BMC Bioinformatics

BioMed Central

Research

Open Access

Towards comprehensive structural motif mining for better fold annotation in the "twilight zone" of sequence dissimilarity

Yi Jia[1], Jun Huan*[1], Vincent Buhr[1], Jintao Zhang[2] and Leonidas N Carayannopoulos[3]

Address: [1]Department of Electrical Engineering & Computer Science, University of Kansas, Lawrence, KS, 66045, USA, [2]Department of Molecular Biosciences, The University of Kansas, Lawrence, KS 66046, USA and [3]School of Medicine, Washington University in St. Louis, St. Louis, MO 63130, USA

Email: Yi Jia - jiayi@ittc.ku.edu; Jun Huan* - jhuan@ittc.ku.edu; Vincent Buhr - vbuhr@ittc.ku.edu; Jintao Zhang - jtzhang@ittc.ku.edu; Leonidas N Carayannopoulos - lcarayan@im.wustl.edu

* Corresponding author

from The Seventh Asia Pacific Bioinformatics Conference (APBC 2009)
Beijing, China. 13–16 January 2009

Published: 30 January 2009

BMC Bioinformatics 2009, **10**(Suppl 1):S46 doi:10.1186/1471-2105-10-S1-S46

This article is available from: http://www.biomedcentral.com/1471-2105/10/S1/S46

Abstract

Background: Automatic identification of structure fingerprints from a group of diverse protein structures is challenging, especially for proteins whose divergent amino acid sequences may fall into the "twilight-" or "midnight-" zones where pair-wise sequence identities to known sequences fall below 25% and sequence-based functional annotations often fail.

Results: Here we report a novel graph database mining method and demonstrate its application to protein structure pattern identification and structure classification. The biologic motivation of our study is to recognize common structure patterns in "immunoevasins", proteins mediating virus evasion of host immune defense. Our experimental study, using both viral and non-viral proteins, demonstrates the efficiency and efficacy of the proposed method.

Conclusion: We present a theoretic framework, offer a practical software implementation for incorporating prior domain knowledge, such as substitution matrices as studied here, and devise an efficient algorithm to identify approximate matched frequent subgraphs. By doing so, we significantly expanded the analytical power of sophisticated data mining algorithms in dealing with large volume of complicated and noisy protein structure data. And without loss of generality, choice of appropriate compatibility matrices allows our method to be easily employed in domains where subgraph labels have some uncertainty.

Background

Genomics efforts continue to yield a myriad of new protein sequences. Among the most valuable are those expressed by mammalian pathogens, organisms that successfully grow and disseminate despite a hostile host immunologic environment. A subset of pathogen-encoded proteins, "immunoevasins", facilitate this success by mediating cellular adhesion and entry, and by dis-

torting the interactions of host receptors and cell-surface ligands [1]. Study of immunoevasins gives insight into host-defense mechanisms, insight that can help guide development of therapies and vaccines against refractory organisms [2].

Though immunoevasins frequently possess protein-recognition domain (PRD) folds common to mammalian proteins of immunologic importance, their divergent amino acid sequences may fall into the "twilight-" or "midnight-" zones where pair-wise sequence identities to known sequences fall below 25% and purely sequence-based attempts at annotations often fail [3,4].

To better annotate these, and any other highly divergent sequences, more generally, some means of explicitly incorporating three-dimensional structural information into the sequence evaluation is required. Inclusion of even rudimentary structural considerations enhances the performance of sequence scoring heuristics such as local alignment tools [5] and hidden Markov models (HMM) [6]. Indeed an HMM constrained with crystallographically determined secondary structure data allowed discovery of a previously unsuspected MHC class I-like immunoevasin in the genomes of orthopoxviruses [7]. A vast literature covers various schemes for structural data incorporation and fold classification. Nevertheless, much progress remains to be made [8].

We are pursuing an approach whereby structural patterns common to a protein fold are collected, assessed for their classification value, and mapped onto statistical models of protein sequences (e.g. HMMs, support vector machines (SVMs), and conditional random fields). As a first step, a comprehensive and objective means is required of identifying and assessing the above common structure patterns, or structure fingerprints.

Automatic identification of structure fingerprints from a group of diverse protein structures is challenging for a number of reasons. First, we have only limited knowledge about the possible location, composition, and geometric shape of these structure patterns. Second, protein structures are large geometric objects that typically contain hundreds of amino acids with thousands of atoms and chemical bonds. Third, due to accumulated mutations in evolution the same structure pattern may appear slightly different in different proteins. If we use terms from computer algorithm design, we say that the problem of automatic structure pattern identification is challenging since (1) the problem has a large combinatory search space (meaning patterns may occur in any part of a protein and in any subset of a group of proteins) and (2) we should use approximate matching rather than exact matching in retrieving such patterns (meaning that we should tolerate certain level of geometric distortion and amino acid mismatch in search for common structure patterns).

In this paper we demonstrate a novel data mining technique that efficiently extracts and scores structure pattern from diverse proteins. Specifically in our method, we encode a protein structure as a geometric graph where a node represents an amino acid residue and an edge represents a physical or a chemical interaction between a pair of residues. We encode structural motifs as subgraphs of a geometric graph and we identify conserved structure fingerprints by searching for frequently occurring approximately subgraphs in a group of graph represented proteins.

Our contributions in designing a new graph data mining method are to develop a solid theoretic framework, to offer a practical software implementation for incorporating prior domain knowledge, such as substitution matrices as studied here, and to devise an efficient algorithm to identify approximate matched frequent subgraphs. By doing so, we expanded the analytical power of data mining algorithms in dealing with large volume of complicated and noisy protein structure data. As evaluated in our driving biological application of recognizing common structure patterns in immunoevasins, our proposed method identifies many structure patterns and affords better structure classification accuracy compared to existing graph mining algorithms.

The rest of the paper is organized in the following way. In the *Related Work* section, we give an overview of related work on subgraph mining and protein structure pattern identification. In the *Methods* section, we introduce the technique about how to translate protein structures into graphs, provide our model for approximate subgraph mining, and present the details of our algorithm. In the *Results* section, we show an empirical study of the proposed algorithm using protein structure data sets. In the *Discussion* section, we discuss the biological significance of the structural motifs mined by our method. Finally in the *Conclusions* section, we conclude with a short discussion of our approach.

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 [10]. 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 [11-13].

• Geometric hashing, originally developed in computer vision, applied pairwise between protein structures to identify structure motifs [14-16].

• 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 [17-19].

• Delaunay Tessellation (DT) [20-22] partitioning the structure into an aggregate of non-overlapping, irregular tetrahedra thus identifying all unique nearest neighbor residue quadruplets for any protein [22].

• Graph matching methods comparing protein structures modeled as graphs and discovering structure motifs by finding recurring subgraphs [23-29].

Graph database mining is an active research field in data mining research. The goal of graph database mining is to locate useful and interpretable patterns in a large volume of graph data. Recent exact matching graph mining algorithms can be roughly divided into three categories. The first category uses the level-wise search strategy, which includes AGM [30] and FSG [31]. And the second category takes the depth-first search strategy, which includes gSpan [32] and FFSM [33]. The third category works by mining frequent trees, for which SPIN [34] and GASTON [35] are the representative. There are many other existing graph mining algorithms, and we refer to [36] for a recent survey.

Frequent subgraph mining with approximate matching capability has also been investigated. The current approximate subgraph mining algorithms can be divided into four categories: (1) proximity measures between graphs [37-39], (2) given a proximity measurement, compute representative frequent subgraphs [40], (3) pattern discovery in a single large graph [41], and (4) pattern discovery from a group of graphs. The last category is what we concentrate on. For algorithms in (4), SUBDUE [42] does not claim completeness. Monkey [43] handles only edge missing and edge label mismatch. Partially Labeled Graphs [44] uses a wild card method to handle node label mismatches. The algorithm may be viewed as a special case of our algorithm.

Different from the existing work, to our best knowledge, we are the first group that incorporates a probability matrix in a graph mining method. We also developed a general framework to fully utilize a probability matrix for approximate match, which we can apply to a number of different applications. In addition, we have developed two ways to demonstrate the statistical significance of the patterns mined from a graph database. Statistical significance is an important but often overlooked issue in evaluating the quality of identified pattern in frequent pattern mining. Finally we offered a practical implementation and evaluated its performance using the synthetic sets.

Methods

In this section, we first briefly describe the technique that translates protein structures into graphs. Then we demonstrate our method called **APGM**(**AP**proximate **G**raph **M**ining) with two steps: introducing the theoretic model, and showing our algorithm in detail.

Almost-Delaunay graph

Since the protein backbone trace defines the overall protein conformation, we choose the C_α atoms as the nodes of protein graphs. Based on this simplified protein model, we compute edges using Almost-Delaunay Tesselation [45]. The Almost-Delaunay edges are a superset of the Delaunay edges. All nearest neighbor residues connected by Delaunay edges are defined using Delaunay Tessellation [46]. This tessellation is defined for a finite set of points by an empty sphere property: A pair of points is joined by an edge iff one can find an empty sphere whose boundary contains those two points. The definition of the Delaunay Tessellation depends on the precise coordinate values given to its points, but these coordinate values are not exact in the case of proteins due to measurement imprecision and atomic motions. In order to address this problem, Almost-Delaunay Edges are defined by relaxing the empty sphere property to say that a pair of points p and q is joined by an Almost-Delaunay edge with parameter ε, or AD(ε), if by perturbing all points by at most ε, p and q can be made to lie on an empty sphere. In Figure 1, we show one segment of the 3D structure and the corresponding AD graph of 1FP5A Immunoglobulin C1-type protein as an example. More detailed information is available in [45] and [47].

Theoretic framework

Definition 1

*A **labeled graph** G is a 5-tuple* $G = \{V, E, \Sigma_V, \Sigma_E, \lambda\}$ *where V is the set of vertices of G and* $E \subseteq V \times V$ *is the set of undirected edges of G.* Σ_V *and* Σ_E *are (disjoint) sets of labels. And labeling function* $\lambda: V \to \Sigma_V \cup E \to \Sigma_E$ *maps vertices and edges in G to their labels. A **graph database** D is a set of graphs.*

We also use $V[G]$ to denote the node set of a graph G and $E[G]$ to denote the edge set of G. We also use $\Sigma_{V[G]}$ to denote the node labels, $\Sigma_{E[G]}$ to denote edge labels, and λ_G to denote the labeling function for a graph G. Before we introduce approximate matching, we define compatibility matrix, which offers a probability framework for approximate subgraph mining.

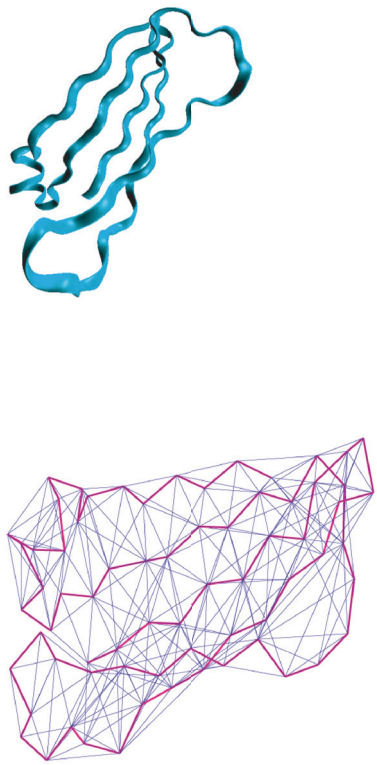


Figure 1
3D structure and corresponding graph of one sample protein. **Upper:** One segment of the 3D structure of the 1FP5A Immunoglobulin C1-type protein (the paired Fcε3 and 4 domains of IgE). **Lower:** The corresponding graph. Vertices are C_α atoms. Covalent edges are represented in heavy magenta while non-covalent edges defined by Almost Delaunay Tesselation(ε = 0.1) appear in thin blue.

Definition 2

*A **compatibility matrix** $M = (m_{i,j})$ is an $n \times n$ matrix indexed by symbols from a label set Σ ($n = |\Sigma|$). An entry $m_{i,j} (0 \le m_{i,j} \le 1, \Sigma_j m_{i,j} = 1)$ in M is the probability that the label i is replaced by the label j.*

A compatibility matrix M is *stable* if the diagonal entry is the largest one in the row (i.e. $M_{i,i} > M_{i,j'}$ for all $j \neq i$). A compatibility matrix being stable means that any label i is more likely to be replaced by itself rather than by any other symbol. For our biological application, we consider substitution matrices as being, in essence, stable matrices since most or all rows fit the criterion. For example, in the BLOSUM62 substitution matrix, there is only one violation of the criterion – the row for methionine(MET). Hence for the rest of the discussion, we will treat substitution matrices as stable compatibility matrices.

Example 1. *We show a graph database D with three labeled graphs P, Q, R on the left side of* Figure 2. *In this database, the node label set is {a, b, c} and the edge label set is {x, y}. On the right part of* Figure 2, *we show a compatibility matrix M, which is a 2D matrix indexed by the set of node labels in D. The probability that the vertex label a is substituted by b is $m_{a,b}$ = 0.3. In M, we use probability 0 to simplify the matrix. In reality these probabilities are never 0.*

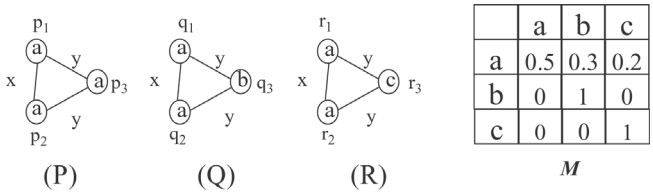
Definition 3

*A labeled graph $G = \{V, E, \Sigma_V, \Sigma_E, \lambda\}$ is **approximately subgraph isomorphic** to another graph $G' = \{V', E', \Sigma'_V, \Sigma'_E, \lambda'\}$ if there exists an injection $f : V \to V'$ such that*

- $\prod_{u \in V} M_{\lambda(u),\lambda'(f(u))} \ge \tau$, *and*
- $\prod_{(u,v) \in E} M'_{\lambda(u,v),\lambda'(f(u),f(v))} \ge \tau'$

The injection f is an *approximate subgraph isomorphism* between G and G'. M is a compatibility matrix for node label sets $\Sigma_V \cup \Sigma'_V$. M' is a compatibility matrix for edge label sets $\Sigma_E \cup \Sigma'_E$. In an edge compatibility matrix, we assume Σ_E and Σ'_E both contain a special label called empty edge. In this way, we handle both topology distortion (missing edges) and edge label mismatches in the same unified way through an edge compatibility matrix. τ $(0 < \tau \le 1)$ is the threshold for node mismatch and $\tau'(0 < \tau' \le 1)$ is the threshold for edge mismatch.

For simplicity in the following discussion, we assume that we only need to handle node label mismatches (i.e. corresponding edge relations and corresponding edge labels should exactly match each other in matching two graphs). In principle, edge label mismatch (including missing edges) can be handled in a similar way as node label mismatch. Hence our assumption does not reduce the complexity of algorithm design, but the assumption significantly simplifies our demonstration and makes our algorithm easy of access.



	a	b	c
a	0.5	0.3	0.2
b	0	1	0
c	0	0	1

M

Figure 2
Graph database and compatibility matrix. Example of a graph database *D* and a compatibility matrix *M*.

With the assumption, the new definition of approximate subgraph isomorphism is:

Definition 4

A graph G is ***approximate subgraph isomorphic*** *to another graph G', denoted by* $G \subseteq_a G'$ *if there exists a 1-1 injection f V*[G] *to V*[G']*, such that*

- $\prod_{u \in V} M_{\lambda(u),\lambda'(f(u))} \geq \tau$,
- $\forall\ u, v \in V, (u, v) \in E \Leftrightarrow (f(u), f(v)) \in E'$, *and*
- $\forall\ (u, v) \in E, \lambda(u, v) = \lambda(f(u), f(v))$

Given a node injection f from graph G to G', the co-domain of f is an *embedding* of G in G'. M is a compatibility matrix for node label sets $\Sigma_V \cup \Sigma'_V$. The *approximate subgraph isomorphism score* of f, denoted by $S_f(G, G')$, is the product of normalized probabilities: $S_f(G, G') = \prod \frac{M_{\lambda(u),\lambda'(f(u))}}{M_{\lambda(u),\lambda(u)}}$. For the case of exception in mutation matrix, we use $MAX(M_{\lambda(u),*})$ as the normalizing factor instead of $M_{\lambda(u),\lambda(u)}$. For a pair of graphs, there may be many different ways of mapping nodes from one graph to another and hence may have different approximate isomorphism scores. The *approximate matching score* (score for simplicity) between two graphs, denoted by $S(G, G')$, is the largest approximate subgraph isomorphism score, or

$$S(G, G') = \max_f \{S_f(G, G')\}$$

Similarly, we define exact subgraph isomorphism below.

Definition 5

A graph G is ***subgraph isomorphic*** *to another graph G', denoted by* $G \subseteq G'$ *if there exists a 1-1 injection f from the node set V of a graph G to V' of a graph G', such that*

- $\forall\ u \in V, \lambda(u) = \lambda'(f(u))$
- $\forall\ u, v \in V, (u, v) \in E \Leftrightarrow (f(u), f(v)) \in E'$, *and*
- $\forall\ (u, v) \in E, \lambda(u, v) = \lambda(f(u), f(v))$

Example 2. *In* Figure 2*, we show a graph database* $D = \{P, Q, R\}$ *and a compatibility matrix M. We set isomorphism threshold* $\tau = 0.4$ *and with this threshold, graph P is approximate subgraph isomorphic to graph Q with the approximate subgraph isomorphic score equaling* 0.6*. To see this, there are a total of 6 different ways to map nodes of P to those of Q. The only two that satisfy edge label constraints are* $f_1 = p_1 \rightarrow q_1\ p_2 \rightarrow q_2\ p_3 \rightarrow q_3$ *and* $f_2 = p_1 \rightarrow q_2\ p_2 \rightarrow q_1\ p_3 \rightarrow q_3$*. The approximate subgraph isomorphism score of* f_1 *equals that of* f_2*.*

Definition 6

Given a graph database D, an isomorphism threshold τ*, a support threshold* $\sigma\ (0 < \sigma \leq 1)$*, the* ***support value*** *of a graph G, denoted by* sup_G*, is the average score of the graph to graphs in the database:*

$$supG = \sum_{G' \in D, G \subseteq_a G'} S(G, G') / |D| \quad (1)$$

G is a *frequent approximate subgraph* if its support value is at least σ. With this definition, we only use those graphs that a subgraph G is approximate subgraph isomorphic to (controlled by the parameter τ) to compute the support value of G. We do this to filter out low quality (but potentially many) graph matchings in counting the support value of a subgraph. For a moderate sized graph database (100 1000), according our experience, the number of frequent subgraphs identified is usually not sensitive to the isomorphism threshold, which makes sense since low quality graph matching has low "weight" in the support computation nevertheless.

Problem statement

Given a graph database D, an isomorphism threshold τ, a compatibility matrix M, and a support threshold σ, the **approximate subgraph mining** problem is to find all the frequent approximate subgraphs in D. In Figure 3, we show all the frequent approximate subgraphs in the graph database D shown in Figure 2. By comparison with the frequent subgraphs acquired by the exact graph mining, the approximate mining method identifies meaningful patterns that cannot be identified by exact graph mining

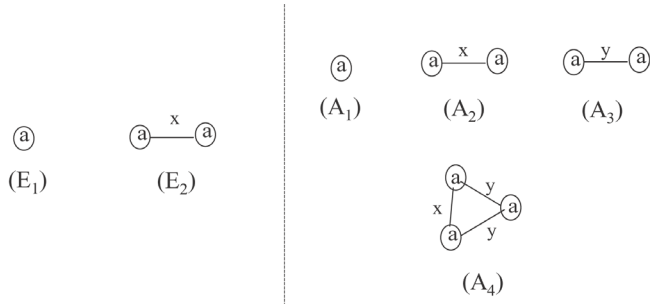


Figure 3
Example of frequent subgraphs and approximate frequent subgraphs. Given the graph database D in *Figure* 2 and the support threshold $\sigma = 2/3$,the left side shows the frequent subgraphs mined by the general exact graph mining. Given the compatibility matrix M in *Figure* 2, isomorphism threshold $\tau = 0{:}4$, and support threshold $\sigma = 2/3$. The right side presents the frequent approximate subgraphs in D.

methods. Since the support value of approximate subgraph mining and that of frequent subgraph mining have different meaning, it is generally hard to do a comparison of approximate subgraph mining and that of frequent subgraph mining. Fortunately with the assumption of stable compatibility matrix, we can see frequent subgraph mining as a special case of approximate subgraph mining.

Example 3. *Given a graph database D, a compatibility matrix M in* Figure 2, *the support threshold* $\sigma = 2/3$ *and isomorphism threshold* $\tau = 0{:}4$, *we show how to calculate the isomorphism score and support value for the approximate frequent patterns in* Figure 3.

$S(A_1, P) = 1$, $S(A_1, Q) = 1$, $S(A_1, R) = 1$, $Sup(A_1) = 3/3$;

$S(A_2, P) = 1$, $S(A_2, Q) = 1$, $S(A_2, R) = 1$, $Sup(A_2) = 3/3$;

$S(A_3, P) = 1$, $S(A_3, Q) = 0.6$, $S(A_3, R) = 0.4$, $Sup(A_3) = 2/3$;

$S(A_4, P) = 1$, $S(A_4, Q) = 0.6$, $S(A_4, R) = 0.4$, $Sup(A_4) = 2/3$.

Algorithm design

Here we demonstrate a new algorithm APGM for approximate subgraph mining. APGM starts with frequent single node subgraphs. At a subsequent step, it adds a node to an existing pattern to create new subgraph patterns and identify their support value. If none of the resulting subgraphs are frequent, APGM backtracks. APGM stops when no more patterns need to be searched. Before we proceed to the algorithmic details, we introduce the following definitions to facilitate the demonstration of the APGM algorithm.

Definition 7

Given a graph T, one of the embeddings $e = v_1, v_2, \cup, v_k$ *of T, a node v is a* ***neighbor*** *of e if* $\exists u \in e$, $(u, v) \in E[G]$.

In other words, a neighbor node of a embedding *e* is any node that connects to at least one node in *e*. The *neighbor set* of an embedding *e*, denoted by $N(e)$, is the set of *e*'s neighbors.

Definition 8

Given a graph T, one of the embeddings $e = v_1, v_2, \cup, v_k$ *of T in a graph G, a node* $v \in N(e)$, *and a node label l, the* ***approximate subgraph****, denoted by* $G|_{T,e,v,l}$, *is a graph* $(V', E', \Sigma'_V, \Sigma'_E, \lambda')$ *such that*

- $V' = \{v_1, v_2, \cup, v_k\} \cup v$
- $E' = V' \times V' \cap E[G]$
- $\Sigma'_V = \Sigma_V$
- $\Sigma'_E = \Sigma_E$
- $\forall\, u \in e : \lambda'(u) = \lambda_T(u)$
- $\lambda'(v) = l$
- $\forall\, u, v \in e : \lambda'((u, v)) = \lambda_G((u, v))$

Example 4. *In* Figure 4, *we show a pattern T and one of its embeddings* $e = (s_1, s_2)$ *in a graph Q. Node* s_3 *is a* neighbor node *of e since it connects to at least one node of e (in fact both). Given a node label* $l =$"a", *we obtain an approximate subgraph* $G' = Q|_{T,e,v,l}$ *of Q shown in the same figure. The G' has an embedding* $e' = (s_1, s_2, s_3)$ *in Q and the score of the embedding is* $\frac{M(a,a)}{M(a,a)}\frac{M(a,a)}{M(a,a)}\frac{M(a,b)}{M(a,a)} = \frac{M(a,b)}{M(a,a)} = 0.6$. *(Recall the score of an embedding is the multiplication of the probability of observed node label replacement, normalized by the probability of node label self-replacement.)*

With the two definitions, we present the pseudo code of APGM below. follows.

Algorithm 1. *APGM_MAIN(D, M,* τ, σ*)*

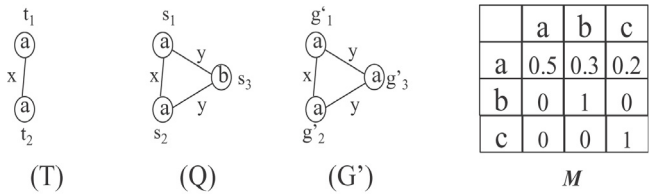
1: Begin

2: C ← {*frequent single node*}

3: F ← *C*

4: ***for*** *each* $T \in C$ ***do***

5: *APGM_SEARCH(T,* τ, σ, *F)*



	a	b	c
a	0.5	0.3	0.2
b	0	1	0
c	0	0	1

M

Figure 4
Approximate subgraph. A pattern *T*, a graph *Q* and approximate subgraph *G'* of *Q*.

6: ***end for***

7: return F

8: End

Algorithm 2. *APGM_SEARCH(T, τ, σ, F)*

1: Begin

2: $C \leftarrow \emptyset$

3: ***for*** *each* (e, v), *e is an embedding of T in G,* $v \in N(e)$ ***do***

4: $CL \leftarrow approximateLabelSet(T, G, e, v)$

5: ***for*** *each* $l \in CL$ ***do***

6: $X \leftarrow G|_{T, e, v, l}$

7: $C \leftarrow C \cup \{X\}$

8: $\mathcal{H}(X) = \mathcal{H}(X) \cup (e, v)$

9: ***end for***

10: ***end for***

11: remove infrequent T from C

12: $F \leftarrow F \cup C$

13: ***for*** *each* $T \in C$ ***do***

14: $APGM\_SEARCH(T, \tau, \sigma, F)$

15: ***end for***

16: End

$\mathcal{H}$ is a hash function to store candidate subgraphs and their embeddings. The hash key of the function in our implementation is a canonical code of the subgraph *X*, which is a unique string presentation of a graph. We use the Canonical Adjacency matrix (CAM) and the Canonical Adjacency Matrix code, developed in [48], to compute the canonical code of a graph.

Algorithm 3. *approximateLabelSet(T, G, e, v)*

1: Begin

2: $R \leftarrow \emptyset$

3: $l_0 \leftarrow \lambda_G(v)$

4: ***for*** *each* $l \in \Sigma_{V[G]}$ ***do***

5: ***if*** $S(e, T) \times \frac{M(l_0, l)}{M(l_0, l_0)} \geq \tau$ ***then***

6: $R \leftarrow R \cup l$

7: ***end if***

8: ***end for***

9: return R

10: End

Example 5. *Applying APGM to the graph database shown in* Figure 2*with the support threshold* $\sigma = 2/3$ *and the isomorphism threshold* $\tau = 0.4$, *we identify one frequent single-node pattern a (shown as* A_1 *in* Figure 3*). Adding one node to the pattern* A_1, *there are two candidate single-edge patterns and both of them are frequent. These two are shown as* A_2 *and* A_3 *in the same figure. From pattern* A_2, *we enumerate one additional pattern* A_4. *We stop here since there is no more candidate patterns to explore.*

Results

Experimental setup

We performed all the experiments on a cluster with 256 Intel Xeon 3.2 Ghz EM64T processors with 4 GB memory each. The approximate graph mining algorithm was

Table 1: Characteristics of domain sequence sets

	Immunoglobulin C1 Set	Immunoglobulin V Set
Number of Proteins	1786	371
Average Length	210	194
Maximum Length	457	444
Minimum Length	98	99

Table 2: Immunoevasins protein lists for research

	PDB ID of proteins in Immunoglobulin C1 set
Proteins for Feature Extraction(10):	*1fp5a 1onqa 1ogad 1pqza 1t7va 1l6xa 1je6a 1mjul 1uvqb 1dn0b*
Proteins for Leave-one-out Testing(11):	*1nfda 1uvqa 1q0xl 1mjuh 1a6za 1k5na 1hdma 3frua 1ogae 1hdmb 1k5nb*
	PDB ID of proteins in Immunoglobulin V set
Proteins for Feature Extraction(10):	*1pkoa 1ogad 1npua 1cdca 1jmaa 1fo0b 1nkoa 1mjuh 1nfdb 1qfoa*
Proteins for Leave-one-out Testing(9):	*1zcza 1f97a 1eaja 1mjul 1cida 1neua 1cdya 1hkfa 1nezg*

implemented in the C++ language and compiled by using the g++ compiler in Linux environment with -O3 optimization.

We downloaded all protein structures from Protein Data Bank (PDB). We followed [45] to use the same software as [47] to calculate Almost-Delaunay(AD) for graph representation of protein geometry. We took BLOSUM62 as the compatibility matrix and back-calculated the conditional probability matrix by following the procedure described in [49]. We normalized the matrix according to *Definition*4.

Data set

We investigated two immunologically relevant protein domain families: the Immunoglobulin V set and the Immunoglobulin C1 set. Immunoglobulin domains are among those used by immunoevasins [50,51]. We collected proteins from SCOP release 1.69. For each family we created a culled set of proteins with maximal pairwise sequence identity percentage below some threshold by using PISCES server [52](Immunoglobulin C1 set below 40%, and Immunoglobulin V set below 30%). The characteristics of the complete domain sequence sets are shown in Table 1. And the PDB IDs of individual proteins for the two culled sets are shown in Table 2.

Experimental protocol

We randomly divided proteins from each family into two groups: 10 proteins to serve as sources for feature extraction, and the remainder(positive sample) for training and testing in "leave-one-out" cross validation. A negative sample set of the the same size as the positive sample set was randomly chosen from PDB. The negative sample was used along with the positive sample in testing. The complete flowchart of our experiment procedure is shown in Figure 5. During this experimental research, we mined frequent clique subgraphs [53] in order to enforce biological constraints on the patterns. We compared APGM with the exact graph mining methods MGM [53]. We chose MGM as the counterpart for the comparison because it is an available clique pattern mining algorithm. (Any exact match method with clique constraint should provide the same number of patterns from a graph database.)

Number of patterns identified

We identified frequent approximate subgraph patterns from 10 positive proteins in each family. There are two parameters that may have significant influence on the set of mined patterns. The first is the support threshold(σ) and the second is the isomorphism threshold(τ). For simplicity, in following experiments in this section we use the new support threshold $\sigma' = \sigma \times |D|$, $|D|$ is the size of graph database, and the same change applied in support value. In Figure 6, we run APGM with different combinations of τ and σ and collect the total number of identified patterns. Our results show that the total number of patterns is not sensitive to the isomorphism threshold, and rather depends on the support threshold heavily. Such fact eases the worry that the parameter τ may be too strong for deciding the number of patterns.

For the purpose of comparison, the number of patterns mined by two mining methods are shown in Table 3 and 4, and the number of patterns acquired by APGM from Immunoglobulin C1 proteins are also shown in Figure 6. In our experiment, we treat a pattern set with the number more than 10000 as a meaningless one because our sample space is comparatively small and the isomorphism check is computationally expensive. From Table 4, we see that exact match fails to provide useful patterns on the Immunoglobulin V proteins, which is the typical data set with very noisy background. In comparison, APGM does find some pattern set with a reasonable size in such situation. (We only use rough parameter combination grids to do the pattern search. If we increase the precision of τ and σ, more patterns will be found.) In order to evaluate the quality of these patterns, we use the identified frequent subgraphs in classification tests as discussed below.

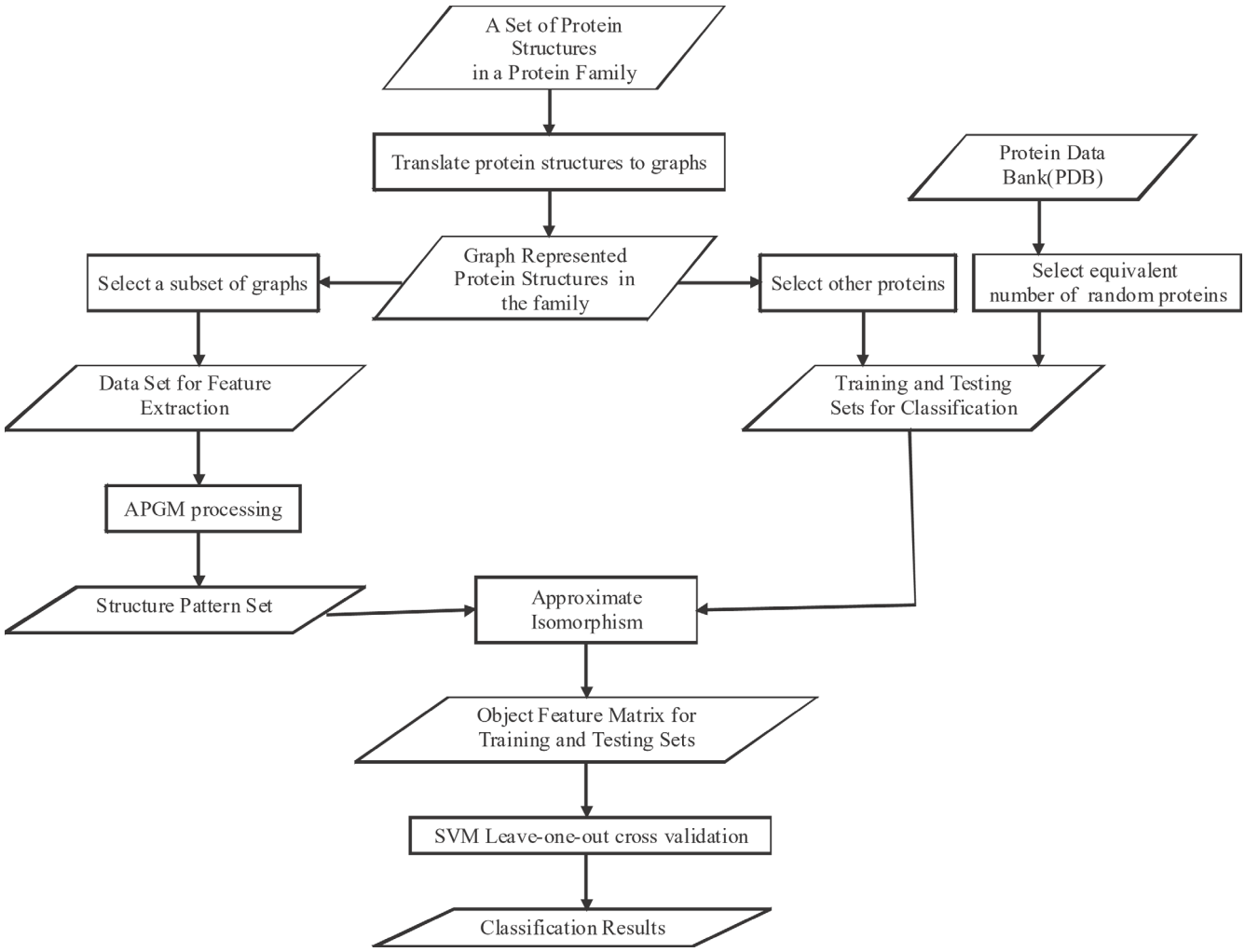


Figure 5
The procedure of experimental research.

Classification performance

In this experimental section, we used *libsvm* SVM package [54] for protein structure classification. We treat each mined pattern as a feature and a protein is represented as a feature vector $V = (v_i)$ where $i \le i \le n$ and n is the total number of identified features. v_i is 1, if the related feature occurs in the protein and otherwise v_i is 0. We used the linear kernel and default parameters for SVM leave-one-out cross validation. The classification results are summarized in Table 5 and 6. For some parameter combinations, there are no accuracies – an event which happens under two circumstances. First, there are no patterns found. Second, the pattern set is too big to be useful. From the tables we see that the classifications with APGM-based feature highly outperform those based on exact match. For Immunoglobulin C1 set, the classification based on feature identified by MGM only can reach 73%, while APGM is between 69%~91%. For Immunoglobulin V set, since the exact match method cannot mine any meaningful patterns, it fails in classification, while by using APGM, we have the accuracy around 78%. This shows that our APGM has more capability to mine useful structure information from very noisy background than general exact match graph mining algorithms.

Statistical significance of patterns

In order to further demonstrate the quality of the patterns mined by using APGM, we chose the parameter combination with the best accuracy for the Immunoglobulin C1 proteins and the Immunoglobulin V proteins to check the distribution and significance of patterns. Figure 7 shows the number of the patterns that the 11 Immunoglobulin C1 proteins contain and the significance scores. Figure 8 shows those for the 9 Immunoglobulin V proteins. Pro-

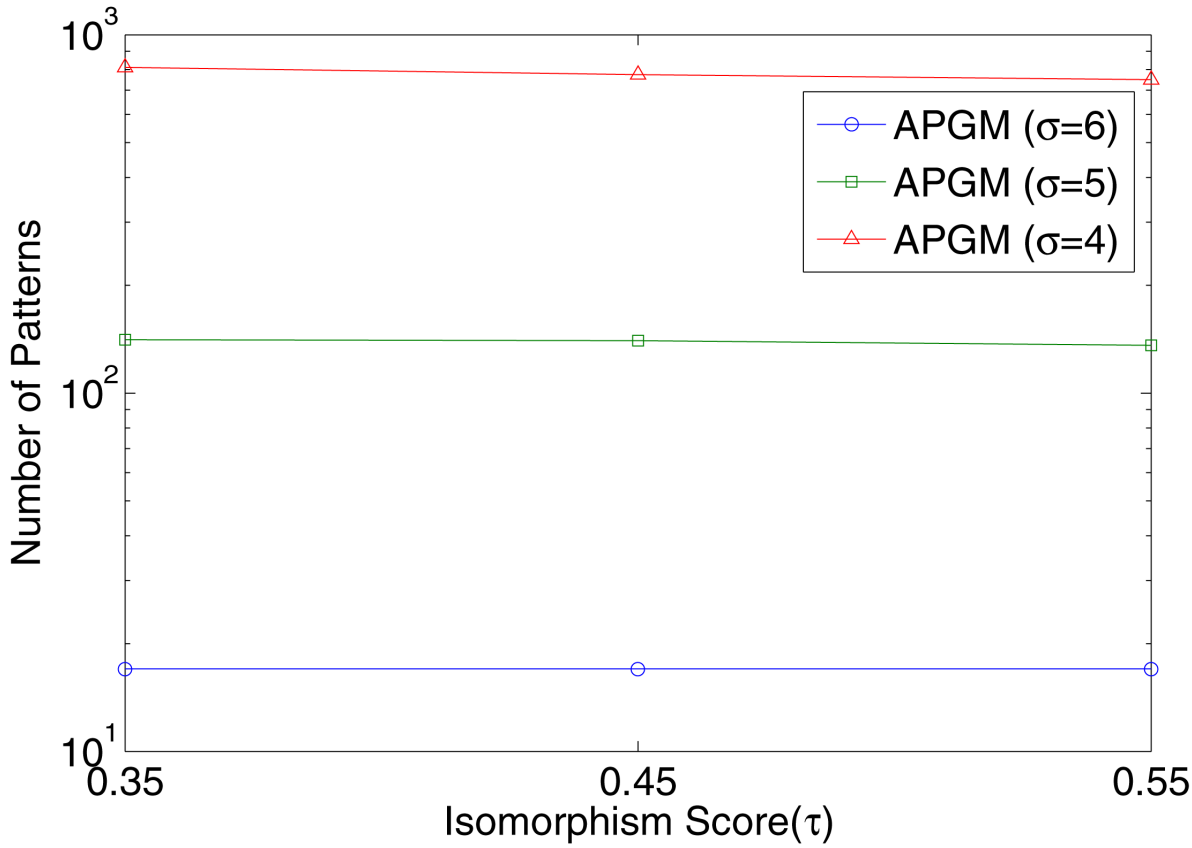


Figure 6
Number of patterns for Immunoglobulin C1 set acquired by APGM. Example of a graph database *D* and a compatibility matrix *M*.

teins in Figure 7 and 8 are numbered according to their appearance order in in Table 2. For example protein "10" in Figure 7 is protein 1nfa (chain A). The proteins in Figure 7 and 8 are sorted according to the number of patterns contained in the proteins. The significance score *P* is defined as follows.

$$P = \log \frac{f^+ / N^+}{f^- / N^-}, if\ f^- \neq 0 \quad f^+ \neq 0 \qquad (2)$$

There are three special cases of P's value. If f- = 0 and f+ ≠ 0, we set P = 10; if f- = 0 and f+ ≠ 0, we set P = -10; and if f- = 0 and f+ = 0, we set P = 0.

Although the patterns do not distribute uniformly among Immunoglobulin C1 proteins, they cover all the positive proteins. The significance score of these patterns shows

Table 3: Number of patterns by APGM(τ = 0.35) and MGM on Immunoglobulin C1

	Support Threshold(σ)				
	6	5.5	5	4.5	4
APGM(τ = 0.35)	17	24	141	202	841
MGM	16	16	126	126	660

Table 4: Number of patterns by APGM(τ = 0.75) and MGM on Immunoglobulin V

	Support Threshold(σ)				
	6	5.5	5	4.5	4
APGM(τ = 0.75)	0	0	0	160	14686
MGM	0	0	0	0	13911

Table 5: Classification accuracy of APGM (τ = 0.35) and MGM on Immunoglobulin C1 Set

	Support Threshold(σ)				
	6	5.5	5	4.5	4
APGM	68.18%	77.27%	86.36%	90.91%	81.82%
MGM	72.73%	72.73%	72.73%	72.73%	72.73%

strong bias toward the Immunoglobulin C1 proteins, and among 202 only 30 noise features(P = -10) exist. For Immunoglobulin V proteins, the features miss two positive proteins, but these features are highly correlated with positive samples with all P equalling 10.

Computational performance

Since the support value of approximate subgraph mining and that of frequent subgraph mining have different meaning, it is generally hard to compare the computational performance of approximate subgraph mining and that of frequent subgraph mining. If τ is less than 1, approximate subgraph mining may obtain more patterns than that of general frequent subgraph mining by taking more running time. Because of this reason, we use the *pattern discovery rate* ("rate" for simplicity), which is computed as the number of discovered patterns N divided by the running time t. We use rate rather than running time as the criteria to compare computational efficiencies of different algorithms. We evaluated the computational efficiency of APGM with synthetic data sets.

We generated the synthetic data set by the same synthetic graph generator as [56]. The synthetic graph generator takes the following set of parameters: D is the total number of graphs; T is the average size of graph; I is the average size of potentially frequent subgraphs; L is the number of potentially frequent subgraphs; V is the number of vertex labels; E is the number of edge labels.

Table 6: Classification accuracy of APGM τ = 0.75) and MGM on Immunoglobulin V set

	Support Threshold (σ)			
	6	5.5	5	4.5
APGM	-	-	-	77.78%
MGM	-	-	-	-

TP, true positive; FP, false positive; TN, true negative; FN, false negative.
Accuracy = (TN+TP)/(TN+TP+FN+FP).
- means accuracies are unavailable.

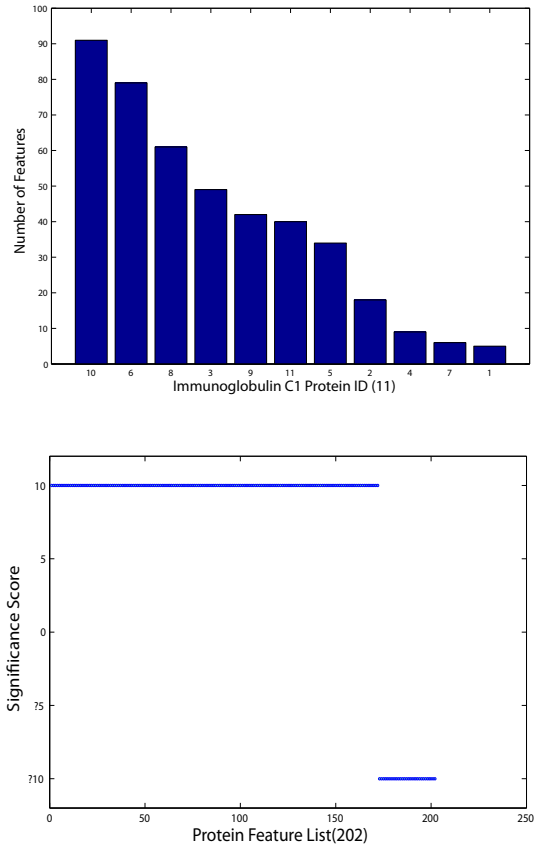


Figure 7
Distribution and significance of features among Immunoglobulin C1 Proteins. Upper: Distribution of frequent subgraph features among Immunoglobulin C1 proteins. **Lower:** Significance of frequent subgraph features among Immunoglobulin C1 proteins. Both figures are constructed for the set for classification. There are 202 patterns that are mined with the support threshold σ = 4.5 and the isomorphism threshold τ = 0.35.

The default parameter values that we use are D = 10000, T = 30, I = 11, L = 200, E = 20, V = 20.

We compare the performance rate between MGM and APGM using different isomorphism threshold values (and hence introduce different level of approximate matching). We use the support threshold (σ) defined in *Definition*6 in this experiment. From Figure 9, we see that with the change of isomorphism threshold, performance of APGM differs narrowly. Even if APGM takes approximate matching, its performance is very similar with MGM. Indeed, with some values of support threshold, APGM with low isomorphism threshold (τ = 0.6) even has much higher rates.

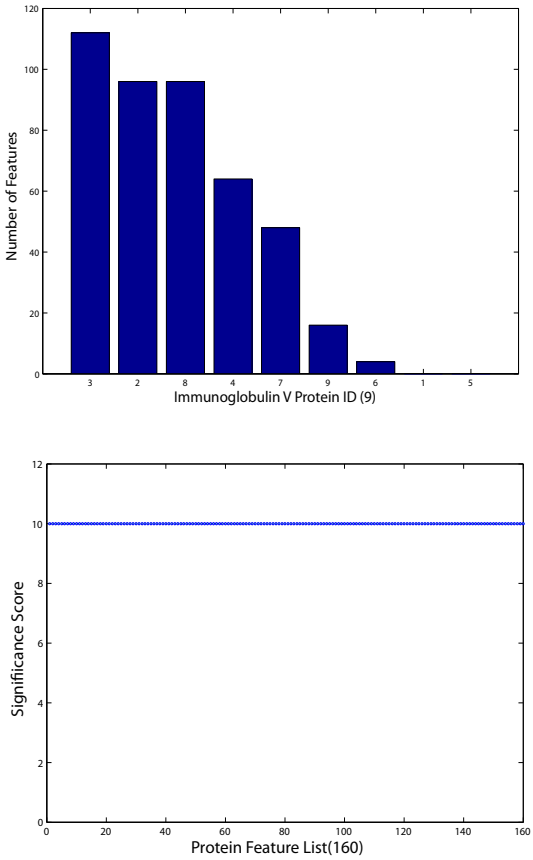


Figure 8
Distribution and significance of features among Immunoglobulin V proteins. Upper: Distribution of frequent subgraph features among Immunoglobulin V proteins. **Lower:** Significance of frequent subgraph features among Immunoglobulin V proteins. Both figures are constructed for the set for classification. There are 160 patterns that are mined with the support threshold σ = 4.5 and the isomorphism threshold τ = 0.75.

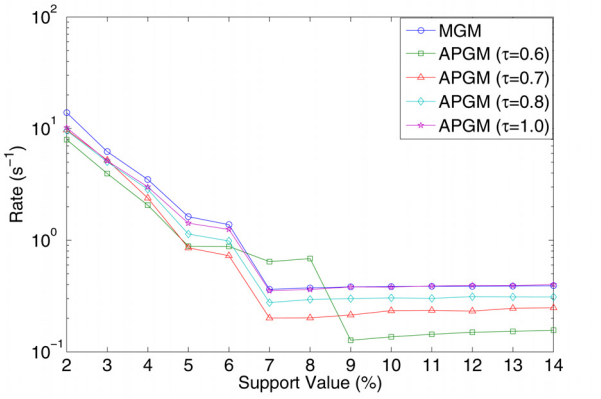


Figure 9
Computational performance comparison. We compared the computational performance between APGM and MGM using synthetic data sets. APGM used isomorphism threshold τ = 1.0, 0.8, 0.7, 0.6. Given the patterns' number *N* and running time *t* (*s*), *rate* = *N*/*t*.

Discussion

Finding features (corresponding to packing motifs) that discriminate one protein family from random selected proteins motivated us to further investigate the possibility of examining these motifs as characteristic signatures of a protein family. We investigated the spatial distribution of the residues covered by our mined structure motifs in individual proteins. We found the residues of structure motifs are highly centralized on a limited number of positions for each protein. We picked up the protein 1mju (chain l) in Immunoglobulin C1 set as one example. 202 patterns, which we obtained, maps to 21 amino acids among the total of 219 residues in 1mju. Through literature search, we found residues identified by APGM are related to the known functional sites in the protein. For example, position 200 and 202 are residues in contact with ligand GOL1406 as studied in [55]. Both positions are not discovered by the exact pattern mining method. This result suggests that APGM is more sensitive in recognizing functional related residues, as compared to exact pattern mining methods. However, we admit that comprehensive experimental study, involving multiple protein families, is needed before we could draw the conclusion convincingly.

Conclusion

In this paper we present a novel data mining algorithm, **APGM**(**AP**proximate **G**raph **M**ining), to perform structure comparison and structure motif identification in diverse proteins. In our method we encode structural motifs as subgraphs of geometric graph of proteins. Instead of using a general graph mining method to extract frequent subgraph motifs, we have developed the approximate graph mining algorithm and taken advantage of known substitution matrices in protein structure motif identification. Compared with general graph mining algorithms, APGM not only offers more qualified patterns that achieve higher classification accuracy, but also shows a reasonable computational performance. By applying this method to other protein families, "structure fingerprints" can be collected and used in domain classification schemes where structural information is desired. Furthermore, without loss of generality, choice of appropriate compatibility matrices allows our method to be employed in any domain where subgraph labels have some uncertainty. For example, networks of personal contacts "mutate" as people die or change employment. Compatibility matrices assigning

probabilities of 'label substitution' within families or organizations may allow the essential natures of personal contact subgraphs to be preserved nevertheless.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

YJ developed methods, implemented the software, and drafted the manuscript. VB and JZ were involved in testing the data set. JH was responsible for all aspects of the project, and helped revise the manuscript. LC provided advices on the biological aspect of the work, and helped revise the manuscript.

Acknowledgements

This work has been partially supported by the Kansas IDeA Network for Biomedical Research Excellence (NIH/NCRR award #P20 RR016475) and a NIH grant #R01 GM868665.

This article has been published as part of *BMC Bioinformatics* Volume 10 Supplement 1, 2009: Proceedings of The Seventh Asia Pacific Bioinformatics Conference (APBC) 2009. The full contents of the supplement are available online at http://www.biomedcentral.com/1471-2105/10?issue=S1

References

1. J L, HL P: **Antigen presentation and the ubiquitin-proteasome system in host-pathogen interactions.** *Adv Immunol* 2006, **92:**225-305.
2. Judson KA, Lubinski JM, Jiang M, Chang Y, Eisenberg RJ, Cohen GH, Friedman HM: **Blocking Immune Evasion as a Novel Approach for Prevention and Treatment of Herpes Simplex Virus Infection.** *J Virol* 2003, **77:**12639-12645.
3. RF D: *Of URFs and ORFs: A Primer on How to Analyze Derived Amino Acid Sequences Volume 92*. *Mill Valley: University Science Books*; 1986.
4. B R: **Twilight zone of protein sequence alignments.** *Protein Eng* 1999, **12(2):**85-94.
5. JU B, R L, D E: **A method to identify protein sequences that fold into a known three-dimensional structure.** *Science* **253(5016):**164-170. 1991 Jul 12
6. Hargbo J, Elofsson A: **Hidden Markov models that use predicted secondary structures for fold recognition.** *Proteins* 1999, **36(1):**68-76.
7. Campbell JA, Trossman DS , WM WM, Carayannopoulos LN: **Zoonotic orthopoxviruses encode a high-affinity antagonist of NKG2D.** *J Exp Med* **204(6):**1311-7.
8. Kryshtafovych A, Venclovas C, Fidelis K, Moult J: **Progress over the first decade of CASP experiments.** *Proteins* 2005, **61(Suppl 7):**225-236.
9. Gibrat J, Madej T, Bryant S: **Surprising similarities in structure comparison.** *Curr Opin Struct Biol* 1996, **6(3):**377-385.
10. Holm L, Sander C: **Mapping the protein universe.** *Science* 1996, **273:**595-602.
11. Bradley P, Kim PS, Berger B: **TRILOGY: Discovery of sequence-structure patterns across diverse proteins.** *Proc Natl Acad Sci USA* 2002, **99(13):**8500-8505.
12. Russell RB: **Detection of protein three-dimensional side-chain patterns: new examples of convergent evolution.** *Journal of Molecular Biology* 1998, **279:**1211-1227.
13. Stark A, Russell R: **Annotation in three dimensions. PINTS: Patterns in Non-homologous Tertiary Structures.** *Nucleic Acids Res* 2003, **31(13):**3341-4.
14. Barker J, Thornton J: **An algorithm for constraint-based structural template matching: application to 3D templates with statistical analysis.** *Bioinformatics* 2003, **19(13):**1644-9.
15. Nussinov R, Wolfson HJ: **efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques.** *PNAS* 1991, **88:**10495-99.
16. Wallace A, Borkakoti N, Thornton J: **TESS: a geometric hashing algorithm for deriving 3D coordinate templates for searching structural databases. Application to enzyme active sites.** *Protein Sci* 1997, **6(11):**2308-23.
17. Jonassen I, Eidhammer I, Conklin D, Taylor WR: **Structure motif discovery and mining the PDB.** *Bioinformatics* 2002, **18:**362-367.
18. Jonassen I, Eidhammer I, Taylor WR: **Discovery of local packing motifs in protein structures.** *Proteins* 1999, **34:**206-219.
19. Taylor WR, Jonassen I: **A Method for Evaluating Structural Models using Structural Patterns.** *Proteins* 2004.
20. Cammer S, Carter C, Tropsha A: **Identification of sequence-specific tertiary packing motifs in protein structures using Delaunay tessellation.** *Lecture notes in Computational Science and Engineering* 2002, **24:**477-494.
21. Krishnamoorthy B, Tropsha A: **Development of a four-body statistical pseudo-potential to discriminate native from non-native protein conformations.** *Bioinformatics* 2003, **19(12):**1540-48.
22. Tropsha A, Carter C, Cammer S, Vaisman I: **Simplicial neighborhood analysis of protein packing (SNAPP) : a computational geometry approach to studying proteins.** *Methods Enzymol* 2003, **374:**509-544.
23. Artymiuk PJ, Poirrette AR, Grindley HM, Rice DW, Willett P: **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.
24. Grindley H, Artymiuk P, Rice D, Willet P: **Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm.** *J Mol biol* 1993, **229:**707-721.
25. Huan J, Wang W, Bandyopadhyay D, Snoeyink J, Prins J, Tropsha A: **Mining Protein Family Specific Residue Packing Patterns >From Protein Structure Graphs.** *Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology (RECOMB)* 2004:308-315.
26. Milik M, Szalma S, Olszewski K: **Common Structural Cliques: a tool for protein structure and function analysis.** *Protein Eng* 2003, **16(8):**543-52.
27. Spriggs RV, Artymiuk PJ, Willett P: **Searching for patterns of amino acids in 3D protein structures.** *J Chem Inf Comput Sci* 2003, **43:**412-421.
28. Stark A, Shkumatov A, Russell RB: **Finding functional sites in structural genomics proteins.** *Structure (Camb)* 2004, **12:**1405-1412.
29. Wangikar P, Tendulkar A, Ramya S, Mali D, Sarawagi S: **Functional sites in protein families uncovered via an objective and automated graph theoretic approach.** *J Mol Biol* 2003, **326(3):**955-978.
30. Inokuchi A, Washio T, Motoda H: **An apriori-based algorithm for mining frequent substructures from graph data.** *PKDD'00* 2000:13-23.
31. Kuramochi M, Karypis G: **Frequent Subgraph Discovery.** *Proc International Conference on Data Mining'01* 2001:313-320.
32. Yan X, Han J: **gSpan: Graph-Based Substructure Pattern Mining.** *Proc International Conference on Data Mining'02* 2002:721-724.
33. Huan J, Wang W, Prins J: **Efficient mining of frequent subgraphs in the presence of isomorphism.** *Proc of ICDM* 2003.
34. Huan J, Prins WWJ, Yang J: **SPIN: Mining Maximal Frequent Subgraphs from Graph Databases.** *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 2004:581-586.
35. Nijssen S, Kok J: **A quickstart in frequent structure mining can make a difference.** *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 2004:647-652.
36. Han J, Cheng H, Xin D, Yan X: **Frequent Pattern Mining: Current Status and Future Directions.** *Data Mining and Knowledge Discovery* 2007, **14:**.
37. Koren Y, North SC, Volinsky C: **Measuring and extracting proximity in networks.** *KDD* 2006:245-255.
38. Tong H, Koren Y, Faloutsos C: **Fast Direction-Aware Proximity for Graph Mining.** .
39. Yan X, Zhu F, Yu PS, Ha J: **Feature based substructure similarity search.** *ACM Transactions on Database Systems* 2006.
40. Hasan M, Chaoji V, Salem S, jeremy Besson, Zaki M: **ORIGAMI: Mining Representative Orthogonal Graph Patterns.** *Proc. 2007 Int. Conf. on Data Mining (ICDM'07)* 2007.

41. Chen C, Yan X, Zhu F, Han J: **gapprox: Mining frequent approximate patterns from a massive network.** *Proc. 2007 Int. Conf. on Data Mining (ICDM'07)* 2007.
42. Holder LB, Cook DJ, Djoko S: **Substructures discovery in the subdue system.** *Proc AAAI'94 Workshop Knowledge Discovery in Databases* 1994:169-180.
43. Zhang S, Yang J, Cheedella V: **Monkey: Approximate Graph Mining Based on Spanning Trees.** *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference* 2007:1247-1249.
44. Vanetik N, Gudes E: **Mining Frequent Labeled and Partially Labeled Graph Patterns.** .
45. Bandyopadhyay D, Snoeyink J: **Almost-Delaunay Simplices : Nearest Neighbor Relations for Imprecise Points.** *ACM-SIAM Symposium On Distributed Algorithms* 2004:403-412.
46. Delaunay B: **Sur la sphere vide. A la memoire de Georges Voronoi.** *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk* 1934, **7:**793C800.
47. Huan J, Wang W, Bandyopadhyay D, Snoeyink J, Prins J, Tropsha A: **Mining Family Specific Residue Packing Patterns from Protein Structure Graphs.** *Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology (RECOMB)* 2004:308-315.
48. Huan J, Wang W, Prins J: **Efficient Mining of Frequent Subgraph in the Presence of Isomorphism.** *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)* 2003:549-552.
49. Eddy SR: **Where did the BLOSUM62 alignment score matrix come from.** *Nature Biotechnology* 2004, **22:**1035-1036.
50. Kelly G, *et al.*: **Structure of the cell-adhesion fragment of intimin from enteropathogenic Escherichia coli.** *Nature Struct Biol* 1999, **6:**313-318.
51. Hamburger Z, *et al.*: **Crystal structure of invasin: a bacterial integrin-binding protein.** *Science* 1999, **286:**291-295.
52. Wang G, Dunbrack RL, PISCES J: **A Protein Sequence Culling Server.** *Bioinformatics* 2003, **19:**1589-1591.
53. Huan J, Bandyopadhyay D, Snoeyink J, Prins J, Tropsha A, Wang W: **Distance-based identification of spatial motifs in proteins using constrained frequent subgraph mining.** *Proceedings of the IEEE Computational Systems Bioinformatics (CSB)* 2006.
54. **LIBSVM** [http://www.csie.ntu.edu.tw/~cjlin/libsvm]
55. **PDBsum Structure Database** [http://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/pdbsum]
56. Kuramochi M, Karypis G: **Frequent subgraph discovery.** *Proc International Conference on Data Mining01* 2001:313C320.