begins by decomposing each sample graph into a number of subgraphs. The subgraphs and the cut connection between subgraphs are stored and form the input to our model. The subgraphs from all sample graphs are clustered to determine which represent similar structure. With each subgraph assigned to a cluster we can change our view of connections between subgraphs to connections between clusters. We then construct models determining the distribution of clusters present in each sample graph, the distribution of subgraphs within each cluster and the distribution of connections between clusters. By sampling from all three models we can generate a new graph that has structural properties similar to those of the sample set.

However, there are some issues with this approach. The key problem is that by grouping connections between clusters together we are making the assumption that subgraphs from the same cluster are always connected in the same way, which is not a realistic assumption. For example, in situations where a number of instances of a cluster appear in a single sample graph it is very unlikely that they will all be connected in the same way. An example of this would be scenes involving multiple instances of the same object. We suggested some solutions to this problem in the chapter and we repeat the most promising of these in the future work mentioned later.

Despite the problems with this approach, when the sample data suited the method we saw good results. We used both synthetic and real-world data to assess the utility

of the generative model. Both data sets took the form of point sets since this provided us with a way to actually visualize the generated graphs. The real-world data set was produced by performing a motion capture step on an articulated object. We were then able to generate graphs describing new poses of the object and visualize them through an optimization based drawing technique.

7.1.4 Generative Models for Chemical Structures

In chapter 6 we considered the application of the generative models described previously to the domain of chemical structure. We combined aspects of both generative models as well as new methods to form our approach. Due to chemical structure being constrained by the laws of chemistry we needed to implement some method of ensuring we only generated valid structures. This could be accomplished in a number of ways that were outlined in section 6.1.2. The solution we selected approximated sampling from the input distribution by (a) taking a sample that would probably represent an incorrect chemical structure (termed a quasi-molecule) and (b) projecting this onto the nearest correct chemical structure. For this to work we required a method of constructing a "projection" set of valid chemical structures that would be as close to the generated quasi-molecules as possible. We also required a method of computing the closest valid molecule to a quasi-molecule.

The construction of the projection set was solved by using the chemical fragments contained in the molecules of the input set to construct it. By decomposing each input molecule into its functional groups we could build new similar molecules by combining the functional groups into a new molecule. This approach has its origins in the parts based approach described in chapter 5, chemoinformatics tools for performing chemical structure elucidation and fragment based drug discovery. The link to fragment based drug discovery also helps our method perform well in the evaluation domain. By noting that the way drugs function at a target is largely due to the functional groups they possess, we generated new candidates for the target by combining the functional groups in previously unseen ways.

The second requirement of assessing the similarity between a molecule from the projection set and a quasi-molecule is a harder problem than it appears. Since we only have a vectorial description of the generated quasi-molecule we must ensure the molecules from the projection set reside in the same vector space. The result

of this is that the input molecules and projection molecules must all be placed in a common alignment. However, due to the size of the projection set this is a challenging task. Furthermore, since the applicability of the method directly depends on the diversity and size of the projection set, the alignment step must be efficient. After considering a number of possible solutions we settled on performing a full hierarchical alignment on the molecules from the input set and then aligning the molecules from the projection set by slotting them into the most appropriate place in the hierarchical alignment tree. The result of this was that each projection molecule must only be aligned to each input molecule and therefore only a linear increase in computation is seen when another molecule is added to the projection set.

The field of drug discovery is largely about exploring chemical space in a search for candidate drugs against a specific target. The method we propose offers a way of doing this. By noting that a drug's function is largely determined by its structure, if we populate the sample set with drugs for a specific target, we should expect to generate chemical structures that also perform well with a specific target. To evaluate this we used a docking program to simulate the interactions that take place between a potential drug and the active site of a protein. We used two data sets, one with drugs for the COX2 protein and another with drugs for the EGFR protein.

By measuring the affinity with which a molecule binds to an active site a score was produced for each generated molecule. By comparing this score to the scores from the input set and a set of randomly sampled molecules, we were able to show that our approach does indeed seem to statistically generate new molecules with properties similar to those in the sample set. While there is no guarantee these chemical structures would be suitable as drugs, or even possible to synthesize in the real-world, in our evaluation domain they performed well.

Furthermore, by comparing the scores of the generated molecules to the scores of a random selection of molecules from the projection set we were able to gain some measure of efficiency. In other words, this comparison gave an indication of whether we were selecting the best subset of molecules from the projection set. Some care must be taken with this measure since we would not expect to produce molecules from areas of the projection set that perform well but are not represented in the sample set. Nevertheless, in both the data sets we used we saw high efficiency.

7.2 Future Work

For the work in mixing spectral representations of graphs we would like to extend the experiments we performed on computing graph medians. Given the recent advances in methods for computing graph medians both exactly and approximately we would like to experiment with larger graphs and larger sets of graphs to see if our approach scales well. The deviation from the true generalized median as found by an exact method would provide a good indication of the true performance of our approach.

We would like to extend the work on vectorial generative models for graphs with an investigation into why applying the orthogonal map to the dual spectral method resulted in little improvement. In theory, it is not clear why it should reduce the performance of the dual spectral method and indeed, we expected it to enhance the performance.

The most pressing issue with the parts-based approach is the assumption that subgraphs representing similar structure are connected similarly. In chapter 5 we outlined a possible solution to this. In order to model a set of subgraphs which are from the same cluster but are connected differently, we require another level of clustering. This time the clustering would be performed on the connections between a cluster pair. This would allow us to construct a mixture model over the connections between a cluster pair thus representing all the different ways that particular cluster pair can be connected. From this model a new set of connections could be sampled.

The applications of our generative models to chemical structure has many possible avenues of future work. We feel the most relevant would be to investigate different ways of constructing the projection set. While the current method is sufficient for our simple model of chemistry, any real world application of our approach would require a more robust method of constructing the projection set.

This method could be one of the reaction transform approaches outlined in the literature. In this method a medicinal chemist can specify the type of chemical structures that are to be produced and our model can then suggest the most statistically significant ones in terms of the input set. Another approach would be to compute a coarse set of similarities between the sample set and a large database of chemical structures, such as the ZINC library. Then the top x most similar structures for example could be extracted and used to populate the projection set.

Bibliography

[1] Daylight Fingerprints - Screening and Similarity. http://www.daylight.com/dayhtml/doc/theory/theory.finger.html.

[2] Daylight Reaction Toolkit. http://www.daylight.com/products/reaction_kit.html.

[3] FRED (Fast Rigid Exhaustive Docking) from Open Eye Scientific Software. http://www.eyesopen.com/products/applications/fred.html.

[4] Open Babel: The Open Source Chemistry Toolbox. http://openbabel.org/wiki/Main_Page.

[5] Open Eye Scientific Software. http://www.eyesopen.com.

[6] OpenEye Toolkit: Chemical Reaction Simulation. http://www.eyesopen.com/docs/docs-v1.7.0-2/html/OEChemTK-c++/reactions.html.

[7] The Daylight Chemical Toolkit. http://www.daylight.com.

[8] The IUPAC International Chemical Identifier. http://www.iupac.org/inchi/.

[9] The Protein Data Bank. http://www.wwpdb.org.

[10] ZINC: A free database for virtual screening. http://zinc.docking.org.

[11] A. T. Balaban. Applications of Graph Theory in Chemistry. *Journal of Chemical Information and Computer Sciences*, 25(3):334–343, 1985.

[12] J. M. Barnard. A Comparison of Different Approaches to Markush Structure Handling. *Journal of Chemical Information and Computer Sciences*, 31(1):64–68, 1991.

[13] J.M. Barnard, G.M. Downs, A. von Scholley-Pfab, and R.D. Brown. Use of Markush structure analysis techniques for descriptor generation and clustering of large combinatorial libraries. *Journal of Molecular Graphics and Modelling*, 18(4-5):452–463, 2000.

[14] S. C. Basak, V. R. Magnuson, G. J. Niemi, and R. R. Regal. Determining structural similarity of chemicals using graph-theoretic indices. *Discrete Applied Mathematics*, 19(1-3):17–44, 1988.

[15] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[16] C. Bissantz, G. Folkers, and D. Rognan. Protein-based virtual screening of chemical databases. 1. Evaluation of different docking/scoring combinations. *Journal of Medicinal Chemistry*, 43(25):4759–4767, 2000.

[17] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[18] H. Bunke. Error correcting graph matching: On the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):917–922, 1999.

[19] H. Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253, 1983.

[20] H. Bunke and S. Günter. Weighted mean of a pair of graphs. *Computing*, 67(3):209–224, 2001.

[21] H. Bunke, A. Münger, and X. Jiang. Combinatorial search versus genetic algorithms: A case study based on the generalized median graph problem. *Pattern Recognition Letters*, 20(11-13):1271–1277, 1999.

[22] T. Caelli and S. Kosinov. An Eigenspace Projection Clustering Method for Inexact Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):515–519, 2004.

[23] R. E. Carhart, D. H. Smith, H. Brown, and C. Djerassi. Applications of artificial intelligence for chemical inference. XVII. Approach to computer-assisted elucidation of molecular structure. *Journal of the American Chemical Society*, 97(20):5755–5762, 1975.

[24] R. E. Carhart, D. H. Smith, N. A. B. Gray, J. G. Nourse, and C. Djerassi. Applications of artificial intelligence for chemical inference. 37. GENOA: A computer program for structure elucidation utilizing overlapping and alternative substructures. *The Journal of Organic Chemistry*, 46(8):1708–1718, 1981.

[25] R. E. Carhart, D. H. Smith, and R. Venkataraghavan. Atom pairs as molecular features in structure-activity studies: definition and applications. *Journal of Chemical Information and Computer Sciences*, 25(2):64–73, 1985.

[26] B. D. Christie and M. E. Munk. Structure generation by reduction: a new strategy for computer-assisted structure elucidation. *Journal of Chemical Information and Computer Sciences*, 28(2):87–93, 1988.

[27] F. Chung. *Spectral graph theory*. American Mathematical Society, 1997.

[28] D.E. Clark, M.A. Firth, and C.W. Murray. MOLMAKER: de novo generation of 3D databases for use in drug design. *Journal of Chemical Information and Computer Sciences*, 36(1):137–145, 1996.

[29] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431(21):931–945, 2004.

[30] M. L. Contreras, R. Rozas, and R. Valdivia. Exhaustive Generation of Organic Isomers. 3. Acyclic, Cyclic, and Mixed Compounds. *Journal of Chemical Information and Computer Sciences*, 34(3):610–616, 1994.

[31] D.A. Cosgrove and P.W. Kenny. BOOMSLANG: A program for combinatorial structure generation. *Journal of Molecular Graphics*, 14(1):1–5, 1996.

[32] S. G. Dahl and I. Sylte. From genomics to drug targets. *Journal of Psychopharmacology*, 20(4 Supplement):95–99, 2006.

[33] J. Drews. Drug Discovery: A Historical Perspective. *Science*, 287(5460):1960, 2000.

[34] J. Dugundji and I. Ugi. An algebraic model of constitutional chemistry as a basis for chemical computer programs. *Topics in Current Chemistry*, 39(1):19–64, 1973.

[35] A. Etkin. Drugs and Therapeutics in the Age of the Genome. *Journal of the American Medical Association*, 284(21):2786–2787, 2000.

[36] M. Ferrer. *Theory and Algorithms on the Median Graph. Application to Graph-based Classification and Clustering*. PhD thesis, Universitat Aut'onoma de Barcelona, 2008.

[37] M. Ferrer, F. Serratosa, and A. Sanfeliu. Synthesis of median spectral graph. *Lecture Notes in Computer Science*, 3523:139–146, 2005.

[38] M. Ferrer, F. Serratosa, and E. Valveny. Evaluation of spectral-based methods for median graph computation. *Lecture Notes in Computer Science*, 4478:580–587, 2007.

[39] M. Ferrer, F. Serratosa, and E. Valveny. On the relation between the median and the maximum common subgraph of a set of graphs. *Lecture Notes in Computer Science*, 4538:351–360, 2007.

[40] M. Ferrer, E. Valveny, and F. Serratosa. Bounding the size of the median graph. *Lecture Notes in Computer Science*, 4478:491–498, 2007.

[41] M. Ferrer, E. Valveny, and F. Serratosa. Median graphs: A genetic approach based on new theoretical properties. *Pattern Recognition*, In Preperation, 2009.

[42] M. Ferrer, E. Valveny, F. Serratosa, and H. Bunke. Exact median graph computation via graph embedding. *Lecture Notes in Computer Science*, 5342:15–24, 2008.

[43] M. Ferrer, E. Valveny, F. Serratosa, K. Riesen, and H. Bunke. An approximate algorithm for median graph computation using graph embedding. In *Proceedings of the 19th International Conference on Pattern Recognition*, pages 1–4, 8–11 Dec. 2008.

[44] M.A.T. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on pattern analysis and machine intelligence*, 24(3):381–396, 2002.

[45] K. Funatsu, N. Miyabayashi, and S. Sasaki. Further development of structure generation in the automated structure elucidation system CHEMICS. *Journal of Chemical Information and Computer Sciences*, 28(1):18–28, 1988.

[46] J. Gallier and D. Xu. Computing exponentials of skew-symmetric matrices and logarithms of orthogonal matrices. *International Journal of Robotics and Automation*, 17(4):10–20, 2002.

[47] V.J. Gillet, W. Newell, P. Mata, G. Myatt, S. Sike, Z. Zsoldos, and A.P. Johnson. SPROUT: Recent developments in the de novo design of molecules. *Journal of Chemical Information and Computer Sciences*, 34(1):207–217, 1994.

[48] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, 1996.

[49] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[50] W. H. Haemers and E. Spence. Enumeration of cospectral graphs. *European Journal of Combinatorics*, 25(2):199–211, 2004.

[51] M. Hann and R. Green. Chemoinformatics - a new name for an old problem? *Current Opinion in Chemical Biology*, 3(4):379–383, 1999.

[52] C.M.W. Ho and G.R. Marshall. DBMAKER: A set of programs to generate three-dimensional databases based upon user-specified criteria. *Journal of Computer-Aided Molecular Design*, 9(1):65–86, 1995.

[53] J. Hser. *High-throughput screening in drug discovery*. Wiley, 2006.

[54] L. Hu, M. L. Benson, R. D. Smith, M. G. Lerner, and H. A. Carlson. Binding MOAD (Mother Of All Databases). *Proteins: Structure, Function, and Bioinformatics*, 60(3):333–340, 2005.

[55] N. Huang, B. K. Shoichet, and J. J. Irwin. Benchmarking sets for molecular docking. *Journal of Medicinal Chemistry*, 49(23):6789–6801, 2006.

[56] R.W. Irving. An Efficient Algorithm for the "Stable Roommates" Problem. *Journal of Algorithms*, 6(4):577–595, 1985.

[57] W. Jahnke and D. A. Erlanson. *Fragment-based approaches in drug discovery*. Wiley, 2006.

[58] X. Jiang, A. Münger, and H. Bunke. On median graphs: Properties, algorithms and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1144–1151, 2001.

[59] M. A. Johnson and G. M. Maggiora. *Concepts and applications of molecular similarity*. Wiley, 1990.

[60] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor. Development and validation of a genetic algorithm for flexible docking. *Journal of Molecular Biology*, 267(3):727–748, 1997.

[61] M. Kontoyianni, L. M. McClellan, and G. S. Sokol. Evaluation of docking performance: Comparative data on docking algorithms. *Journal of Medicinal Chemistry*, 47(3):558–565, 2004.

[62] D. E. Koshland. Application of a theory of enzyme specificity to protein synthesis. *Proceedings of the National Academy of Sciences*, 44(2):98–104, 1958.

[63] H. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[64] A. Kume and A. T. A. Wood. Saddlepoint approximations for the Bingham and Fisher-Bingham normalising constants. *Biometrika*, 92(2):465–476, 2005.

[65] I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langridge, and T. E. Ferrin. A geometric approach to macromolecule-ligand interactions. *Journal of Molecular Biology*, 161(2):269–288, 1982.

[66] P.S. Kutchukian, D. Lou, and E.I. Shakhnovich. FOG: Fragment Optimized Growth Algorithm for the de Novo Generation of Molecules Occupying Druglike Chemical Space. *Journal of Chemical Information and Modeling*, 49:1630–1642, 2009.

[67] J. S. Lazo. Rear-view Mirrors and Crystal Balls: A Brief Reflection on Drug Discovery. *Molecular Interventions*, 8(2):60–63, 2008.

[68] A. R. Leach, J. Bradshaw, D. V. S. Green, M. M. Hann, and J. J. Delany III. Implementation of a System for Reagent Selection and Library Enumeration, Profiling, and Design. *Journal of Chemical Information and Computer Sciences*, 39(6):1161–1172, 1999.

[69] A. R. Leach and V. J. Gillet. *An introduction to chemoinformatics*. Springer, 2003.

[70] A. R. Leach, B. K. Shoichet, and C. E. Peishoff. Docking and Scoring. *Journal of Medicinal Chemistry*, 49(20):5851–5855, 2006.

[71] C. Lipinski and A. Hopkins. Navigating chemical space for biology and medicine. *Nature*, 432:855–861, 2004.

[72] C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*, 23(1-3):3–25, 1997.

[73] B. Luo and E.R. Hancock. Structural graph matching using the EM algorithm and singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1120–1136, 2001.

[74] B. Luo, R. C. Wilson, and E. R. Hancock. Spectral embedding of graphs. *Pattern Recognition*, 36(10):2213–2230, 2003.

[75] B. Luo, R. C. Wilson, and E. R. Hancock. A linear generative model for graph structure. *Lecture Notes in Computer Science*, 3434:54–62, 2005.

[76] P. D. Lyne. Structure-based virtual screening: an overview. *Drug Discovery Today*, 7(20):1047–1055, 2002.

[77] S. Makino, T.J.A. Ewing, and I.D. Kuntz. DREAM++: flexible docking program for virtual combinatorial libraries. *Journal of Computer-Aided Molecular Design*, 13(5):513–532, 1999.

[78] K. V. Mardia and P. E. Jupp. *Directional statistics*. Wiley New York, 2000.

[79] G. B. McGaughey, R. P. Sheridan, C. I. Bayly, J. C. Culberson, C. Kreatsoulas, S. Lindsley, V. Maiorov, J. F. Truchon, and W. D. Cornell. Comparison of topological, shape, and docking methods in virtual screening. *Journal of Chemical Information and Modeling*, 47(4):1504–1519, 2007.

[80] J. Meiler and M. Will. Genius: a genetic algorithm for automated structure elucidation from 13C NMR spectra. *Journal of the American Chemical Society*, 124(9):1868–1870, 2002.

[81] J.B. Moon and W.J. Howe. Computer design of bioactive molecules: a method for receptor-based de novo ligand design. *Proteins: Structure, Function, and Genetics*, 11(4):314–328, 1991.

[82] M. E. Munk. Computer-based structure determination: then and now. *Journal of Chemical Information and Computer Sciences*, 38(6):997–1009, 1998.

[83] T. I. Oprea. Chemical space navigation in lead discovery. *Current Opinion in Chemical Biology*, 6(3):384–389, 2002.

[84] T. I. Oprea, A. M. Davis, S. J. Teague, and P. D. Leeson. Is there a difference between leads and drugs? A historical perspective. *Journal of Chemical Information and Computer Sciences*, 41(5):1308–1315, 2001.

[85] D. E. Patterson, R. D. Cramer, A. M. Ferguson, R. D. Clark, and L. E. Weinberger. Neighborhood behavior: A useful concept for validation of molecular diversity descriptors. *Journal of Medicinal Chemistry*, 39(16):3049–3059, 1996.

[86] X. Pennec, P. Fillard, and N. Ayache. A riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

[87] A. Porquet, J. F. L. Duval, and J. Buffle. Random Computer Generation of 3D Molecular Structures: Theoretical and Statistical Analysis. *Macromolecular Theory and Simulations*, 15(2):147–162, 2006.

[88] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. A fast flexible docking method using an incremental construction algorithm. *Journal of Molecular Biology*, 261(3):470–489, 1996.

[89] D. C. Rees, M. Congreve, C. W. Murray, and R. Carr. Fragment-based lead discovery. *Nature Reviews Drug Discovery*, 3(8):660–672, 2004.

[90] K. Riesen, M. Neuhaus, and H. Bunke. Graph Embedding in Vector Spaces by Means Of Prototype Selection. *Lecture Notes in Computer Science*, 4538:383–393, 2007.

[91] D.C. Roe and I.D. Kuntz. BUILDER v. 2: Improving the chemistry of a de novo design strategy. *Journal of Computer-Aided Molecular Design*, 9(3):269–282, 1995.

[92] S.H. Rotstein and M.A. Murcko. GenStar: a method for de novo drug design. *Journal of Computer-Aided Molecular Design*, 7(1):23–43, 1993.

[93] S.H. Rotstein and M.A. Murcko. GroupBuild: a fragment-based method for de novo drug design. *Journal of Medicinal Chemistry*, 36(12):1700–1710, 1993.

[94] N. Salim, J. Holliday, and P. Willett. Combination of fingerprint-based similarity coefficients using data fusion. *Journal of Chemical Information and Computer Sciences*, 43(2):435–442, 2003.

[95] A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13(3):353–362, 1983.

[96] G. Schneider and H. J. Böhm. Virtual screening and fast automated docking methods. *Drug Discovery Today*, 7(1):64–70, 2002.

[97] G. Schneider and U. Fechner. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649–663, 2005.

[98] G. Schneider, M.L. Lee, M. Stahl, and P. Schneider. De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks. *Journal of Computer-Aided Molecular Design*, 14(5):487–494, 2000.

[99] G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two images. *Proceedings: Biological Sciences*, 244:21–26, 1991.

[100] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[101] A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker. Indexing using a spectral coding of topological structure. In *IEEE Conference on Computer Vision and Pattern Recognition*, number 491–497, 1999.

[102] A. Torsello. An Importance Sampling Approach to Learning Structural Representations of Shape. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, 23-28 June 2008.

[103] A. Torsello and D. L. Dowe. Learning a Generative Model for Structural Representations. *Lecture Notes in Artifical Intelligence*, 5360:573–583, 2008.

[104] A. Torsello and E. R. Hancock. Learning Mixtures of Tree-Unions by Minimizing Description Length. *Lecture Notes in Computer Science*, 2683:130–146, 2003.

[105] G.V. Trunk. A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(3):306–307, 1979.

[106] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):695–703, 1988.

[107] B. J. Van Wyk and M. A. Van Wyk. A POCS-Based Graph Matching Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1526–1530, 2004.

[108] C. M. Venkatachalam, X. Jiang, T. Oldfield, and M. Waldman. LigandFit: a novel method for the shape-directed rapid docking of ligands to protein active sites. *Journal of Molecular Graphics and Modelling*, 21(4):289–307, 2003.

[109] J. J. Verbeek, N. Vlassis, and B. Krose. Efficient Greedy Learning of Gaussian Mixture Models. *Neural Computation*, 15(2):469–485, 2003.

[110] H.M. Vinkers, M.R. de Jonge, F.F.D. Daeyaert, J. Heeres, L.M.H. Koymans, J.H. van Lenthe, P.J. Lewi, H. Timmerman, K. Van Aken, and P.A.J. Janssen. Synopsis: synthesize and optimize system in silico. *Journal of Medicinal Chemistry*, 46(13):2765–2773, 2003.

[111] R. Wang, X. Fang, Y. Lu, and S. Wang. The PDBbind database: collection of binding affinities for protein- ligand complexes with known three-dimensional structures. *Journal of Medicinal Chemistry*, 47(12):2977–2980, 2004.

[112] R. Wang, Y. Gao, and L. Lai. LigBuilder: a multi-purpose program for structure-based drug design. *Journal of Molecular Modeling*, 6(7):498–516, 2000.

[113] Y. K. Wang, K. C. Fan, and J. T. Horng. Genetic-based search for error-correcting graph isomorphism. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 27(4):588–597, 1997.

[114] D. Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

[115] D. White and R. C. Wilson. Mixing Spectral Representations of Graphs. In *Proceedings of the 18th International Conference on Pattern Recognition*, volume 4, pages 140–144, 2006.

[116] D. White and R. C. Wilson. Spectral Generative Models for Graphs. In *Proceedings of the 14th International Conference on Image Analysis and Processing*, pages 35–42, 2007.

[117] D. White and R. C. Wilson. Parts based generative models for graphs. In *Proceedings of the 19th International Conference on Pattern Recognition*, pages 1–4, 2008.

[118] P. Willett. Chemoinformatics - similarity and diversity in chemical libraries. *Current Opinion in Biotechnology*, 11(1):85–88, 2000.

[119] P. Willett, J. M. Barnard, and G. M. Downs. Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, 38(6):983–996, 1998.

[120] P. Willett, V. Winterman, and D. Bawden. Implementation of nearest-neighbor searching in an online chemical structure search system. *Journal of Chemical Information and Computer Sciences*, 26(1):36–41, 1986.

[121] R. C. Wilson. Chemical structure matching using spectral features. *In Preperation*.

[122] R. C. Wilson and E. R. Hancock. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):634–648, 1997.

[123] R. C. Wilson, E. R. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1112–1124, 2005.

[124] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.

[125] B. Xiao and E. R. Hancock. Geometric characterisation of graphs. *Lecture Notes in Computer Science*, 3617:471–478, 2005.

[126] B. Xiao and E. R. Hancock. A Spectral Generative Model for Graph Structure. *Lecture Notes in Computer Science*, 4109:173–181, 2006.

[127] L. Xue, J. W. Godden, and J. Bajorath. Database searching for compounds with similar biological activity using short binary bit string representations

of molecules. *Journal of Chemical Information and Computer Sciences*, 39(5):881–886, 1999.

[128] P. Zhu and R. C. Wilson. Stability of the eigenvalues of graphs. *Lecture Notes in Computer Science*, 3691:371–378, 2005.