DISTANCE-BASED IDENTIFICATION OF STRUCTURE MOTIFS IN PROTEINS USING CONSTRAINED FREQUENT SUBGRAPH MINING

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

Structure motifs are amino acid packing patterns that occur frequently within a set of protein structures. We define a labeled graph representation of protein structure in which vertices correspond to amino acid residues and edges connect pairs of residues and are labeled by (1) the Euclidian distance between the C_α atoms of the two residues and (2) a boolean indicating whether the two residues are in physical/chemical contact. Using this representation, a structure motif corresponds to a labeled clique that occurs frequently among the graphs representing the protein structures. The pairwise distance constraints on each edge in a clique serve to limit the variation in geometry among different occurrences of a structure motif. We present an efficient constrained subgraph mining algorithm to discover structure motifs in this setting. Compared with contact graph representations, the number of spurious structure motifs is greatly reduced.

Using this algorithm, structure motifs were located for several SCOP families including the Eukaryotic Serine Proteases, Nuclear Binding Domains, Papain-like Cysteine Proteases, and FAD/NAD-linked Reductases. For each family, we typically obtain a handful of motifs within seconds of processing time. The occurrences of these motifs throughout the PDB were strongly associated with the original SCOP family, as measured using a hyper-geometric distribution. The motifs were found to cover functionally important sites like the catalytic triad for Serine Proteases and co-factor binding sites for Nuclear Binding Domains. The fact that many motifs are highly family-specific can be used to classify new proteins or to provide functional annotation in Structural Genomics Projects.

Keywords: protein structure comparison, structure motif, graph mining, clique

1. INTRODUCTION

This paper studies the following structural comparison problem: given a set $\mathcal{G}$ of three dimensional (3D) protein structures, identify all structure motifs that occur with sufficient frequency among the proteins in $\mathcal{G}$. Our study is motivated by the large number of ($> 35,000$) 3D protein structures stored in public repositories such as the Protein Data Bank (PDB, [4]). The recent Structural Genomics projects [28] aim to generate many new protein structures in a high-throughput fashion, which may further increase the available protein structures significantly. With fast growing structure data, automatic and effective knowledge discovery tools are needed to gain insights from the available structure data in order to generate testable hypotheses about the functional role of proteins and the evolutionary relationship among proteins.

Our study is also motivated by the complex relationship between protein structure and protein function [8]. It is well known that global structure similarity does not necessarily imply similar function. For example, the TIM barrels are a large group of proteins with remarkably similar global structures, yet widely varying functions [23]. Conversely, similar function does not necessarily imply similar global structure: the most versatile enzymes, hydro-lyases and the O-glycosyl glucosidases, are associated with 7 different global structural families [11]. Many globally dissimilar structures show convergent evolution of biological function. Because of the puzzling relationship between global protein structure and function, recent research effort in protein structure comparison has shifted to identifying local structural features (referred to as *structure motifs*) responsible for biological functions including protein-protein interaction, ligand binding, and catalysis [3, 5, 30–33]. A recent review of methods and applications involved in protein structure motif identification can be found in [19].

Using a graph representation of proteins, we formalize the structure motif identification problem as a frequent clique mining problem in a set of graphs $\mathcal{G}$ and present a novel constrained clique mining algorithm to obtain recurring cliques from $\mathcal{G}$ that satisfy certain additional constraints. The constraints are encoded in the graph representation of protein structure as pair-wise amino acid residue distances, pair-wise amino acid residue interactions, and the physical/chemical properties

of the amino acid residues and their interactions in a protein structure.

Compared to other methods, our method offers the following advantages. First, our method is efficient. It usually takes only a few seconds to process a group of proteins of moderate size (ca. 30 proteins), which makes it suitable for processing protein families defined by various classifications such as SCOP or EC (Enzyme Commission). Second, our results are specific. As we show in our experimental study section, by requiring structure motifs to recur among a group of proteins, rather than in just two proteins, we significantly reduce spurious patterns without losing structure motifs that have clear biological relevance. With a quantitative definition of significance based on the hyper-geometric distribution, we find that the structure motifs we identify are specifically associated with the original family. This association may significantly improve the accuracy of feature-based functional annotation of structures from structural genomics projects.

The rest of this paper is organized as follows. In Section 1.1, we review recent progress in discovering protein structure motifs. In Section 2, we review definitions related to graphs and introduce the constrained subgraph mining problem. In Section 3, we discuss our graph representation of proteins structures. In Section 4, we present a detailed description of our method. We also include a practical implementation of the algorithm that supports the experimental study in Section 5. Finally, Section 6 concludes with a brief discussion of future work.

1.1. Related work

There is an extensive body of literature on comparing and classifying proteins using multiple sequence or structure alignment, such as VAST [9] and DALI [12]. Here we focus on the recent algorithmic techniques for discovering structure motifs from protein structures. The methods can be classified into the following five types:

- Depth-first search, starting from simple geometric patterns such as triangles, progressively finding larger patterns [5, 25, 30].
- Geometric hashing, originally developed in computer vision, applied pairwise between protein structures to identify structure motifs [3, 24, 35].
- String pattern matching methods that encode the local structure and sequence information of a protein as a string, and apply string search algorithms to derive motifs [17, 18, 32].
- Delaunay tessellation (DT) [6, 20, 33] partitioning the structure into an aggregate of non-overlapping, irregular tetrahedra thus identifying all unique nearest neighbor residue quadruplets for any protein [33].
- Graph matching methods comparing protein structures modeled as graphs and discovering structure motifs by finding recurring subgraphs [1, 10, 14, 22, 29, 31, 38].

Geometric hashing [21] and graph matching [38] methods have been extended for inferring recurring structure motifs from multiple structures, but both methods have exponential running time in the number of structures in a data set.

2. CONSTRAINED FREQUENT CLIQUE MINING

2.1. Labeled graphs

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

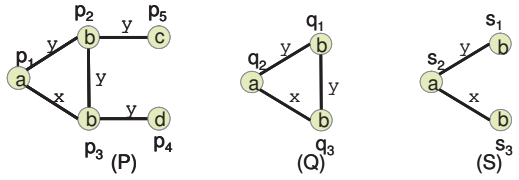

Fig. 1. Database $\mathcal{G}$ of three labeled graphs. The mapping (isomorphism) $q_1 \rightarrow p_2$, $q_2 \rightarrow p_1$, and $q_3 \rightarrow p_3$ demonstrates that clique Q is isomorphic to a subgraph of P and so we say that Q occurs in P. Set $\{p_1, p_2, p_3\}$ is an embedding of Q in P. Similarly, graph S (non-clique) occurs in both graph P and graph Q.

$G' = (V', E')$ is a *subgraph* of G, denoted by $G' \subseteq G$, 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'.

2.2. Constraints on structure motifs

A *constraint* in our discussion is a function that assigns a boolean value to a subgraph such that true implies that the subgraph has some desired property and false indicates otherwise. For example, the following statement "each amino acid residue in a structure motif must have a solvent accessible surface of sufficient size" is a constraint. This constraint selects only those structure motifs that are close to the surface of proteins. The task of formulating the right constraint(s) is left for domain experts. As part of our computational concern, we answer the following two questions: (1) what types of constraints can be efficiently incorporated into a subgraph mining algorithm and (2) how to incorporate a constraint if it can be efficiently incorporated. The answer to the two questions is the major contribution of this paper and is discussed in details in Section 4.

2.3. Graph matching

A fundamental part of our constrained subgraph mining method is to find an occurrence of a graph H within another graph G. To make this more precise, we say that graph H *occurs* in G if we can find an *isomorphism* between graph $H = (V_H, E_H, \Sigma, \lambda_H)$ and some subgraph of $G = (V_G, E_G, \Sigma, \lambda_G)$. An isomorphism from H to the subgraph of G defined by vertices $V \subseteq V_G$ is a 1-1 mapping between vertices $f : V_H \rightarrow V$ that preserves edges and edge/node labels. The set V is an *embedding* of H in G. This definition is illustrated in Figure 1.

In this paper, we restrict ourselves to matching *cliques*, i.e. fully connected subgraphs. For example, the graph Q in Figure 1 is a clique since each pair of (distinct) nodes is connected by an edge in Q while S is not. In protein structure graphs, a clique corresponds to a structure motif with all pairwise inter-residue distances specified.

2.4. The constrained frequent clique mining problem

Given a set of graphs, or a *graph database* $\mathcal{G}$, we define the *support* of a clique C, denoted by $s(C)$, as the fraction of graphs in $\mathcal{G}$ in which C occurs. We choose a support threshold $0 < \sigma \leq 1$, and define C to be *frequent* if it occurs in at least fraction σ of the graphs in $\mathcal{G}$. Note that while C may occur many times within a single graph, for the purpose of measuring support, these count as only one occurrence. Given a constraint ρ, the problem of *Constrained Frequent Clique Mining* is to identify all frequent cliques C in a graph database $\mathcal{G}$ such that $\rho(C)$ is true. Figure 2 shows all cliques (without any constraint) which appear in at least two graphs in the graph database shown in Figure 1. If we use support threshold $\sigma = 2/3$ without any constraint, all six cliques will be reported to users. If we increase σ to 3/3, only cliques A_1, A_2, A_3, and A_4 will be reported. If we use support threshold $\sigma = 2/3$ and the constraint that each clique should contains at least one node with label "a", the constrained frequent cliques are A_1, A_3, A_4, and A_6.

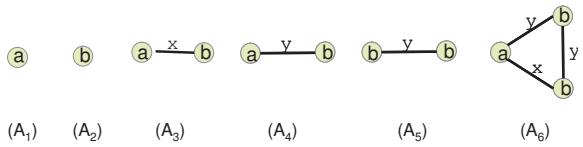

Fig. 2. All (non-empty) frequent cliques with support $\geq 2/3$ in $\mathcal{G}$ from Figure 1. The actual support values are: (3/3, 3/3, 3/3, 3/3, 2/3, 2/3) for cliques from A_1 to A_6.

3. HYBRID GRAPH REPRESENTATION OF PROTEIN STRUCTURES

3.1. Graph representation overview

We model a protein structure as a labeled undirected graph by incorporating pairwise amino acid residue distances and contact relation in the following way. The nodes of our protein graphs represent the C_α atoms of each amino acid residue. We create edges connecting each and every pair of (distinct) residues, labeled by two types of information: (1) The Euclidian distance between the two related C_α atoms and (2) A boolean indicates whether the two residues have physical/chemical contact. More precisely, a protein in our study is a labeled graph $P = (V, E, \Sigma, \lambda)$ where

- V is a set of nodes that represents the set of amino acid residues in the protein
- $E = V \times V$ - (u, u) for all $u \in V$
- $\Sigma = \Sigma_V \cup \Sigma_E$ is the set of disjoint node labels (Σ_V) and edge labels (Σ_E)
- Σ_V is the set of 20 amino acid types
- $\Sigma_E = \mathbb{R}^+ \times \{true, false\}$ where $\mathbb{R}^+$ is the set of positive real numbers
- λ assigns labels to nodes and edges.

Our graph representation can be viewed as a hybrid of two popular representation of proteins: that of distance

matrix representation [8] and that of contact map representation [13].

In practice we are not concerned with interactions over long distances (say > 13 Å), so proteins need not be represented by complete graphs. Since each amino acid occupies a real volume, the number of edges per vertex in the graph representation can be bounded by a small constant.

The graph representation presented here is similar to those used by other groups [25, 38]. The major difference is that in our representation, geometric constraints are in the form of pairwise distance constraints and are embedded into the graph representation to model geometrically conserved patterns. The absence of geometric constraints can lead to many spurious matches as noticed in [25, 38]. Another difference is that we explicitly specify the "contact" relation. The contact relation enables us to incorporate various constraints into the subsequent graph mining process and further reduce irrelevant patterns.

In the following, we discuss how to discretize distances into distance bins, which is important for our structure motif identification algorithm.

$l \leq 4$	$4 < l \leq 5.5$	$5.5 < l \leq 7$	$7 < l \leq 8.5$
1	2	3	4
$8.5 < l \leq 10$	$10 < l \leq 11.5$	$11.5 < l$	
5	6	7	

Fig. 3. Mapping distances l to bins. The unit is Å.

3.2. Distance discretization

To map continuous distances to discrete values, we discretize distances into bins. The width of such bins is commonly referred to as the *distance tolerance*, and popular choices are 1 Å [22], 1.5 Å [5], and 2 Å [26]. In our system, we choose the median number 1.5 Å as shown in Figure 3, which empirically delivers patterns with good geometric conservation.

4. THE CONSTRAINED CLIQUE MINING ALGORITHM

In this section, we present a detailed discussion on (1) what types of constraints can be incorporated efficiently into a subgraph mining algorithm and (2) how to incorporate them.

Our strategy relies on designing graph normalization functions that map cliques to one dimensional sequences of labels. A *graph normalization function* is a 1-1 mapping $\mathcal{N}$ such that $\mathcal{N}(G) = \mathcal{N}(G')$ if and only if $G = G'$. In other words, a graph normalization function always assigns a unique string to each unique graph. The string $\mathcal{N}(G)$ is the *canonical code* (code in short) of the graph G with respect to the function $\mathcal{N}$.

Many graph normalization functions have a very desirable property: prefix-preservation. A graph normalization function is *prefix-preserving* if for every graph G, there always exists a subgraph $G' \subset G$ such that the code of G' is a prefix of the code of G. Examples of prefix-preserving graph normalization functions include the DFS code [39] and the CAM code [15]. As we prove in Theorem 4.8, with a generic depth first search algorithm, a prefix-preserving graph normalization function guarantees that no frequent constrained patterns can be missed. The design challenge here is to construct a graph normalization function that is prefix-preserving in the presence of constraints.

4.1. A synthetic example of constraints

The following constraint is our driving example for constrained clique mining. The constraint states that we should only report frequent cliques that contain at least one edge label of "y". The symbol "y" is selected to make the constraint works best with the graph example we show in Figure 1. Applying this constraint to all the frequent cliques shown in Figure 2, we find that there are only three cliques satisfying the constraint, namely A_4, A_5, and A_6. We name this simple constraint an *edge label constraint* and show a specific graph normalization function that is prefix-preserving for this edge label constraint. Before we do that, we introduce a normalization that does not support any constraints. Our final solution will adapt this constraint-unaware graph normalization function.

4.2. A graph normalization function that does not support constraints

We use our previous canonical code [15] for graph normalization, outlined below for completeness.

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 node labels as diagonal entries) 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 edges are undirected, we are concerned only with the lower triangular entries of M. Figure 4 shows examples of adjacency matrices and codes for the labeled graph Q shown in the same figure.

The lexicographic order of sequences defines a total order over adjacency matrix codes. Given a graph G, its *canonical code*, denoted by $F(G)$, is the maximal code among all its possible codes. For example, $F(M_1)$ =“*bybyxa*” shown in Figure 4 is the canonical code of the graph.

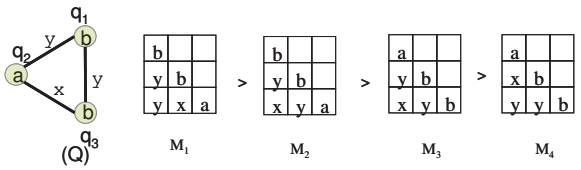

Fig. 4. All possible adjacency matrices for the clique Q. Since adjacency matrices for undirected graph are symmetric, only the lower triangular part of the matrices are shown. Using lexicographic order of symbols: $y > x > d > c > b > a > 0$, we have $code(M_1)$ =“*bybyxa*” $> code(M_2)$ = “*bybxya*” $> code(M_3)$ = “*aybxyb*” $> code(M_4)$ =“*aybxyb*”. Hence $code(M_1)$ is the canonical code for the graph Q.

4.3. A graph normalization function that supports constraints

In this section, we introduce the definition of a generic graph normalization function ψ. In the following two sections, we show the applications of the generic graph normalization function ψ. In Section 4.4, we show that several widely used constraints for protein structure motifs lead to a well-defined function ψ. In Section 4.5, we show that the function ψ can be used in a depth-first constrained clique search procedure to make sure that each constrained frequent clique is searched once and exactly once.

Definition 4.1. Given the graph normalization function F with its codomain Γ and a constraint ρ, a **generic graph normalization function** ψ, is a function that maps a graph G to Γ^* recursively as:

$$\psi(G) = \begin{cases} F(G) & \text{if } \rho(G) \text{ is false} \\ F(G) & \text{if } \mid V(G) \mid = 1 \\ \max\limits_{G' \subset G, \rho(G')} \psi(G')\$F(G) & \text{otherwise} \end{cases}$$

where G' is a subgraph of G with size one less than G; $\psi(G')\$f(G)$ is the concatenation of the code produced by $\psi(G')$, the symbol \$, and the code $F(G)$. We assume that the symbol \$ is not in the set Γ and we use this symbol to separate different parts of the generic graph normalization. The total ordering on strings (max) is defined by lexicographical order with the underlying symbol ordering assumed from Γ.

Example 4.1. Applying the generic graph normalization to the graph Q shown in Figure 4, with the graph normalization F be the canonical code we discussed in Section 4.2, $\psi(G)$ is $b\$byb\$bybyxa$. $bybyxa$ is a suffix of the code since it is the canonical code of graph Q. In a search for $\psi(Q)$, two subgraphs of Q that satisfy the edge label constraint are searched. One is a single edge connecting nodes with labels “b” and “b” with an edge label “y” and the other is also a single edge connecting nodes with labels “a” and “b” with an edge label “y”. Since the canonical code for the first (“byb”) is greater than that of the second (“bya”), we put string “byb” before the string “bybyxa” and obtain “byb\$bybyxa”. Finally we add a single “b” before the string “byb\$bybyxa” at the last step of the recursive definition and we have $\psi(G) = b\$byb\$bybyxa$.

Theorem 4.1. *$\psi(G)$ exists for every graph G with the edge label constraint.*

Proof. Let’s first assume that a graph G contains an edge label “y”. We claim that there always exists a subgraph G' of G that also contains the same label. This observation suggests that we can always find at least one G' in the recursive definition, and hence $\psi(G)$ is defined. If the original G does not contain any edge label “y”, its code is $F(G)$, which is also defined. □

Theorem 4.2. *ψ is a 1-1 mapping and thus a graph normalization function.*

Proof. If two graphs are isomorphic, they must give the same string, according to the definition of ψ. To prove that two graphs that have the same canonical strings are isomorphic, noticing that the last element of a label sequence produced by ψ is $F(G)$ where F is a graph normalization procedure. Therefore two identical sequence must imply the same graph, as guaranteed by F. □

Theorem 4.3. *For all G such that $\rho(G)$ is true, there exists a subgraph $G' \subset G$ with size one less than G such*

that $\rho(G')$ is true and $\psi(G')$ is a prefix of $\psi(G)$.

Proof. This property is a direct result of the recursive definition 4.1. □

We notice that in proving Theorems 4.2 and 4.3, we do not use the definition of the constraint ρ. In other words, Theorems 4.2 and 4.3 can be proved as long as we have Theorem 4.1. Therefore, we have the following theorem:

Theorem 4.4. *If ψ is defined for every graph with respect to a given constraint ρ, ψ is 1-1 and prefixing-preserving.*

Proof. This is a direct result of the recursive definition 4.1. □

4.4. More examples related to protein structure motifs

Let's first view a real-world example of constraint that is widely used in structure motif discovery. The *connected component constraint* (CC constraint for short) asserts that in a structure motif, each amino acid residue is connected to at least another amino acid residue by a contact relation and that the motif is a connected component with respect to the contact relation. The intuition of the CC constraint is that a structure motif should be compact and hence has no isolated amino acid residue. To be formal, the CC constraint is a function cc that assigns value true to a graph if it is a connected component according to the contact relation and false otherwise.

As another example, the *contact density constraint* asserts that the ratio of the number of contacts and the total number of edges in a structure motif should be greater than a predefined threshold. Such ratio is referred to as the *contact density* (density) of the motif and the constraint is referred to as the density constraint. The intuition of the density constraint is that a structure motif should be compact and the amino acid residues in the motif should well interact with other. This constraint may be viewed as a more strict version of the CC constraint which only requires a motif to be connected component. Again, to be formal, the density constraint is a function d that assigns value true to a graph if its contact density is at least some predefined threshold and false otherwise.

It would be an awkward situation if we need to define a new graph normalization procedure for each of the constraints we discuss above. Fortunately, this is not the case. In the following discussion, we show that generic graph normalization function ψ is well defined for these two constraints.

Theorem 4.5. *$\psi(G)$ exist for every graph G with respect to the CC constraint or the density constraint.*

Proof. We only show the proof of the theorem for the CC constraint and that for the density constraint can be proved similarly. The key observation is for every graph G of size n that is a connected component with respect to the node contact relation, there exists a subgraph $G' \subseteq G$ such that G' is a connected component according to the same contact relation. The observation is a well-known result from graph theorem and a detail proof can be found in [15]. □

Following Theorem 4.4, we have the following theorem.

Theorem 4.6. *ψ is a 1-1 mapping and prefix-preserving for the CC constraint or the density constraint.*

After working several example constraints, we study the sufficient and necessary condition such that our graph normalization function ψ is well defined for a constraint ρ. The following theorem formalize the answer.

Theorem 4.7. *Given a constraint ρ, $\psi(G)$ exist for every graph G with respect to the constraint ρ if and only if for each graph G of size n such that $\rho(G)$ is true, there exists a subgraph $G' \subset G$ of size $n-1$ such that $\rho(G')$ is also true.*

Proof. (if) For a graph G such that $\rho(G)$ is true, if there exists one $G' \subset G$ such that $\rho(G')$ is also true, by the definition of ψ, $\psi(G)$ exists.

(only if) If $\psi(G)$ exists for every graph G with respect to a constraint ρ, for a graph G such that $\rho(G)$ is true, by the definition of ψ, we always have at least one $G' \subset G$ such that $\rho(G')$ is also true. □

4.5. cliquehashing

We have designed an efficient algorithm identifying frequent cliques from a labeled graph database with constraints, as described below. At the beginning of the algorithm we scan a graph database and find all frequent node

CliqueHashing($\mathcal{G}$, σ, ρ)
begin
1. **for** each node label $t \in \lambda(v), v \in V[G], G \in \mathcal{G}$ **do**
2. $counter[t] \leftarrow counter[t] \cup \{v\}$
3. $C \leftarrow C \cup \{t\}$
4. **end for**
5. **for** $t \in C$ **do**
6. **if**($s(t) \geq \sigma$, $\rho(s)$ is true) **do**
7. $\mathcal{F} \leftarrow \mathcal{F} \cup$ backtrack_search($t, counter[t]$)
8. **end if**
9. **end for**
10. return $\mathcal{F}$
end

backtrack_search($t_0, \mathcal{O}, \rho$)
begin
1. **for** each clique $h \in \mathcal{O}$ **do**
2. $\mathcal{O}' \leftarrow \{f | f = h \cup v, h \subset V[G], v \in (V[G] - h)\}$
3. **for** each occurrence of a clique $f \in \mathcal{O}'$ **do**
4. $t \leftarrow \psi(f)$
5. $counter[t] \leftarrow counter[t] \cup \{f\}$
6. $C \leftarrow C \cup \{t\}$
6. **end for**
7. **end for**
8. **for** each $t \in C$ **do**
9. **if**($s(t) \geq \sigma$, $t_0 \sqsubseteq t$, and $\rho(t)$ is true) **do**
10. $\mathcal{F} \leftarrow \mathcal{F} \cup$ backtrack_search(t, counter[t])
11. **end if**
12. **end for**
13. return $\mathcal{F}$
end

Fig. 5. The CliqueHashing algorithm which reports frequent cliques, $\mathcal{F}$, from a group of graphs $\mathcal{G}$ with support at least σ and with a constraint ρ. ψ is the graph normalization function defined in Definition 4.1. $x \sqsubseteq y$ if string x is a prefix of string y. $s(G)$ is the support of a graph G.

types (line 1-4, Figure 5). The node types and their occurrences are kept in a hash table *counter*. At a subsequent step, a frequent clique with size $n \geq 1$ is picked from the hash table and is extended to all possible $n + 1$ cliques by attaching one additional node to its occurrences in all possible ways. These newly discovered cliques and their occurrences, are again indexed in a separate hash table and enumerated recursively. The algorithm backtracks to the parents of a clique if no further extension from the clique is possible. The overall algorithm stops when all frequent node types have been enumerated. We illustrate the CliqueHashing algorithm, with the edge label constraint, in Figure 6.

Theorem 4.8. *If ψ is well defined for all possible graphs with the constraint ρ, the CliqueHashing algorithm identifies all frequent constrained cliques from a graph database exactly once.*

Proof. The prefix preserving property of Definition 4.1 implies that at least one subclique of a frequent clique will pass the IF statement of line 9, in the backtrack_search procedure in CliqueHashing. Therefore the algorithm will not miss any frequent cliques in the presence of a constraint ρ.

The proof that the algorithm discovers every constrained frequent cliques exactly once may not be obvious at first glance. The key observation is that for a clique G of size n, there is only one subclique with size $n - 1$ that has a code matching a prefix of $\psi(G)$. If we can prove the observation, by the line 9 of the backtrack_search procedure, the CliqueHashing algorithm guarantees that each constrained frequent cliques will be discovered exactly once.

To prove the observation, we assume to the contrary that there are at least two such subcliques with the same size and both give codes as prefixes of $\psi(G)$. We claim that one of the two codes must be a prefix of the other (by the definition of prefix). The claim leads to the conclusion that one of two subcliques must be a subclique of the other (by the definition of ψ). The conclusion contradicts our assumption that the two subcliques have the same size. □

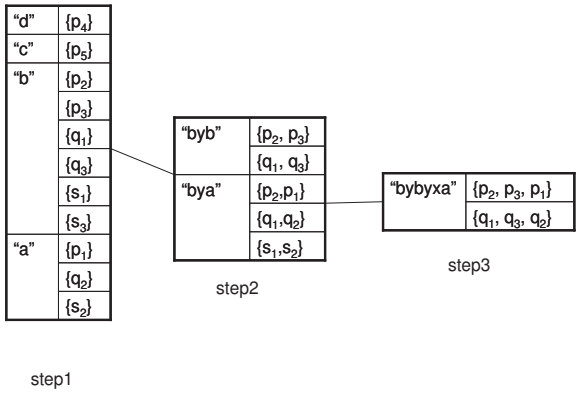

Fig. 6. The contents of the hash table counter after applying the CliqueHashing algorithm to the data set shown in Figure 1 with the edge label constraint.

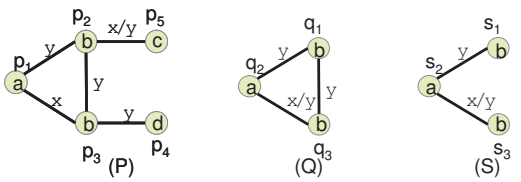

Fig. 7. A graph database of three graphs with multiple labels.

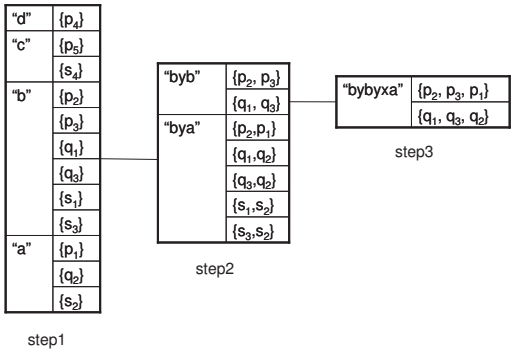

Fig. 8. the contents of the hash table counter after applying the CliqueHashing algorithm to the data set shown left.

4.6. CliqueHashing on Multi-labeled Graphs

A *multi-labeled graph* is a graph where there are two or more labels associated with a single edge in the graph. The CliqueHashing algorithm can be applied to multi-labeled graphs directly without major modifications. The key observation is that our enumeration is based on occurrences of cliques (line 3 in function backtrack_search). In Figure 7, we show a graph database with three multi-labeled graphs. In figure 8, we show (pictorially) how the CliqueHashing algorithm can be applied to graphs with multilables.

In the context of the structure motifs detection, handling multi-labeled graphs is important for the following reason. First, due to the imprecision in 3D coordinates data in motif discovery, we need to tolerate distance variations between different instances of the same motif. Second, partitioning the 1D distance space into distance bins is not a perfect solution since distance variations can not be well handled at the boundary of the bins. In our application distance bins may lead to a significant number of missing motifs. Using a multi-labeled graph we can solve the boundary problem by using "overlapping" bins to take care of boundary effect.

5. EXPERIMENTAL STUDY

5.1. Experimental setup

To exclude redundant structures from our analysis, we used the culled PDB list (http://www.fccc.edu/research/labs/dunbrack/pisces/culledpdb.html) with sequence similarity cutoff value 90% (resolution = 3.0, R factor = 1.0). This list contains about one quarter of all protein structures in PDB; remaining ones are regarded as duplicates to proteins in the list. We study four SCOP families: Eukaryotic Serine Protease (ESP), Papain-like Cysteine Protease (PCP), Nuclear Binding Domains (NB), and FAD/NAD-linked reductase (FAD). Each protein structure in a SCOP family was converted to its graph representation as outlined in Section 3. The pairwise amino acid residue contacts are obtained by computing the almost-Delaunay edges [2] with $\epsilon = 0.1$ and with length up to 8.5 Å, as was also done in [14]. Structure motifs from a SCOP family were identified using the CliqueHashing algorithm with the CC constraint that states "each amino acid residue in a motif should contact at least another residue and the motif should be a connected component with respect to the contact relation". Timings of the search algorithm were reported using the same hardware configuration used in [14].

In Table 1, we document the four families including their SCOP ID, total number of proteins in the family (N), the support threshold we used to retrieve structure motifs (σ), and the processing time (T, in seconds). In the same table, we also record all the structure motifs identified, giving the motifs' compositions (a sequence of one-letter residue codes), actual support values (κ), the number of occurrences outside the family in the representative structures in PDB (referred to as the *background frequencies* hereafter) (δ), and their statistical significance in the family (P). The statistical significance is computed by a hyper-geometric distribution, specified in Appendix 7.1. Images of protein structures were produced using VMD [16] and residues in the images were colored by the residue identity using default VMD settings.

5.2. Eukaryotic serine protease

The structure motifs identified from the ESP family were documented at the top part of Table 1. The data indicated that the motifs we found are highly specific to the ESP family, measured by $P\text{–}value \leq 10^{-82}$. We have investigated the spatial distribution of the residues covered by

Table 1. Motifs

Motif	Composition	κ	δ	$-log(P)$	Motif	Composition	κ	δ	$-log(P)$	Motif	Composition	κ	δ	$-log(P)$
Eukaryotic Serine Protease (ID: 50514) N: 56 σ: 48/56, T: 31.5														
1	DHAC	54	13	100	14	DHAC	50	6	100	27	DASC	49	20	92
2	ACGG	52	9	100	15	HACA	50	8	100	28	SAGG	49	31	90
3	DHSC	52	10	100	16	ACGA	50	11	100	29	DGGL	49	53	83
4	DHSA	52	10	100	17	DSAG	50	16	100	30	DSAGC	48	9	99
5	DSAC	52	12	100	18	SGGC	50	17	100	31	DSSC	48	12	97
6	DGGG	52	23	100	19	AGAG	50	27	95	32	SCSG	48	19	93
7	DHSAC	51	9	100	20	AGGG	50	58	85	33	AGAG	48	19	93
8	SAGC	51	11	100	21	ACGAG	49	4	100	34	SAGG	48	23	88
9	DACG	51	14	100	22	SCGA	49	6	100	35	DSGS	48	23	94
10	HSAC	51	14	100	23	DACS	49	7	100	36	DAAG	48	27	89
11	DHAA	51	18	100	24	DGGS	49	8	100	37	DASG	48	32	87
12	DAAC	51	32	99	25	SACG	49	10	98	38	GGGG	48	71	76
13	DHAAC	50	5	100	26	DSGC	49	15	98					
Papain-like cysteine protease (ID: 54002) N: 24, σ: 18/24, T: 18.4														
1	HCQS	18	2	34	3	WWGS	18	3	44	5	WGSG	18	5	43
2	HCQG	18	3	34	4	WGNS	18	4	44					
Nuclear receptor ligand-binding domain (ID: 48509) N: 23, σ: 17/23, T: 15.3														
1	FQLL	20	21	43	3	DLQF	17	8	39	4	LQLL	17	40	31
2	DLQF	18	7	42										
FAD/NAD-linked reductase (ID: 51943) N : 20 σ: 15/20, T: 90.0														
1	AGGG	17	34	34	2	AGGA	17	91	27					

those motifs, by plotting all residues covered by at least one motif in the structure of a trypsin: 1HJ9, shown in Figure 9. Interestingly we found that all these residues are confined to the vicinity of the catalytic triad of 1HJ9, namely: HIS57-ASP102-SER195, confirming a known fact that the geometry of the catalytic triad and its spatially adjacent residues are rigid, which is probably responsible for functional specificity of the enzyme.

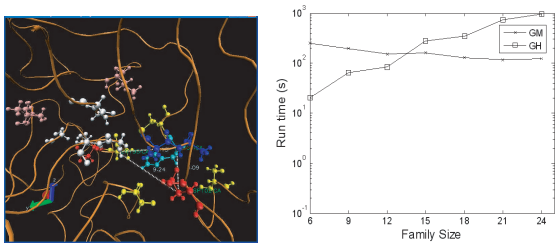

Fig. 9. Left: Spatial distribution of residues found in 38 common structure motifs within protein 1HJ9. The residues of catalytic triad, HIS57-ASP102-SER195, are connected by white dotted lines. Right: Performance comparison of graph mining (GM) and geometric hashing (GH) for structure motif identification.

We found that there are five motifs that occur significantly ($P\text{–}value < 10^{-7}$) in another SCOP family: Prokaryotic Serine Protease (details not shown). This is not surprising since both prokaryotic and eukaryotic serine proteases are quite similar at both structural and functional levels and they share the same SCOP superfamily classification. None of the motif has significant presence outside these two families.

Comparing to our own previous study that uses generic subgraph mining algorithm (without constraints and without utilizing pairwise amino acid residue distance information), and pairwise structural comparison performed by other groups [1, 10, 22, 31, 29, 38], we report a significant improvement of the "precision" of structure motifs. For example, rather than reporting thousands of motifs for a small data set like serine proteases [38, 14], we report a handful of structure motifs that are highly specific to the serine protease family (as measured by low P-values) and highly specific to the catalytic sites of the proteins (as shown in Figure 9).

To further evaluate our algorithm, we randomly sample two proteins from the ESP family and search for common structure motifs. We obtain an average of 2300 motifs per experiment for a total of thousand runs. Such motifs are characterized by poor statistical significance and were not specific to known functional sites in the ESP. If we require a structure motif to appear in at least 24 of a 31 randomly selected ESP proteins and repeat the same experiment, we obtain an average of 65 motifs per experiment with improved statistical significance. This experiment demonstrates that comparing a group of proteins improves the quality of the motifs, as observed by [38].

Beside improved quality of structure motifs, we observe a significant speed up for our structure motif com-

parison algorithm comparing to other methods such as geometric hashing. At the right part of Figure 9, we show performance comparison of graph mining (GM) and geometric hashing (GH)[21] (executable download from the companion website) for serine proteases. We notice a general trend that with the increasing number of proteins structures, the running time of graph mining decreases (since there are fewer common structure motifs) but the running time of geometric hashing increases. The two techniques have different set of parameters that make any direct comparison of running time difficult, however, the trend is very clear that graph mining has better scalability than geometric hashing for data set contains large number of proteins structures.

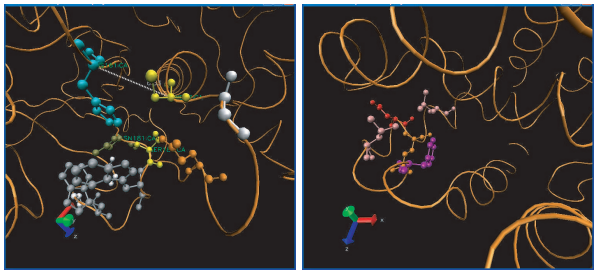

Fig. 10. Left: Residues included in the motifs from PCP family in protein 1CQD. The residues in catalytic dyad CYS27-HIS161 are connected by a white dotted line and two important surrounding residues ASN181 and SER182 are labeled. Right: Residues included in motifs from the NB family in protein 1OVL. The labeled residue GLN 435 has direct interaction with the cofactor of the protein.

5.3. Papain-like cysteine protease and nuclear binding domain

We applied our approach to two additional SCOP families: Papain-Like Cysteine Protease (PCP, ID: 54002) and Nuclear Receptor Ligand-Binding Domain (NB, ID: 48509). The results are documented in the middle part of Table 1.

For the PCP family, we identified five structure motifs which covered the catalytic CYC-HIS dyad and nearby residues ASN and SER which are known to interact with the dyad [7], as shown in Figure 10. For the NB family, we identified four motifs [a] which map to the cofactor binding sites [37], shown in the same figure. In addition, four members missed by SCOP: 1srv, 1khq, and 1o0e were identified for the PCP family and six members 1sj0, 1rkg, 1osh, 1nq7, 1pq9, 1nrl were identified for the NB family.

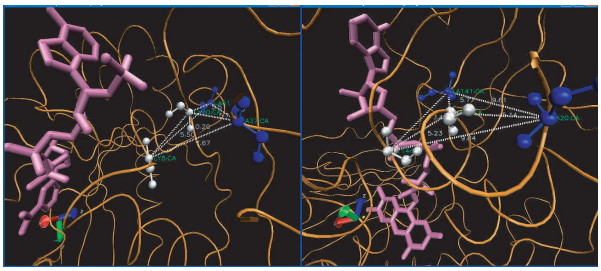

Fig. 11. The motif appears in two proteins 1LVL (belongs to the FAD/NAD-linked reducatase family without Rossman fold) and 1JAY (belongs to the 6-phosphogluconate dehydrogenase-like, N-terminal domain family with Rossman fold) with conserved geometry.

5.4. FAD/NAD binding proteins

In the SCOP database, there are two superfamilies of NADPH binding proteins, the FAD/NAD(P)-binding domains and the NAD(P)-binding Rossmann-fold domains, which share no sequence or fold similarity to each other. This presents a challenging test case for our system to check whether we would be able to find patterns across the two groups with biological significance.

To address the question, we applied our algorithm to the largest family in SCOP FAD/NAD(P)-binding domain: FAD/NAD-linked reductases (SCOPID: 51943). With support threshold 15/20, we obtained two recurring structure motifs from the family, and both showed strong statistical significance in the NAD(P)-binding Rossmann-fold superfamily as shown in bottom part of Table 1.

In Figure 11, we show a motif that is statistically enriched in both families; it has conserved geometry and is interacting with the NADPH molecule in two proteins belonging to the two families. Notice that we do not include any information from NADPH molecule during our search, and we identified this motif due to its strong structural conservation among proteins in a SCOP superfamily. The two proteins have only 16% sequence similarity and adopt different folds (DALI z-score 4.5). The result suggests that significantly common features can be inferred from proteins with no apparent sequence and fold similarity.

[a] Structure motifs 2 and 3 have the same residue composition but they have different residue contact patterns and therefore regarded as two patterns. They do not map to the same set of residues.

5.5. Random proteins

Our last case study is a control experiment to empirically evaluate the statistical significance of the structure motifs regardless of the $P\text{–}value$ definition. To that end, 20 proteins were randomly sampled from the culled PDB list in order to obtain common motifs with support ≥ 15. The parameters 20 and 15 were set up to mimic the size of a typical SCOP family. We repeated the experiment a million times, and did not find a single recurring structure motif. Limited by the available computational resources, we did not test the system further; however, we are convinced that the chance of observing a random structure motif in our system is rather small.

6. CONCLUSION

We present a method to identify recurring structure motifs in a protein family with high statistical significance. This method was applied to selected SCOP families to demonstrate its applicability to finding biologically significant motifs with statistical significance. In future studies, we will apply this approach to all families in SCOP as well as from other classification systems such as Gene Ontology and Enzyme Classification. The accumulation of all significant motifs characteristic of known protein functional and structural families will aid protein structures resulting from structural genomics projects.

References

1. Peter J. Artymiuk, Andrew R. Poirrette, Helen M. Grindley, David W. Rice, and Peter Willett. A graph-theoretic approach to the identification of three-dimensional patterns of amino acid side-chains in protein structures. *Journal of Molecular Biology*, 243:327–44, 94.
2. D. Bandyopadhyay and J. Snoeyink. Almost-Delaunay simplices : Nearest neighbor relations for imprecise points. In *ACM-SIAM Symposium On Distributed Algorithms*, pages 403–412, 2004.
3. JA Barker and JM Thornton. An algorithm for constraint-based structural template matching: application to 3d templates with statistical analysis. *Bioinformatics*, 19(13):1644–9, 2003.
4. H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.
5. Philip Bradley, Peter S. Kim, and Bonnie Berger. TRILOGY: Discovery of sequence-structure patterns across diverse proteins. *Proceedings of the National Academy of Sciences*, 99(13):8500–8505, June 2002.
6. SA Cammer, CW Carter, and A. Tropsha. Identification of sequence-specific tertiary packing motifs in protein structures using delaunay tessellation. *Lecture notes in Computational Science and Engineering,*, 24:477–494, 2002.
7. K. H. Choi, R. A. Laursen, and K. N. Allen. The 2.1 angstrom structure of a cysteine protease with proline specificity from ginger rhizome, zingiber officinale. *Biochemistry*, 7, 38(36):11624–33, 1999.
8. I. Eidhammer, I. Jonassen, and W. R. Taylor. *Protein Bioinformatics: An Algorithmic Approach to Sequence and Structure Analysis*. John Wiley & Sons, Ltd, 2004.
9. JF Gibrat, T Madej, and SH. Bryant. Surprising similarities in structure comparison. *Curr Opin Struct Biol*, (6(3)):683–92, 1996.
10. H.M. Grindley, P.J. Artymiuk, D.W. Rice, and P. Willet. Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. *J. Mol. biol.*, 229:707–721, 1993.
11. H. Hegyi and M. Gerstein. The relationship between protein structure and function: a comprehensive survey with application to the yeast genome. *J Mol Biol*, 288:147–164, 1999.
12. L. Holm and C. Sander. Mapping the protein universe. *Science*, 273:595–602., 1996.
13. J. Hu, X. Shen, Y. Shao, C. Bystroff, and M. J. Zaki. Mining protein contact maps. *2nd BIOKDD Workshop on Data Mining in Bioinformatics*, 2002.
14. J. Huan, W. Wang, D. Bandyopadhyay, J. Snoeyink, J. Prins, and A. Tropsha. Mining protein family specific residue packing patterns from protein structure graphs. In *Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 308–315, 2004.
15. J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, pages 549–552, 2003.
16. William Humphrey, Andrew Dalke, and Klaus Schulten. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14:33–38, 1996.
17. I. Jonassen, I. Eidhammer, D. Conklin, and W. R. Taylor. Structure motif discovery and mining the PDB. *Bioinformatics*, 18:362–367, 2002.
18. I. Jonassen, I. Eidhammer, and W. R. Taylor. Discovery of local packing motifs in protein structures. *Proteins*, 34:206–219, 1999.
19. Susan Jones and Janet M Thornton. Searching for functional sites in protein structures. *Current Opinion in Chemical Biology*, 8:3–7, 2004.
20. Bala Krishnamoorthy and Alexander Tropsha. Development of a four-body statistical pseudo-potential to discriminate native from non-native protein conformations. *Bioinformatics*, 19(12):1540–48, 2003.
21. N. Leibowitz, ZY Fligelman, R. Nussinov, and HJ Wolfson. Automated multiple structure alignment and detection of a common substructural motif. *Proteins*, 43(3):235–45, May 2001.
22. M Milik, S Szalma, and KA. Olszewski. Common structural cliques: a tool for protein structure and function analysis. *Protein Eng.*, 16(8):543–52., 2003.
23. N Nagano, CA Orengo, and JM Thornton. One fold with many functions: the evolutionary relationships between

tim barrel families based on their sequences, structures and functions. *Journal of Molecular Biology*, 321:741–765, 2002.
24. Ruth Nussinov and Haim J. Wolfson. efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. *PNAS*, 88:10495–99, 1991.
25. Robert B. Russell. Detection of protein three-dimensional side-chain patterns: new examples of convergent evolution. *Journal of Molecular Biology*, 279:1211–1227, 1998.
26. S. Schmitt, D. Kuhn, and G. Klebe. A new method to detect related function among proteins independent of sequence and fold homology. *J. Mol. Biol.*, 323(2):387–406, 2002.
27. J. P Shaffer. Multiple hypothesis testing. *Ann. Rev. Psych*, pages 561–584, 1995.
28. Jeffrey Skolnick, Jacquelyn S. Fetrow, and Andrzej Kolinski. Structural genomics and its importance for gene function analysis. *nature biotechnology*, 18:283–287, 2000.
29. R. V. Spriggs, P. J. Artymiuk, and P. Willett. Searching for patterns of amino acids in 3D protein structures. *J Chem Inf Comput Sci*, 43:412–421, 2003.
30. A Stark and RB Russell. Annotation in three dimensions. pints: Patterns in non-homologous tertiary structures. *Nucleic Acids Res*, 31(13):3341–4, 2003.
31. A. Stark, A. Shkumatov, and R. B. Russell. Finding functional sites in structural genomics proteins. *Structure (Camb)*, 12:1405–1412, 2004.
32. William R. Taylor and Inge Jonassen. A method for evaluating structural models using structural patterns. *Proteins*, July 2004.
33. A. Tropsha, C.W. Carter, S. Cammer, and I.I. Vaisman. Simplicial neighborhood analysis of protein packing (SNAPP) : a computational geometry approach to studying proteins. *Methods Enzymol.*, 374:509–544, 2003.
34. J. R. Ullman. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23:31–42, 1976.
35. AC Wallace, N Borkakoti, and JM Thornton. Tess: a geometric hashing algorithm for deriving 3d coordinate templates for searching structural databases. application to enzyme active sites. *Protein Sci*, 6(11):2308–23, 1997.
36. G. Wang and R. L. Dunbrack. PISCES: a protein sequence culling server. *Bioinformatics*, 19:1589-1591, 2003. http://www.fccc.edu/research/labs/dunbrack/pisces/culledpdb.html.
37. Z. Wang, G. Benoit, J. Liu, S. Prasad, P. Aarnisalo, X. Liu, H. Xu, NP. Walker, and T. Perlmann. Structure and function of nurr1 identifies a class of ligand-independent nuclear receptors. *Nature*, 423(3):555–60, 2003.
38. PP Wangikar, AV Tendulkar, S Ramya, DN Mali, and S Sarawagi. Functional sites in protein families uncovered via an objective and automated graph theoretic approach. *J Mol Biol*, 326(3):955–78, 2003.
39. X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proc. International Conference on Data Mining'02*, pages 721–724, 2002.

7. APPENDIX

7.1. Statistical significance of structure motifs

Any cliques that are frequent in a SCOP family are checked against a data set of 6500 representative proteins from CulledPDB [36], selected from all proteins in the Protein Data Bank. For each clique c, we used Ullman's subgraph isomorphism algorithm [34] to search for its occurrence(s) and record the search result in an *occurrence vector* $V = v_1, v_2, \ldots, v_n$, where v_i is 1 if c occurs in the protein p_i, and 0, otherwise. Such cliques are referred to as structure motifs. We determine the statistical significance of a structure motif by computing the related P-value, defined by a hyper-geometric distribution [5]. There are three parameters in our statistical significance formula: a collection of representative proteins M, which stands for all known structures in PDB; a subset of proteins $T \subseteq M$ in which a structure motif m occurs, a subset of proteins $F \subseteq M$ stands for the family we would like to establish the statistical significance. The probability of observing a set of motif m containing proteins $K = F \cap T$ with size at least k is given by the following formula:

$$P\text{–}value = 1 - \sum_{i=0}^{k-1} \frac{\binom{|F|}{i}\binom{|M|-|F|}{|T|-i}}{\binom{|M|}{|T|}}. \quad (1)$$

where $|X|$ is the cardinality of a set X. For example, if a motif m occurs in every member of a family F and in no proteins outside F (i.e. $K = F = T$) for a large family F, we would estimate that this motif is specifically associated with the family; the statistical significance of such case is measured by a $P\text{–}value$ close to zero.

We adopt the Bonferroni correction for multiple independent hypotheses [27]: $0.001/|C|$, where $|C|$ is the set of categories, is used as the default threshold to measure the significance of the P-value of individual test. Since the total number of SCOP families is 2327, a good starting point of P-value upper bound is 10^{-7}.

7.2. Background frequency

Using the culledpdb list (http://www.fccc.edu/research/labs/dunbrack/pisces/culledpdb.html) as discussed in Section 5.1, we obtain around 6000 proteins as the "representative proteins" in PDB. We treat the proteins as a sample from PDB and for each motif, we estimate its background frequency (the number of occurrences in proteins) using graph matching. Specifically, each sample protein is transformed to its graph representation using the procedure outline in Section 3 and we use subgraph isomorphism testing to obtain the total number of proteins the motif occurs in.