Comparing Graph Representations of Protein Structure for Mining Family-Specific Residue-Based Packing Motifs

Jun Huan[1], Deepak Bandyopadhyay[1], Wei Wang[1], Jack Snoeyink[1],
Jan Prins[1], Alexander Tropsha[2]
[1]Department of Computer Science, [2]The Laboratory for Molecular Modeling,
Division of Medicinal Chemistry and Natural Products, School of Pharmacy,
University of North Carolina at Chapel Hill
[1]{huan, debug, weiwang, snoeyink, prins}@cs.unc.edu, [2] tropsha@email.unc.edu

Abstract

We find recurring amino-acid residue packing patterns, or spatial motifs, that are characteristic of protein structural families, by applying a novel frequent subgraph mining algorithm to graph representations of protein three-dimensional structure. Graph nodes represent amino acids, and edges are chosen in one of three ways: first, using a threshold for contact distance between residues; second, using Delaunay tessellation; and third, using the recently developed almost-Delaunay edges.

For a set of graphs representing a protein family from the Structural Classification of Proteins (SCOP) database, subgraph mining typically identifies several hundred common subgraphs corresponding to spatial motifs that are frequently found in proteins in the family but rarely found outside of it. We find that some of the large motifs map onto known functional regions in two families explored in this study, i.e., Serine Proteases and Kinases.

1 Introduction

1.1 Spatial Motif Discovery in Proteins

Recurring substructural motifs in proteins reveal important information about protein structure and function. Common structural fragments of various sizes have fixed 3D arrangements of residues that may correspond to active sites or other functionally relevant features, such as Prosite patterns [Hofmann *et al.*, 1999]. Identifying such spatial motifs in an automated and efficient way may have a great impact on protein classification [Chakraborty & Biswas, 1999], protein function prediction [Fischer *et al.*, 1994] and protein folding [Kleywegt, 1999].

Protein structures have been modeled using graphs in many applications, including identification of active site clusters, folding clusters, aromatic clusters in relation to thermodynamic stability, and the analysis of protein-protein interaction. (See Vishveshwara *et al.* [2002] for a recent and comprehensive review on applying graph theory to protein structure analysis.) The choice of a graph representation is a key aspect of protein structure analysis. Several representations have been developed, ranging from coarse representations in which each node is a secondary structure segment [Grindley *et al.*, 1993] to fine representations in which each node is an atom [Jacobs *et al.*, 2001].

We represent a protein structure by a labeled graph. Every node of the graph represents a distinct amino acid residue in a protein and has the residue type as its label; each residue is represented by its C_α atom. We distinguish two types of edges that connect residues: *bond edges* connect pairs of residues that are adjacent in the primary sequence, and *proximity edges* connect pairs of (non-bonded) residues identified as spatial neighbors by the following three representations.

In all representations, we only consider edges no longer than a threshold δ, usually chosen between 6 and 10 Å. For the first representation, the *contact distance graph (**CD**)*, this is the only criterion. The second representation uses the edges from the Delaunay tessellation (**DT**), applied to the coordinates of the C_α atoms. Proximity edges thus satisfy an empty sphere property [Delaunay, 1934], which implies that they join residues that are neighbors in 3D space. The Delaunay tessellation has been used to analyze protein packing [Richards, 1974; Tsai *et al.*, 1999] and structure [Liang *et al.*, 1998; Singh *et al.*, 1996; Wernisch *et al.*, 1999; Wako & Yamato, 1998; Tropsha *et al.*, 2003]. The third representation, derived from the recently-defined almost-Delaunay edges [Bandyopadhyay & Snoeyink, 2004], expands the set of Delaunay edges to account for perturbation or motion of point coordinates, controlled by a parameter ϵ. Thus, we actually define a family of graphs, $\mathbf{AD}(\epsilon)$, that interpolates between the DT and CD representations so that $DT \subseteq AD(\epsilon) \subseteq CD$ for all $\epsilon \geq 0$. The AD representation allows us to reduce the number of edges in a CD graph without losing common patterns due to imprecise positioning of the chosen points of residues.

In this paper we expand the scope of our earlier study of graph representations for subgraph mining in proteins [Huan *et al.*, 2004a]. We apply a generalized form of frequent subgraph mining to identify subgraphs common to proteins of a given structural family found in the SCOP database [Murzin *et al.*, 1995]. Given a group of proteins with a fixed support threshold, general frequent subgraph mining always finds a superset of subgraphs inclusive of those identified by the means of either induced [Huan *et al.*, 2003] or coherent [Huan *et al.*, 2004d] subgraph mining and yet it is more robust for local connectivity variations. The residue packing patterns that correspond to frequent subgraphs can be regarded as motifs specific to protein families, which can be explored for their role in protein stability and function. We evaluate the influence of the different graph representations on the performance of subgraph mining. We explore the nature and biological significance of *fingerprints*, patterns that are highly specific to a family and rarely seen in the rest of the Protein Data Bank (PDB) [Berman *et al.*, 00].

We find that AD graphs significantly reduce the number of edges in the graph representation, yet the features extracted from such graphs have approximately equivalent biological interpretation as those extracted from CD graphs. Performance of fingerprint identification depends on the sequence and structural variability within the family. In highly homogeneous families such as Serine Proteases, DT graphs are adequate for finding fingerprints, AD graphs are better but slower, and the computation on CD graphs could not be completed after several days. In structurally diverse protein families such as Protein Kinases, the DT graphs do not find any significant fingerprints while AD and CD graphs do, and even calculations using CD graphs can be completed in a reasonable time.

The remainder of the paper is organized as follows. Section 1.2 presents the related work. Section 2 presents definitions for the subgraph isomorphism and discusses the details of different graph representations of proteins. Section 3 presents the data structure and the algorithm for subgraph mining and Section 4 presents the results of our study of three protein families from the SCOP database, including eukaryotic and prokaryotic serine proteases, and protein kinases. Finally, Section 5 includes conclusions and a brief discussion of future studies.

1.2 Related Work

To find patterns from graphs is a challenging task. Several algorithms have been developed recently in the data mining community to find all frequent subgraphs of a group of labeled graphs [Kuramochi & Karypis, 2001; Yan & Han, 2002; Huan *et al.*, 2003, 2004b]. These algorithms can be roughly classified into two types: The first type uses a level-wise search scheme to enumerate the recurring subgraphs [Inokuchi *et al.*, 2000; Kuramochi & Karypis, 2001] and the second type uses a depth-first enumeration for frequent subgraphs, which usually has better memory utilization and therefore better performance [Yan & Han, 2002; Borgelt & Berhold, 2002; Huan *et al.*, 2003, 2004b]. In fact, depth-first search can outperform FSG [Kuramochi & Karypis, 2001], the current state-of-the-art level-wise search scheme, by an order of magnitude [Yan & Han, 2002].

Graphs have long been used to study organic molecules [Dehaspe *et al.*, 1998; Borgelt & Berhold, 2002] and macromolecules such as nucleic acids [Klein, 1998; Wang *et al.*, 1998] and proteins [Milik *et al.*, 2003; Mitchell *et al.*, 1990; Singh & Brutlag, 1997]. Non-graph based approaches have been reported as well. Several research groups have addressed the problem of finding spatial motifs by using techniques from computational geometry and computer vision. If a protein is represented as a set of points in $\mathcal{R}^3$, then the problem of spatial motif finding for pairs of molecules may be modeled as the Largest Common Pointset (LCP) problem: identifying the largest common subset of two sets of points [Akutsu *et al.*, 1997]. A number of variations have been explored, which include approximate LCP [Chakraborty & Biswas, 1999; Indyk *et al.*, 1999] and LCP-α: identifying a subset that approximates the LCP with a factor α characterizing the degree of approximation [Finn *et al.*, 1997].

The TRILOGY program [Bradley *et al.*, 2002] looks for patterns among the conserved residues within a family, and is able to find matches that correspond to functional motifs. It handles sequence matches as regular expressions of similar residues separated by a fixed range of sequence gaps and structure matches by calculating the distances between C_αs and angles between paired $C_\alpha - C_\beta$ vectors. TRILOGY constructs longer matches from smaller ones, beginning with triples as seeds. Our search method finds spatial packing patterns in families with low sequence homology or varying sequence separation between packed residues, and thus is complementary to the TRILOGY method.

2 Methodology

2.1 Frequent Subgraphs

We define a *labeled graph* G as a five element tuple $G = (V, E, \Sigma_V, \Sigma_E, \lambda)$ where V is a set of vertices or nodes and $E \subseteq V \times V$ is a set of undirected edges. Σ_V and Σ_E are disjoint sets of vertex and edge labels, respectively, and λ is a function that assigns labels to vertices and edges: $V \rightarrow \Sigma_V$ and $E \rightarrow \Sigma_E$. We assume that a total ordering is defined on the labels in $\Sigma_V \cup \Sigma_E$.

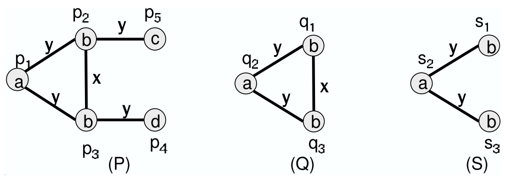

Figure 1. A graph database $\mathcal{G}$ of three labeled graphs. The mapping $q_1 \rightarrow p_2$, $q_2 \rightarrow p_1$, and $q_3 \rightarrow p_3$ demonstrates that graph Q is isomorphic to a subgraph of P, and so we say that Q occurs in P. Similarly graph S occurs in both graph P and graph Q.

$G' = (V', E')$ is a *subgraph* of G if vertices $V' \subseteq V$, and edges $E' \subseteq (E \cap (V' \times V'))$, i.e. E' is a subset of the edges of G that join vertices in V'. Subgraph G' is an *induced subgraph* of G if $E' = (E \cap (V \times V))$, i.e. if E' includes *all* edges in G that join vertices in V'.

A fundamental part of our method is to find an occurrence of a graph H within another graph G. To make this more precise, we say that H *occurs in* G if we can find an *isomorphism* between graph $H = (V_H, E_H, \Sigma_V, \Sigma_E, \lambda_H)$ and some subgraph of $G = (V_G, E_G, \Sigma_V, \Sigma_E, \lambda_G)$. An isomorphism from H to the subgraph of G defined by vertices $V \subseteq V_G$ is a bijection between vertices $f\colon V_H \rightarrow V$ that preserves edges and labels:

$$\begin{aligned} \forall u \in V_H, \qquad & \lambda_H(u) = \lambda_G(f(u)), \text{ and} \\ \forall (u,v) \in E_H, \qquad & (f(u), f(v)) \in E_G \wedge \\ & \qquad \lambda_H(u,v) = \lambda_G(f(u), f(v)) \end{aligned}$$

This definition is illustrated in Figure 1.

In the previous work [Huan *et al.*, 2003, 2004a,d] we have used a more restrictive definition of occurrence, in which a graph H occurs in graph G only if H is isomorphic to an *induced subgraph* of G. In Figure 1, graph Q is isomorphic to an induced subgraph of P, but graph

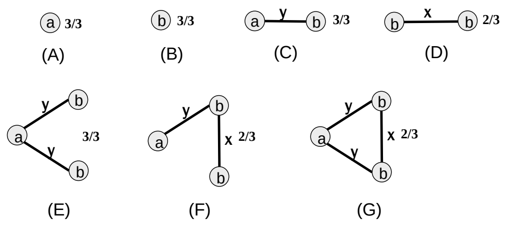

Figure 2. All (non-empty and connected) frequent subgraphs with support $\geq 2/3$ in $\mathcal{G}$ from Figure 1. The actual support value is given with the frequent subgraphs. All are subgraphs of the one maximal frequent graph, which is the graph G at the lower right.

S is not isomorphic to an induced subgraph of P. Under the definitions used in this paper, both Q and S are said to occur in P. The basic problem of testing subgraph isomorphism or induced subgraph isomorphism is NP complete, although the most intractable cases arise in large and dense graphs with few distinguishing labels. The lower density representations investigated in this paper, together with improvements in our algorithms, aim to obtain tractable methods for mining fully general subgraphs from sets of graphs representing families of proteins.

Given a *graph database* $\mathcal{G}$, which is a set of graphs, we define the *support* of a graph H as the fraction of graphs in $\mathcal{G}$ in which H occurs. When the graph database $\mathcal{G}$ is understood, we let sup_H denote this fraction. We choose a threshold $0 < \sigma \leq 1$, and define H to be *frequent* if and only if $sup_H \geq \sigma$. Note that while H may occur many times within a single graph G, for the purposes of support, only one occurrence is counted per graph.

The problem of *Frequent Subgraph Mining* is to identify all frequent subgraphs for a graph database $\mathcal{G}$. Figure 2 shows all frequent connected subgraphs with $\sigma = 2/3$ in the graph database of Figure 1.

Since any subgraph of a frequent graph H is also frequent, we can confine our search to find *maximal frequent graphs* [Huan *et al.*, 2004b], which are not subgraphs of any other frequent graph. If we increase the support threshold to $\sigma > 2/3$, then the graph E in Figure 2 becomes the maximal frequent subgraph, while using the more restrictive induced subgraph criterion can only yield the graph C as a maximal induced frequent subgraph. Thus we believe that the definition of occurrence based on simple subgraph isomorphism is more robust under variations in local connectivity.

2.2 Building Protein Graphs

The vertices of our graph represent the amino acid residues in the protein. Since the protein backbone defines the overall protein conformation, we have chosen to use the C_α atom to represent the residue. Other choices are possible including the C_β [Lovell *et al.*, 2003] or the side chain centroid [Cammer *et al.*, 2002]. Two vertices are connected by a *bond edge* when the residues they represent are consecutive in the primary sequence. Starting from this simplified protein model, we compute proximity edges using three different approaches.

In the first representation, we connect two vertices by an edge if the distance between them does not exceed a threshold δ. Since we are interested in neighboring residues within a physical interaction radius, we chose δ to vary over values ranging from 6.5–9.5 Å. As stated in the Section 1, we refer to this distance-threshold dependent graph as the CD graph.

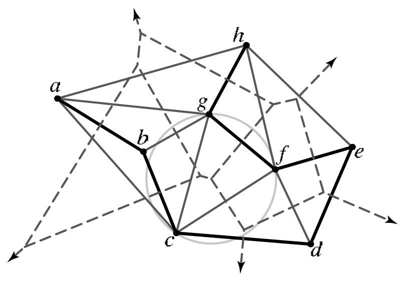

Figure 3. Examples of a Voronoi diagram and its dual Delaunay Tessellation for 2D points [a–f]

In the second representation, proximity edges come from the Delaunay tessellation [Delaunay, 1934], which is defined for a finite set of points by an *empty sphere property*: A pair of points is joined by an edge if and only if there exists an empty sphere with those two points on its boundary. The Delaunay captures neighbor relationships in the sense that there is a point in space that has the two chosen points as closest neighbors. (The Delaunay is dual to the Voronoi diagram—two points are joined by an edge in the Delaunay iff their Voronoi cells share a common face. Figure 3 illustrates the Delaunay in 2D with solid lines, and the dual Voronoi with dashed.) We removed edges longer than δ in a postprocessing step. We refer to this representation of protein structure as the DT graph.

The definition of the Delaunay tessellation depends on the precise coordinate values given to its points, but we know that these values are not exact in proteins due to measurement imprecision and atomic motions. Thus, the third representation uses the almost-Delaunay edges [Bandyopadhyay & Snoeyink, 2004] that are based on a generalization of the empty sphere property: a pair of points p and q is joined by an almost-Delaunay edge with parameter ϵ, or $AD(\epsilon)$, if by perturbing all points by at most ϵ, the chosen p and q can be made to lie on an empty sphere. The precise parameter value for each edge of the contact distance graph can be computed by an algorithm that is much like the roundness algorithms from the computer-aided design (CAD) field of computational metrology. They look for a shell of width 2ϵ, formed by concentric spheres, so that p and q are on the outer sphere, and all points are outside the inner sphere. Code is available from http://www.cs.unc.edu/~debug/papers/AlmDel, or see Bandyopadhyay & Snoeyink [2004] for algorithmic details.

All Delaunay edges are in $AD(0)$, and $AD(\epsilon) \subseteq AD(\epsilon')$ for $\epsilon \leq \epsilon'$. Therefore, the almost-Delaunay edges are a superset of the Delaunay edges, where the size of the set is controlled by the parameter ϵ. Various values of ϵ correspond to different allowed perturbations or motions. 0.1–0.25 Å would model decimal inaccuracies in the PDB coordinates or small vibrations, and 0.5–0.75 Å would model perturbations due to coarser motions. Thus, a protein graph constructed with the almost-Delaunay edges of parameter ϵ is called the AD(ϵ) graph.

3 Frequent Subgraph Mining

In this section, we outline the framework we use to find frequent subgraphs in a protein graph database. Since these graphs encode aspects of the spatial structure of the proteins, the frequent subgraphs correspond to spatial motifs common to the proteins in the database.

3.1 Canonical Adjacency Matrix of Labeled Graphs

We represent each graph by an adjacency matrix M such that every diagonal entry of M is filled with the label of the corresponding node and every off-diagonal entry is filled with the label of the corresponding edge, or zero if there is no edge. This is slightly different from the standard adjacency matrix representation for unlabeled graphs [Cormen *et al.*, 2001].

Given an $n \times n$ adjacency matrix M of a graph G with n nodes, we define the *code* of M, denoted by $code(M)$, as the sequence of lower triangular entries of M (including the entries at diagonal) in the order: $M_{1,1}M_{2,1}M_{2,2}...M_{n,1}M_{n,2}...M_{n,n-1}M_{n,n}$ where $M_{i,j}$ represents the entry at the ith row and jth column in M. Since our edges are undirected, we are concerned only

$$\begin{array}{|c|c|c|c|c|}\hline a & & & & \\\hline y & b & & & \\\hline y & x & b & & \\\hline 0 & y & 0 & c & \\\hline 0 & 0 & y & 0 & d \\\hline\end{array}_{M_1} > \begin{array}{|c|c|c|c|c|}\hline a & & & & \\\hline y & b & & & \\\hline y & x & b & & \\\hline 0 & 0 & y & d & \\\hline 0 & y & 0 & 0 & c \\\hline\end{array}_{M_2} > \begin{array}{|c|c|c|c|c|}\hline b & & & & \\\hline x & b & & & \\\hline y & 0 & d & & \\\hline 0 & y & 0 & c & \\\hline y & y & 0 & 0 & a \\\hline\end{array}_{M_3}$$

Figure 4. Three adjacency matrices for the graph P in Figure 1. Using the following ordering of label values: $a > b > c > d > x > y > 0$, we have $code(M_1)$ =“*aybyxb0yxc00y0d*” $\geq code(M_2)$ = “*aybyxb00yd0yx0c*” $\geq code(M_3)$ = “*bxby0dxy0cyy00a*”. Hence M_1 is the CAM for graph P.

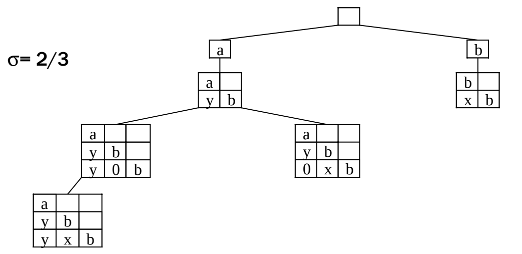

Figure 5. The CAM Tree for the frequent subgraphs in the graph P shown in Figure 1.

with the lower diagonal entries of M. Figure 4 shows examples of adjacency matrices and codes for the labeled graph P shown in Figure 1.

We use lexicographic order of sequences to define a total order over adjacency matrix codes. Given a graph G, its *canonical form* is the maximal code among all its possible codes. The adjacency matrix M which produces the canonical form is denoted as G's *canonical adjacency matrix* (CAM). For example, the adjacency matrix M_1 shown in Figure 4 is the CAM of the graph P from Figure 1, and $code(M_1)$ is the canonical form of the graph.

For a matrix N, we define the *proper maximal submatrix* (*submatrix* for short) as the matrix M obtained by removing the last row and column from N.

One valuable property of the canonical form we are using (compared to the forms of Inokuchi *et al.* [2000] and Kuramochi & Karypis [2001] is that, given a graph database $\mathcal{G}$, all frequent subgraphs (represented by their CAMs) can be organized into a rooted tree. This tree is referred to as the *CAM Tree* of $\mathcal{G}$ and is formally described as follows:

(i) The root of the tree is the empty matrix;

(ii) Each node in the tree is a distinct frequent connected subgraph in $\mathcal{G}$, represented by its CAM;

(iii) The parent of a given non-root node (with CAM M) is the graph represented by M's proper maximal submatrix.

3.2 Algorithm Overview

The search algorithm shown below is used to systematically traverse the CAM tree while maintaining all frequent subgraphs. Additional details can be found in Huan *et al.* [2003].

A number of recent improvements have been incorporated in the graph mining algorithm used for the experiments reported here. These enabled us to solve problems that were intractable in our preliminary study [Huan *et al.*, 2004a]. As a result, however, the running times reported and set of subgraphs found here are not directly comparable to the results reported there.

There are two significant changes. First, the definition of what it means for a graph H to occur in G has been relaxed as described in Section 2.1. Second, *maximal frequent subgraphs* [Huan *et al.*, 2004c,b] are mined instead of coherent subgraphs [Huan *et al.*, 2004d]. The net effect of these two changes is that every subgraph reported by the previous algorithms is a subgraph of some maximal frequent subgraph reported by the current algorithm. The overview of the algorithm is presented in Table 1 and 2.

```
FFSM($\mathcal{G}$):
begin
1. $S \leftarrow$ { the CAMs of the frequent nodes in $\mathcal{G}$}
2. $P \leftarrow$ { the CAMs of the frequent edges in $\mathcal{G}$}
3. FFSM-Explore($P, S$)
end
```

Table 1. An algorithm for finding frequent subgraphs from a group of graphs $\mathcal{G}$.

```
FFSM-Explore ($P, S$)
begin
1. for { $X \in P$} do
2.   $S \leftarrow S \cup \{X\}$
3.   $C \leftarrow$ { all matrices $M$ | $X$ is submatrix of $M$}
4.   $C \leftarrow$ { $y$| $y \in C$, $y$ is a frequent and maximal CAM}
5.   FFSM-Explore($C, S$)
6. end for
end
```

Table 2. Enumerate all frequent subgraphs recursively.

3.3 Algorithm Details

In the following discussion, we present two important details about the FFSM algorithm: (1) how to compute the CAM of a given subgraph and (2) how to compute whether a subgraph is frequent or not. With these, we guarantee that subgraphs reported by our algorithm are correct in that their frequencies are validated; they are also non-redundant since only subgraphs in their CAM form are reported. Finally, the completeness of the result (that *all* frequent subgraph are identified by our algorithm) is guaranteed by using the CAM tree since all subgraphs have one unique position in the CAM tree. Because we are primarily concerned with the application, we substitute algorithm proofs with intuitive description of the algorithms. Interested readers are referred to [Huan *et al.*, 2004c] for further information about formal correctness proof.

We introduce a greedy algorithm to compute the CAM of a subgraph G. The algorithm works by picking up maximal labeled nodes in G as a group of single-node matrices. It iteratively grows those matrices by attaching one additional node to each of them in all possible ways. The resulting group of matrices are inspected one by one. Only those which are maximal among their peers are selected for the next iteration. This computation is determined to converge to the CAM of the input subgraph G.

```
Compute-CAM ($G$)
begin
1. $S \leftarrow \{u|\ u$ is a single-node matrix with the maximal
   node label in $G\}$
2. do
3.   $Q \leftarrow \{y|\ y$ is a adjacency matrix of a subgraph of $G$ by
             including one additional node to a matrix $x \in S\}$
4.   $S \leftarrow \{y|\ y \in Q$ and $y$ is maximal in $S\}$
5. until $S$ contains adjacency matrix of $G$
end
```

Table 3. An algorithm to compute the canonical adjacency matrix of a graph G

In order to compute the support value of a subgraph and decide whether it is frequent or not, we need to invoke a subgraph isomorphism test for the subgraph against each graph in a group of input graphs. This procedure is generally expensive since subgraph isomorphism test is an NP-complete problem. Fortunately, we can empirically speed up the computation by (1) using a depth first enumeration of the CAM tree and (2) keep embeddings of each subgraph we visited. To that end, we need the following definition:
Given an arbitrary $n \times n$ CAM A and a labeled graph $G = (V, E, \Sigma_V, \Sigma_E, \lambda)$, a vertex list $L = u_1, u_2, \ldots, u_n \subset V$ is an **embedding** of A in G if and only if:

- $\forall\, i, (a_{i,i} = \lambda(u_i))$;
- $\forall\, i, j(a_{i,j} \neq 0 \Rightarrow a_{i,j} = \lambda(u_i, u_j))$;

where $0 < j < i \leq n$ and $a_{i,j}$ represents the element at the ith row and the jth column in A .

Using an embedding list, we can check subgraph isomorphism incrementally at successive levels of the CAM tree and avoid repeating subgraph isomorphism tests performed previously. Empirically, this optimization speeds up the computation significantly [Huan *et al.*, 2003].

4 Results

4.1 Experimental Setup

We applied the subgraph mining procedure to two datasets from the SCOP database [Murzin *et al.*, 1995]. The first dataset is a group of serine proteases from SCOP superfamily “Trypsin-like serine proteases,” which is referred to as the **SP** dataset. We used both prokaryotic and eukaryotic serine proteases in the superfamily since they share the same catalytic site geometry by evolutionary convergence, but differ in the surrounding residues. One interesting question for the SP dataset is whether we can find subgraphs including known active site residues in SP.

The second dataset is a group of sequence-diverse protein kinases from the SCOP family “Protein kinases, catalytic subunit”. This group is referred to as the **Kinase** dataset hereafter. For the Kinase dataset, we are interested in whether we can find biologically significant motifs using our frequent subgraph mining procedure.

To represent all the structures in the PDB, we selected around 4600 non-redundant structures using the *culled PDB* list (http://www.fccc.edu/research/labs/dunbrack/pisces/culledpdb.html) with no more than 60% pair-wise sequence similarity, in order to remove highly homologous proteins. We retrieved proteins with resolution $\leq$ 3.0 and R factor $\leq$ 1.0 to ensure that we use high quality x-ray structures. The SP and Kinase proteins selected for the analysis were confined to the non-redundant structures in the same culled PDB list and therefore had pair-wise sequence identity less than 60%. Our final list of the SP dataset contained 35 ESP proteins and 8 PSP proteins and the Kinase dataset contained 29 proteins. We obtained the coordinates for all proteins from the Protein Data Bank (PDB).

All calculations were run on a single processor, 2.8GHz Pentium PC with 2GB memory, operating on

RedHat Linux 7.3. The frequent subgraph mining algorithm was implemented in C++ and compiled using g++ with O3 optimization. We calculated the Delaunay tessellation for a set of coordinates using Quickhull [Barber *et al.*, 1996]. The almost-Delaunay computation [Bandyopadhyay & Snoeyink, 2004] used the code available at http://www.cs.unc.edu/~debug/papers/AlmDel/.

4.2 Comparing Three Graph Representations

Three graph representations, CD, DT and AD, were constructed for each protein. Figure 6 shows the average number of edges per vertex as a function of the distance threshold for the Kinase dataset. For small distance thresholds, the graphs are nearly the same. As the distance threshold grows, the number of edges in CD graphs grows as a third power of the distance, while it remains almost constant in DT graphs. The number of $AD(\epsilon)$ edges as a function of the distance interpolates between the CD and DT representations.

Using both the SP and Kinase datasets, we have compared the performance of the subgraph mining algorithm using the three graph representations. Note that we used a general subgraph mining algorithm in the place of the induced subgraph mining algorithm in our previous publication [Huan *et al.*, 2004a]. Our current approach finds a superset of sungraphs identified by the previous published method. Figure 7 shows the CPU time and the number of found patterns per second as a function of the distance threshold for the SP and Kinase datasets. We chose to plot the rate of discovering patterns since in practice one has to compare the output of the three representations after they have been allowed to run for a fixed time.

As expected, the running times for the three graph representations are similar when the graph representations are similar (at the distance threshold around 6.5 Å). As the distance threshold increases, the three graph representations differ, and the performance gaps between them increase. CD graphs reach a prohibitively long running time at the distance threshold 9.5 Å for Kinases. The SP dataset is more computationally challenging than the Kinase dataset since proteins in SP are more conserved at both the sequence and structure levels, and hence are expected to have more and larger common subgraphs in their graph representations. The CD, AD(0.25), and AD(0.5) graphs reach a prohibitively long running time at the distance threshold of 8.5 Å.

4.3 Fingerprint Identification

In this section, we investigate finding common and characteristic features of a protein family. For a group

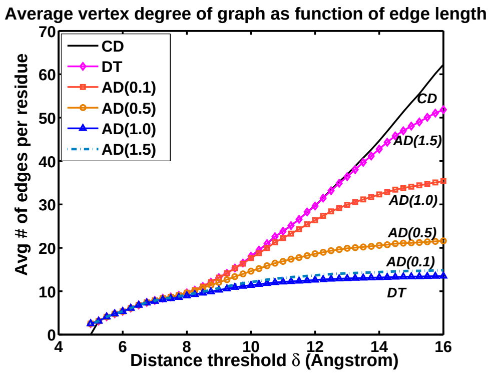

Figure 6. Average vertex degree as a function of the distance threshold for the three types of graphs representing the cytoplasmic domain of the Type I Tgf- Receptor in complex with Fkbp12 (1bc6).

P of proteins, represented by label graphs, we define a *fingerprint* of P as a subgraph whose support value in P is at least a certain high threshold (*minSupport* or σ) whereas its support in the whole PDB database (or a large non-redundant representative subset) does not exceed a low upper-bound (*maxBackground* or λ). In all our experimental studies, we used σ = 90% and λ= 5%.

4.3.1 Serine Proteases

We have identified fingerprints in the SP dataset using the AD(0.1) graph representation with the distance threshold of 8.5. AD(0.1) was chosen because the corresponding graph is relatively sparse leading to a reasonable running time with this dataset, as explained in Section 4.2. We have identified a total of 5569 fingerprints (sizes range from 2 to 13 residues); 19 of them had background frequency less than 1%, which implies that they appear in no more than 46 proteins in the 4600 non-redundant structures (excluding serine proteases). One of these fingerprints with the largest number of residues included the ASP-HIS-SER catalytic triad. This fingerprint is shown in Figure 8 in the form of corresponding subgraph as well as its occurrence in the structure of Kallikrein 6 (PDB code 1lo6).

Table 4 summarizes the background frequency distribution of the 5569 subgraphs. 830 subgraphs (sizes range from 2 to 11 residues) are identified based on the DT representation with the same distance threshold and none of them have background frequency less than 1%. Experiments on CD graphs could not complete after three days of calculations.

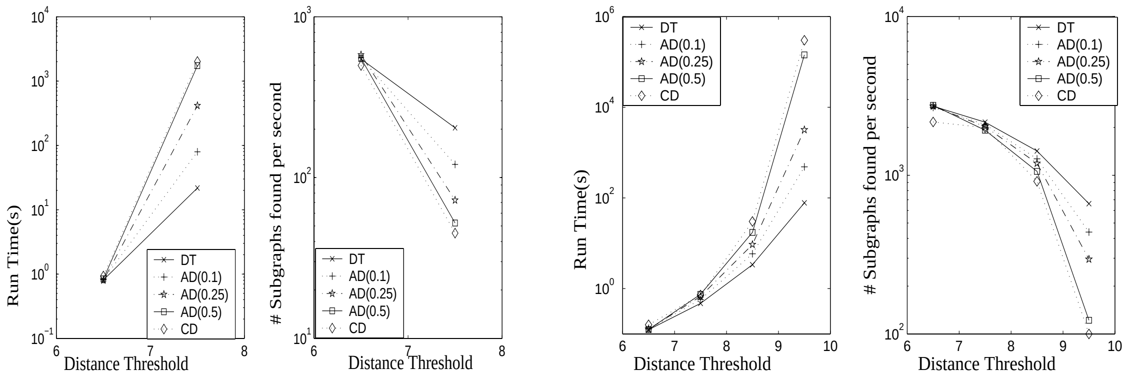

Figure 7. The total running time of the subgraph mining algorithm (left) and the average number of identified patterns per second (right) using the SP dataset (the left two figures) and the Kinase dataset (the right two figures). The support threshold is set to 90% across all experiments. The running time and average number of identified patterns per second for CD graphs at distance 9.5 Å are extrapolated from previous data because it takes too long to complete the run.

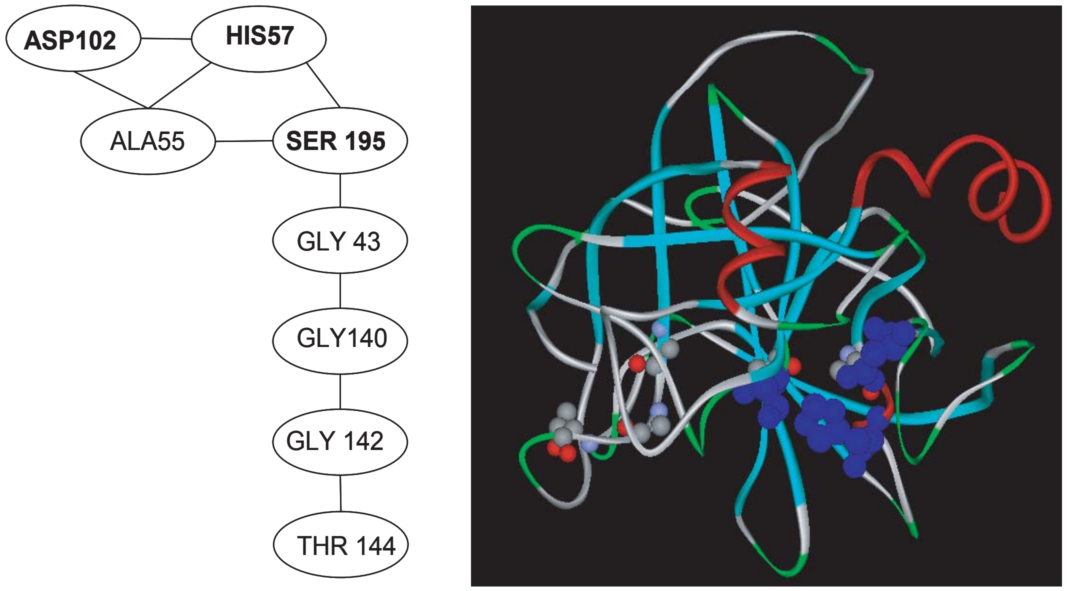

Figure 8. One of the fingerprints in the SP dataset contains the catalytic triad. Left: graph representation. Right: mapping of this motif onto the backbone of Kallikrein 6 (1lo6). The catalytic residues (His57, Asp102, and Ser 195) are highlighted in the graph representation and color coded by blue in the protein structure. The rest of the five residues are shown by ball and stick model.

4.3.2 Protein Kinases

Using the distance threshold 8.5 Å and the AD(0.5) representation, we obtained 30 fingerprints which appear in at least 90% of the members of the Kinase dataset and less than 5% of background structures. We use a large perturbation for AD graphs, since the structural diversity of kinases implies fewer common subgraphs, and computational efficiency is not a bottleneck as it was for the SP dataset. This calculation took less than one minute to complete.

Table 4 also summarizes the background frequency distribution of the 30 subgraphs. 37 subgraphs were obtained using CD graphs, and 4 were obtained using DT graphs with the same distance threshold. Across all representations, none of the fingerprints had background frequency less than 1%.

From the results, we find that the DT fingerprints of the kinase family are at most 4 residues long, the

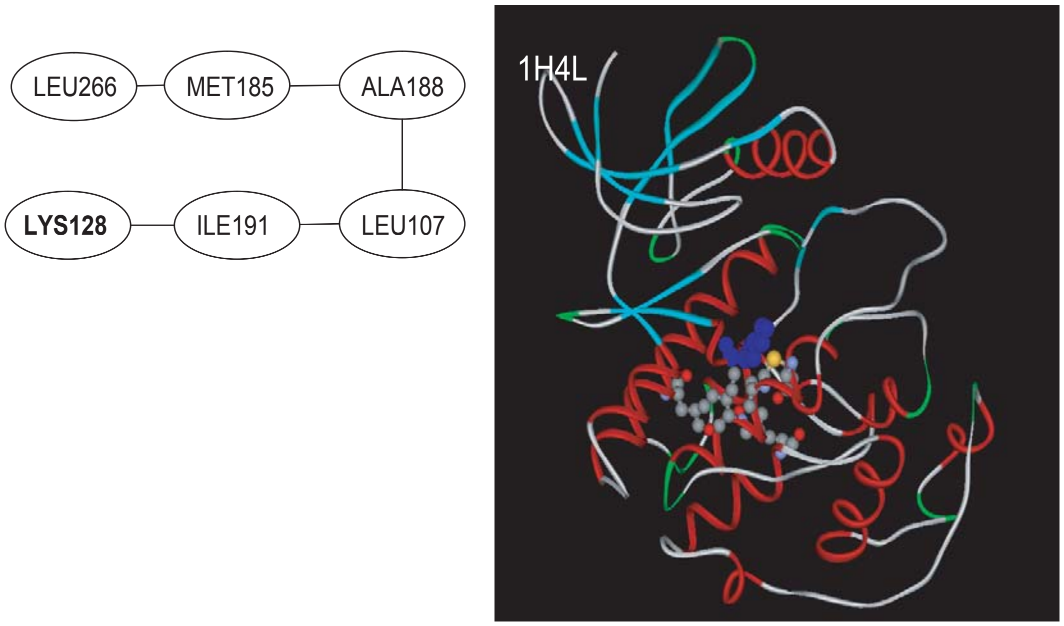

Figure 9. Large subgraph motif found in more than 90% of the Protein Kinase family members that includes a catalytic residue. Left: graph representations. All edges are proximity edges. Right: mapping of this motif onto the backbone of Cell Division Kinase 5 (1h4l). The motif includes the invariant catalytic residue Lys128, highlighted in the graph representation and colored blue in the protein structure, and neighboring hydrophobic residues that contact the ligand.

$\lambda(\%)$	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
SP	0.0	0.3	1.4	3.4	6.0	9.6	16.1	31.3	59.3	100
$Kinase$	0.0	0.0	20.0	36.7	50.0	66.6	73.3	86.7	93.3	100

Table 4. The background occurrence of features mined from the SP and Kinase dataset using the AD graphs. Entries in the table are the fraction of features which have background frequency less than λ.

AD(0.5) fingerprints are 3 to 7 residues long, and the CD fingerprints are at most 8 residues long. The larger fingerprints could carry biologically significant information about conserved residues which are separated along the sequence within protein families that is difficult to derive from sequence comparisons. For example, the largest AD(0.5) fingerprint that includes a catalytic lysine (shown in Figure 9 for CDK5, PDB ID 1h4l) has six residues (Leu-Met-Ala-Leu-Ile-Lys), no two of which are adjacent in the sequence, and the sequence gaps between residues are not conserved. The fingerprint starts from the C-terminal hydrophobic pocket and ends at an invariant catalytic lysine. The second catalytic residue, Asp, is not contained in a fingerprint in most kinases, and no fingerprint contains both catalytic residues. The fingerprint mentioned above is often repeated; in CDK5 the second copy includes a non-conserved lysine in the N-terminal domain. This highlights structural similarities between the arrangement of residues in these two regions.

4.3.3 Do Fingerprints Superimpose?

Since fingerprints are derived from graphs which are formally dimensionless, an interesting questions arises as to whether these fingerprints are geometrically conserved, i.e., if the identical fingerprints found in different members of the family could be structurally superimposed. We have investigated this question as applied to the fingerprints in the active sites of SP and Kinases. Consider the largest fingerprint of SP, which is shown in Figure 8 and contains an Asp-His-Ser catalytic triad.

Figure 10 shows the least-squares superposition of this fingerprint derived from 30 SP proteins. We also

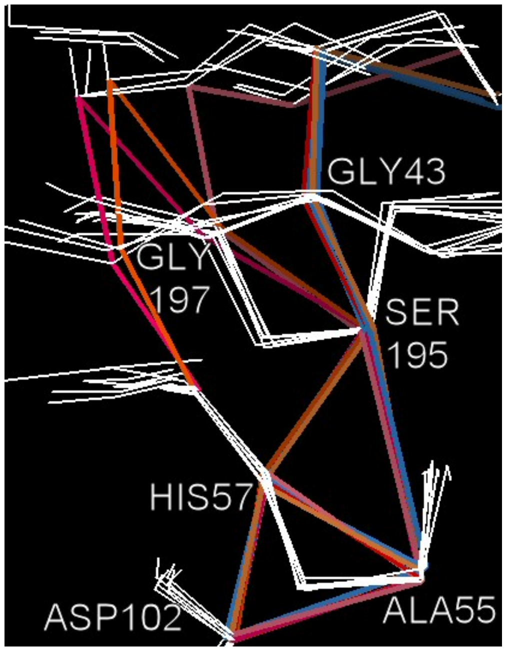

Figure 10. Least-squares superposition of the largest fingerprint that contains the whole active site in 30 proteins from our dataset of 35 eukaryotic and 8 prokaryotic serine proteases. Maximum RMSD is 0.5 Å RMSD in the first four residues (Asp-His-Ala-Ser). Only 7 serine proteases (ESP: 1lo6A,1eq4A,1fiwA,1eaxA; PSP: 1qq4A, 1sgpE, 1hpgA) are shown superposed, for clarity. The surrounding conserved C_α trace is also shown.

show that several surrounding C_α atoms superimpose as well. The observed good superposition supports the hypothesis that common fingerprints correspond to geometrically conserved packing patterns within protein families. Notice that our algorithm for finding common subgraphs does not require identified subgraphs to superimpose, unlike those found by LCP [Akutsu *et al.*, 1997] or other methods.

The largest fingerprints found in the Kinase dataset do not superpose, but differ widely in their conformation. This is because the sparse and unconstrained nature of the largest Kinase fingerprints and the large overall geometric differences in the structures themselves. Such large and unique packing patterns, despite their structural diversity, could be useful for the annotation of new structures.

5 Discussion and Future Work

In this paper we report on the application of the general frequent subgraph mining algorithm to the identification of common packing patterns in protein structures represented as graphs. The goal of this investigation was to identify frequent subgraphs common to all (or the majority of) proteins belonging to the same structural and functional family in the SCOP database and explore these subgraphs as amino acid residue fingerprints specific to the underlying family. Although protein graphs are complex, this application has become possible, thanks to several advanced features of the frequent subgraph mining algorithm [Huan *et al.*, 2003, 2004b] employed in this paper.

Three graph representations of the protein structures termed CD, DT, and AD graphs have been explored to identify fingerprints. As discussed in Section 2.2, all three representations used the vertices defined by the proteins' C_α atoms, but differ by the approach used to define the edges. These three representations have been compared in terms of the number and composition of the unique protein family-specific features identified by subgraph mining, and computational efficiency of identifying these features. Our results demonstrate that using a simplified graph representation such as DT is needed when structures of a protein family are conserved and a huge number of patterns can be found by the graph mining algorithm. On the other hand, CD graphs may support more patterns than DT graphs if the related computation is feasible. AD graphs provide a good trade-off due to their relative computational efficiency and their robustness in taking into account possible experimental errors in determining protein atomic coordinates. We demonstrate that common fingerprints could include active site residues as well as correspond to other structurally conserved residue patterns.

The success of these preliminary studies encourages us to consider several possible directions for future investigations of protein graphs with the frequent subgraph mining algorithms. We plan to develop an incremental subgraph mining approach that repeatedly increases the parameter ϵ in the AD graphs until it has found the maximum number of fingerprints. We further hypothesize that some of the individual subgraph-fingerprints identified as the result of protein structure analysis may also carry sequence specificity, i.e., contain characteristic residues found in the same order and approximately at the same separation distances in the underlying primary sequences of the protein family. Instances of such structure-derived primary sequence motifs formally similar to PROSITE patterns [Hofmann *et al.*, 1999] have been implicated in our earlier analyses of protein packing with Delaunay tessellation, summarized in [Tropsha *et al.*, 2003]. This latter hypothesis offers an exciting new avenue in exploring structure-sequence-function relationships in proteins by using subgraphs (i.e., residue packing patterns) for functional and structural annotation of not only novel protein structures but also sequences from the ongoing genomics projects.

6 Acknowledgements

This work was supported in part by NSF grant 0076984 for JS, NSF grant MCB/ITR-0112896 and NIH grant GM066940-01 awarded to AT, Microsoft eScience applications award to WW and the bioinformatics graduate fellowship from UNC General Administration to JH.

References

Akutsu, T., Tamaki, H. & Tokuyama, T. (1997). Distribution of distances and triangles in a point set and algorithms for computing the largest common point sets. In *13th Annual ACM Symp. on Computational Geometry*. pp. 314–323.

Bandyopadhyay, D. & Snoeyink, J. (2004). Almost-Delaunay simplices : Nearest neighbor relations for imprecise points. In *ACM-SIAM Symposium On Distributed Algorithms*. pp. 403–412.

Barber, C. B., Dobkin, D. P. & Huhdanpaa, H. (1996). The Quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, **22**, 469–483.

Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I. & Bourne, P. (00). The protein data bank. *Nucleic Acids Research*, **28**, 235–42.

Borgelt, C. & Berhold, M. R. (2002). Mining molecular fragments: Finding relevant substructures of molecules. In *Proc. International Conference on Data Mining'02*.

Bradley, P., Kim, P. S. & Berger, B. (2002). Trilogy: Discovery of sequence-structure patterns across diverse proteins. *Proceedings of the National Academy of Sciences*, **99**, 8500–8505.

Cammer, S., Carter, C. W. & Tropsha, A. (2002). Identification of sequence-specific tertiary packing motifs in protein structures using delaunay tessellation. In *Lecture Notes in Computational Science and Engineering (Schlick, T. and Gan H.H.,eds.), Springer-Verlag, Berlin*, volume 24. pp. 477–494.

Chakraborty, S. & Biswas, S. (1999). Approximation algorithms for 3-d common substructure identification in drug and protein molecules. *Workshop on Algorithms and Data Structures*, 253–264.

Cormen, T. H., Leiserson, C. E. & Rivest, R. L. (2001). Introduction to Algorithms. MIT press.

Dehaspe, L., Toivonen, H. & King, R. D. (1998). Finding frequent substructures in chemical compounds. In *4th International Conference on Knowledge Discovery and Data Mining*. pp. 30–36.

Delaunay, B. (1934). Sur la sphère vide. A la memoire de Georges Voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, **7**, 793–800.

Finn, P. W., Kavraki, L. E., Latombe, J., Motwani, R., Shelton, C. R., Venkatasubramanian, S. & Yao, A. (1997). Rapid: Randomized pharmacophore identification for drug design. *Symposium on Computational Geometry*, 324–333.

Fischer, D., Wolfson, H., Lin, S. L. & Nussinov, R. (1994). Three-dimensional, sequence order-independent structural comparison of a serine protease against the crystallographic database reveals active site similarities: potential implication to evolution and to protein folding. *Protein Science*, **3**, 769–778.

Grindley, H., Artymiuk, P., Rice, D. & Willet, P. (1993). Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. *J. Mol. biol.*, **229**, 707–721.

Hofmann, S. K., Bucher, P., Falquet, L. & Bairoch, A. (1999). The prosite database, its status in 1999. *Nucleic Acids Res*, **27(1)**, 215–219.

Huan, J., Wang, W., Bandyopadhyay, D., Snoeyink, J., Prins, J. & Tropsha, A. (2004a). Mining protein family specific residue packing patterns from protein structure graphs. In *Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB)*. pp. 308–315.

Huan, J., Wang, W. & Prins, J. (2003). Efficient mining of frequent subgraph in the presence of isomorphism. In *Proc. International Conference on Data Mining'03*.

Huan, J., Wang, W., Prins, J. & Yang, J. (2004b). Spin: Mining maximal frequent subgraphs from graph databases. *ACM SIGKDD*.

Huan, J., Wang, W., Prins, J. & Yang, J. (2004c). Spin: Mining maximal frequent subgraphs from graph databases. *UNC Technical Report*.

Huan, J., Wang, W., Washington, A., Prins, J., Shah, R. & Tropsha, A. (2004d). Accurate classification of protein structural families based on coherent subgraph analysis. In *Proc. Pacific Symposium on Biocomputing*.

Indyk, P., Motwani, R. & Venkatasubramanian, S. (1999). Geometric matching under noise: Combinatorial bounds and algorithms. *ACM Symposium on Discrete Algorithms*.

Inokuchi, A., Washio, T. & Motoda, H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. *In PKDD'00*.

Jacobs, D., Rader, A., Kuhn, L. & Thorpe, M. (2001). Graph theory predictions of protein flexibility. *Proteins: Struct. Funct. Genet.*, **44**, 150–155.

Klein, P. N. (1998). Computing the edit-distance between unrooted ordered trees. In *Proc. 6th Annual Enropean Symposium*. pp. 91–102.

Kleywegt, G. J. (1999). Recognition of spatial motifs in protein structures. *Journal of Molecular Biology*, **285(4)**, 1887–1897.

Kuramochi, M. & Karypis, G. (2001). Frequent subgraph discovery. In *Proc. International Conference on Data Mining'01*.

Liang, J., Edelsbrunner, H., Fu, P., Sudhakar, P. & Subramaniam, S. (1998). Analytical shape computing of macromolecules I: molecular area and volume through alpha shape. *Proteins*, **33**, 1–17.

Lovell, S., Davis, I., III, W. A., de Bakker, P., Word, J., Prisant, M., Richardson, J. & Richardson, D. (2003). Structure validation by c_α geometry: ϕ, ψ and c_β deviation. *Proteins: Structure, Function and Genetics*, **50**, 437–50.

Milik, M., Szalma, S. & Olszewski, K. (2003). Common structural cliques: a tool for protein structure and function analysis. *Protein Eng*, **16**.

Mitchell, E. M., Artymiuk, P. J., Rice, D. W. & Willett, P. (1990). Use of techniques derived from graph theory to compare secondary structure motifs in proteins. *J Mol Biol*, **212**, 151–166.

Murzin, A., Brenner, S., Hubbard, T. & Chothia., C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, **247**, 536–40.

Richards, F. M. (1974). The interpretation of protein structures: total volume, group volume distributions, and packing density. *J. Molecular Biology*, **82**, 1–14.

Singh, A. P. & Brutlag, D. L. (1997). Hierarchical protein structure superposition using both secondary structure and atomic representations. In *ISMB*. pp. 284–293.

Singh, R., Tropsha, A. & Vaisman, I. (1996). Delaunay tessellation of proteins. *J. Comput. Biol.*, **3**, 213–222.

Tropsha, A., Carter, C., Cammer, S. & Vaisman, I. (2003). Simplicial neighborhood analysis of protein packing (SNAPP) : a computational geometry approach to studying proteins. *Methods Enzymol.*, **374**, 509–544.

Tsai, J., Taylor, R., Chothia, C. & Gerstein, M. (1999). The packing density in proteins: Standard radii and volumes. *Journal of Molecular Biology*, **290**, 253–266.

Vishveshwara, S., Brinda, K. V. & Kannan, N. (2002). Protein structure: Insights from graph theory. *J. of Theo. and Comp. Chem.*, **1(1)**, 187–211.

Wako, H. & Yamato, T. (1998). Novel method to detect a motif of local structures in different protein conformations. *Protein Engineering*, **11**, 981–990.

Wang, J., Shapiro, B., Sasha, D., Zhang, K. & Currey, K. M. (1998). A algorithm for finding the largest approximately common substructures of two trees. *IEEE Transactions on Pattern Anaylsis and Machine Intelligence*.

Wernisch, L., Hunting, M. & Wodak, S. (1999). Identification of structural domains in proteins by a graph heuristic. *Proteins*, **35**, 338–352.

Yan, X. & Han, J. (2002). gspan: Graph-based substructure pattern mining. In *Proc. International Conference on Data Mining'02*.