IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 20, 2018, accepted September 9, 2018, date of publication October 1, 2018, date of current version October 25, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872496

Classifying Protein Specific Residue Structures Based on Graph Mining

SUN-YUAN HSIEH[1], (Senior Member, IEEE), CHIA-WEI LEE[2], ZONG-YING YANG[3], HENG-WEI WANG[4], JUN-HAN YU[4], BO-CHENG CHAN[4], AND TAI-LING YE[4]

[1]Department of Computer Science and Information Engineering, Institute of Medical Information, National Cheng Kung University, Tainan 701, Taiwan
[2]Department of Computer Science and Information Engineering, National Taitung University, Taitung 950, Taiwan
[3]Institute of Medical Information, National Cheng Kung University, Tainan 701, Taiwan
[4]Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan

Corresponding author: Sun-Yuan Hsieh (hsiehsy@mail.ncku.edu.tw)

The work of C.-W. Lee was supported by the Ministry of Science and Technology of Taiwan under Grant MOST 105-2811-E-006-022.

ABSTRACT We proposed a 3-D protein structure using a simple and connected graph, where nodes indicate amino acids and edges represent contact distances between amino acids. Based on these graph structures, we present a graph mining algorithm to determine the crucial subgraphs in these graphs, which can be applied to classify protein structural families. The proposed algorithm was compared with BLAST, BLAT, and DALI. Moreover, an experiment was conducted, in which characteristic sub-structural patterns were found in several protein families within the Protein Data Bank.

INDEX TERMS B-factor, bioinformatic algorithms, protein classifications, protein specific residue subgraph mining.

I. INTRODUCTION

In biology, graphs are commonly used to describe chemical compounds [14], [22] and protein sequences. Moreover, they can be applied for identifying residue packing patterns from protein structures [15], [40], protein networks [35], and functional sites in proteins [1], [4], [5], as well as multiple sequence alignment algorithms of proteins with analogous structures [10]. Graphs can also be used to provide information about possible collaborative substructures in which conserved sequence patterns in a group of homologous proteins may have similar 3D arrangements.

In general, an algorithmic approach for mining frequent subgraphs entails finding all connected subgraphs that appear frequently in a large graph database. Several algorithms are available for this problem, including the SUBDUE algorithm [11], fast frequent subgraph mining algorithm [14]–[16], [40], graph-based substructure pattern mining algorithm [41], frequent subgraph discovery algorithm [20], and graph/sequence/tree extraction algorithm [27], [28]. However, many scientific and commercial applications may require locating frequent subgraphs in complex graph data sets with numerous vertices and edges, which is highly time consuming.

Although sequence alignment is associated with efficient computation time in protein classification, its accuracy can be further improved. In the current study, data mining and graph theory approaches were used for protein family classification on the basis of specific motif sequences from [21], [23], [24], [31], and [32]. A recurring structural motif is an important characteristic of protein structural series that are composed of amino acids (residues) in proteins. A mining enumeration algorithm was applied to the test protein to find the mark pattern subgraph, which is useful for classification. Therefore, recurring structural motifs within protein structures are crucial for several applications such as the prediction of protein functions, analysis of active sites, and classification of protein families. Recent studies have discovered millions of such motifs [26], [44]. In Fig. 1, rectangular boxes indicate Bcl-2 homolgy "BH" regions present in Bcl-xL. However, the process of locating structural motifs remains impeded by computational difficulties induced by the increasing complexity of protein structures. Hence, the current study applied the protein graph mining approach to identify the recurring structural motifs. This process can considerably reduce the time requirements.

When graphs are used to represent protein three-dimensional structures [34], the recurring protein structure motifs can be efficiently found using a frequent subgraph mining algorithm. Graph nodes represent the 20 types of amino acids, and edges are generated according to a threshold

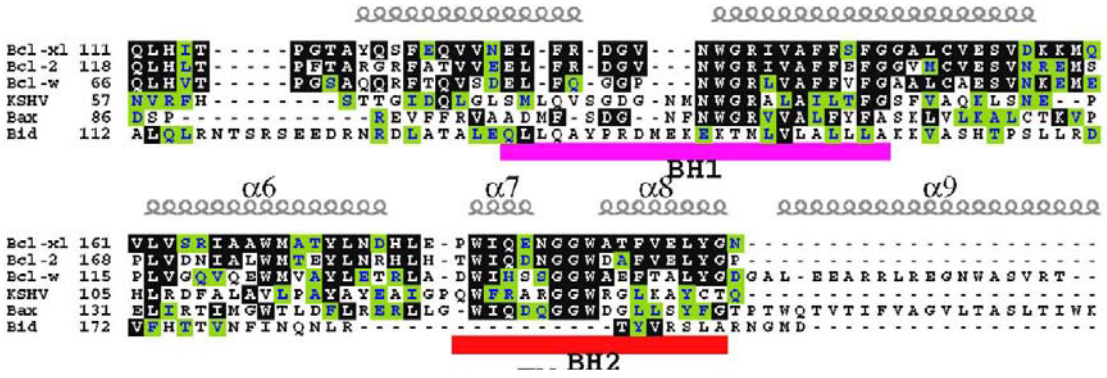


FIGURE 1. In the aligned sequences, black boxes represent strictly conserved residues, and gray boxes highlight conservative substitutions [31].

regarding to the contact distance between each pair of amino acids. For protein graph conformation, we used a particular value, called the ''*B*-factor,'' to considerably reduce the search space and to ensure that we used stable atomic coordinates of the amino acids.

The remainder of this paper is organized as follows: Section II surveys some related studies. Section III gives definitions and notations. Section IV proposes algorithms and data structures for subgraph mining. Section V shows the performance of the proposed algorithm using data sets. Finally, we conclude this paper with some remarks in Section VI.

II. RELATED WORKS

There are two types of subgraph mining algorithms. Algorithms of the first type utilize a level-wise search including Apriori to enumerate the recurring subgraphs [17]. Moreover, AGM [17] and FSG [20] are typical algorithms for mining graphs. Algorithms of the second type utilize a depth-first search (DFS) for identifying candidate frequent subgraphs [19]. Some examples of representative algorithms are graph-based substructure pattern (gSpan), GASTON, and FFSM. Krishna *et al.* [19] conducted a comparative survey of algorithms for frequent subgraph discovery, focusing on the classification of frequent subgraph discovery algorithms according to three factors: ''search strategy,'' ''nature of input,'' and ''completeness of output.'' The performance of each algorithm on three benchmark data sets was evaluated by comparing execution time and the number of frequent graphs. SUBDUE was proposed by Holder *et al.* [11] and is the earliest algorithm to be based on a greedy approach. Nevertheless, it cannot produce a complete set of frequent subgraphs.

A. BFS STRATEGY

The AGM algorithm was proposed by Inokuchi *et al.* [17], and can efficiently find all the frequent induced subgraphs in a large database. The meaning of the induced subgraph is that if there is an induced subgraph H of graph G, the vertices $V(H) \subseteq V(G)$ contain all edges of G connecting vertices in $V(H)$. If two vertices appear both in graph H and graph G, it is obvious that the connected edge between two vertices will certainly appear in graph G and graph H. Graphs are represented by the ''adjacency matrix'' and searching is performed using a level-wise strategy. BFS is the first algorithm based on the principle of the Apriori algorithm.

The other algorithm, called FSG, was proposed by Kuramochi and Karypis [20]. Its purpose is to identify all frequent connected subgraphs in a database. The greatest difference between AGM and FSG lies in the extension of mining subgraphs. AGM extends subgraphs by adding one vertex at each level. By contrast, FSG extends subgraphs by adding one edge at each level.

B. DFS STRATEGY

The gSpan algorithm was proposed by Yan and Han [41] in 2002. The motivation was to solve the shortcomings of the Apriori algorithm such as those related to candidate generation and the graph isomorphism problem in calculating the frequency of a graph. These are the major computational costs of frequent subgraph mining. The gSpan algorithm is the first algorithm that uses DFS and applies a canonical labeling system to avoid redundant computation. Nijssen and Kok [27], [28] developed the GASTON algorithm, which extracts the frequent paths, non cyclic trees, and cyclic graphs. The GASTON algorithm utilizes an ''embedding list'' to store all graph information. According to the different types of graph, GASTON split up the search process into different stages in order to reduce the running times, especially for large databases. Huan *et al.* [14]–[16] and Williams *et al.* [40] developed the FFSM algorithm for finding all frequent connected subgraphs in a graph database.

III. PRELIMINARIES

A. GRAPH THEORY

For definitions and notations of graphs, please refer to [39].

B. PROTEIN STRUCTURE CHARACTERISTICS

1) LEVELS OF PROTEIN STRUCTURE

There are four levels of protein structure, as shown in Fig. 2. In primary structures, proteins are constituted of and linked together by staggered amino acids. Approximately 20 types of amino acid occur naturally. Fig. 3 shows the relation

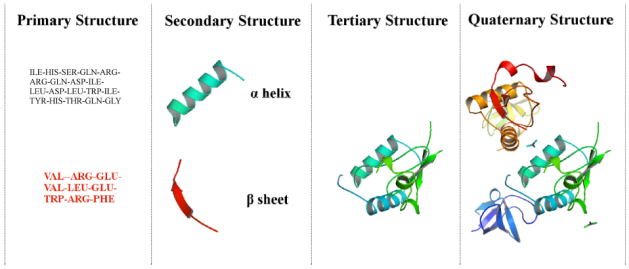


FIGURE 2. Four levels of protein structure.

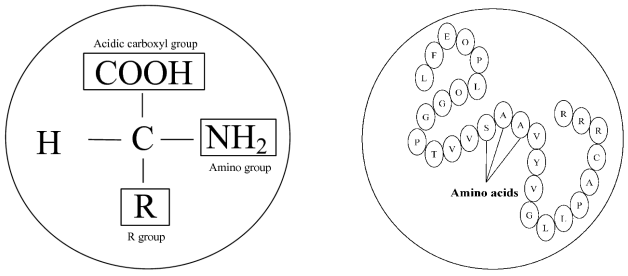


FIGURE 3. Primary structure of protein. (a) Amino acid structure with four groups: C atom, $-NH_2$ representing an amino group, $-COOH$ representing an acid group, and R representing a side chain. (b) Primary protein structure: a chain of amino acids.

between amino acids and protein structure. The simplest amino acid structure is composed of four groups as illustrated in Fig. 3(a): (a) Central carbon atom (C atom), (b) amino group ($-NH_2$), (c) acid group ($-COOH$), and (d) R group(side chain). The secondary structure comprises areas of folding or coiling within a protein such as alpha helices and beta sheets, which are stabilized by hydrogen bonding. The tertiary structure comprises the packing of random coils, alpha helices, and beta sheets, regarding to each other on the level of one complete polypeptide chain. The quaternary structure exists only when more than one polypeptide chain appear in a complex protein.

2) PDB

The PDB archive can be viewed as a repository of atomic coordinates and other information illustrating proteins and some other crucial biological macromolecules. Structural biologists utilize approaches including nanomagnetic resonance (NMR) spectroscopy and X-ray crystallography to identify the locations of atoms within the molecules. The content of a PDB file is shown in Table 1. In this file, vital information regarding amino acid residues, such as coordinates and B-factors, can be obtained. The B-factor describes the displacement of the atomic positions from an average value. In other words, if the atom has a higher B-factor value, it has a high mobility. B-factors are typically between 15 and 30, but are often high than 30 for more flexible regions.

IV. PROPOSED ALGORITHM

This section outlines the framework of the proposed method for identifying characteristic patterns of subgraphs in a test

TABLE 1. Atomic coordinates of each amino acid in a PDB file from the PDB [30].

Line	ATOM	Amino Acid	No.	Coordinates (x, y, z)	Occupancy	B-factor	Element
1	N	VAL	16	35.654 -32.603 34.832	1.00	3.69	N
2	CA	VAL	16	36.647 -33.353 35.656	1.00	3.57	C
3	C	VAL	16	37.779 -33.931 34.776	1.00	4.47	C
4	O	VAL	16	37.513 -34.709 33.870	1.00	2.16	O
5	CB	VAL	16	35.941 -34.530 36.411	1.00	4.92	C
6	CG1	VAL	16	36.957 -35.363 37.303	1.00	5.04	C
7	CG2	VAL	16	34.759 -33.915 37.273	1.00	6.10	C
8	N	VAL	17	39.005 -33.522 35.041	1.00	5.26	N
9	CA	VAL	17	40.162 -34.099 34.332	1.00	4.54	C
10	C	VAL	17	40.548 -35.375 35.076	1.00	7.42	C
11	O	VAL	17	40.531 -35.400 36.296	1.00	6.25	O
12	CB	VAL	17	41.347 -33.131 34.334	1.00	6.61	C
13	CG1	VAL	17	42.553 -33.736 33.624	1.00	5.28	C
14	CG2	VAL	17	40.932 -31.809 33.623	1.00	8.57	C
15	N	GLY	18	40.885 -36.447 34.359	1.00	6.75	N
16	CA	GLY	18	41.292 -37.652 35.052	1.00	4.54	C
17	C	GLY	18	40.172 -38.369 35.782	1.00	6.75	C
18	O	GLY	18	40.410 -39.040 36.791	1.00	8.07	O
19	N	GLY	19	38.947 -38.216 35.290	1.00	7.92	N
20	CA	GLY	19	37.834 -38.857 35.970	1.00	10.76	C
21	C	GLY	19	37.491 -40.211 35.393	1.00	11.20	C
22	O	GLY	19	38.201 -40.669 34.508	1.00	11.15	O
23	N	THR	20	36.485 -40.878 35.975	1.00	11.78	N
24	CA	THR	20	35.947 -42.129 35.451	1.00	11.35	C

protein graph database, including the enumeration of protein graphs and their representation by using proper data structures. We then present an algorithm for classifying the test proteins to determine whether they belong to a characteristic sub structural pattern family.

A. CREATING PROTEIN GRAPHS

In this study, we used graphs to represent protein structures. The vertices of each graph represent the amino acids so that there are 20 labels. Due to the fact that the protein backbone determines the whole protein conformation, we use the central carbon atom (C_α atom) to represent the amino acids. To reduce computational complexity, the coordinate frameworks are generated barely from the amino acids instead of from all the atoms.

Two amino acids are connected by a "bond edge" in the primary sequence, if they are consecutive. To construct the three-dimensional structure of the protein, the "proximity edge" is used to represent the graph edge.

Definition 1 (Proximity Edge): If the distance between the two associated C atoms is less than or equal to the threshold in a protein graph with n C atoms, where $D(i, j)$ for $1 \leq i, j \leq n$ is the distance between two atomic coordinates i and j, then there exists a proximity edge between i and j and $D(i, j) \equiv D(j, i)$. The threshold (set by experiment) is inserted only if the distance does not exceed 12Å [38].

There are several applications using displacement parameters (B-factor) [29], [43] (e.g., predicting protein flexibility prediction, protein thermal stability, and active sites and protein structures analysis [33], [43]). B-factors indicate how accurate a localized part of a structure is. The proposed algorithm sets a threshold μ between 0 and 10 on the central C atom in each amino acid. A C atom with a large μ is pruned and not considered (Fig. 4).

B. REPRESENTING A PROTEIN GRAPH BY AN ADJACENCY MATRIX

Each protein graph is assumed to be represented by an adjacency matrix.

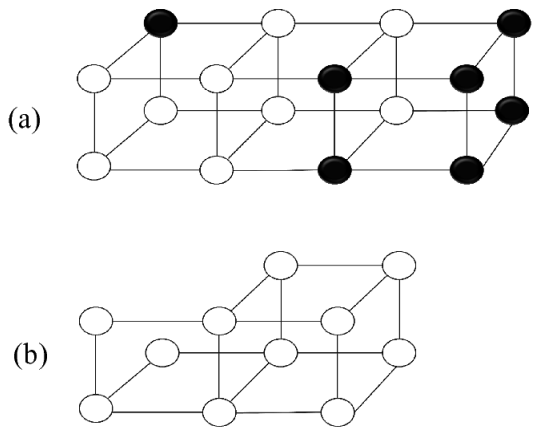
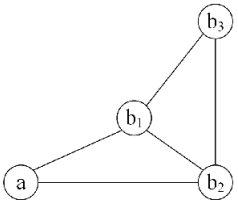


FIGURE 4. (a) Graph G with completed nodes and edges. The B-factor values of the black nodes are larger than the threshold μ. (b) Resulting graph after pruning the nodes.

Definition 2 (Adjacency Matrix): A adjacency matrix M of a graph G with n vertices is an $n \times n$ triangle matrix in which $m_{i,j}$ represents the entry of the ith row and jth column in M. In addition, the entry $m_{i,i}$, $1 \leq i \leq n$, indicates the label of a vertex in G, and each entry $m_{i,j}$, $0 < j \leq i \leq n$, is 1 if and only if there exists an edge between vertices $m_{i,i}$ and $m_{j,j}$, and is 0 if no edge exists between $m_{i,i}$ and $m_{j,j}$. Moreover, $code(M)$ is a string that represents M in a row-major manner.



a			
1	b_1		
1	1	b_2	
0	1	1	b_3

FIGURE 5. Example of a test protein graph G and its adjacency matrix M, where $code(M)$=a1$b_1$11$b_2$011b_3.

As shown in Fig. 5, the algorithm sorts all the amino acids by using (1) one-letter codes (the label of a node in G), and (2) the B-factor value of the node. For example, if there are four nodes a, b_1, b_2, b_3 and the B-factor value of the nodes is (10.0, 12.0, 12.5, 13.0) respectively, then the ordering of the four nodes is (a, b_1, b_2, b_3).

C. CANONICAL ADJACENCY MATRIX TREE

In the proposed method, a test protein graph G is constructed based on a data structure called the canonical adjacency matrix (CAM), and then the desired subgraphs are enumerated step by step. If some tag family pattern is matched by an enumerated subgraph, G is assigned to the marked patterns. The definition of a CAMs is as follows:

1) Each vertex is different from others.
2) A unique empty matrix represents the root.
3) Labeled vertices comprise Level 1 of the tree. Each labeled vertex is a child of the root.
4) Vertices in level 2 are generated from those vertices in level 1 by creating one adjacent vertex to each of them.
5) After generating the vertices of level 2, the CAM Tree-Extension Case algorithm and the CAM Tree-Join Case algorithm are applied to enumerate the desired subgraphs until the matrix of G can be formalized.

1) CASES FOR CAM-TREE-JOIN

The CAM-tree-join operation "superimposes" two graphs to generate a new candidate graph. The CAM-tree-join may produce one or two candidate graphs. Initially, two types of matrix must be defined, namely (1) the inner matrix, and (2) the outer matrix. An "inner" matrix is an adjacency matrix A for graph G if the last row of A contains at least two edge entries (two entries containing 1). Otherwise, if the last row of A contains only one edge entry, then A is an "outer" matrix. For two adjacency matrices $m \times m$ matrix A and $n \times n$ matrix B, let $a_{m,f}$ denote the last edge of A and $b_{n,k}$ denote the last edge of B. Join(A, B) can be obtained by executing the following two algorithms.

Algorithm 1 CAM_TREE_JOIN_CASE_1

Input: Two outer matrixes $m \times m$ matrix A and $n \times n$ matrix B, where the last edge of A is $a_{m,f}$ and the last edge of B is $b_{n,k}$.

Output: A CAM C.

if $(m = n) \wedge (f \neq k) \wedge (a_{m,m} = b_{n,n})$ **then**
  /* Case 1a */
  C is an $m \times m$ matrix, in which
  $c_{i,j} = a_{i,j} \vee b_{i,j}$
  Join(A, B) = {C}
if $(m = n) \wedge (a_{m,m} \neq b_{n,n})$ **then**
  /* Case 1b */
  D is an $(n+1) \times (n+1)$ matrix, in which

$$d_{i,j} = \begin{cases} a_{i,j}, & 0 < i,\ j \leq m, \\ b_{n,j}, & i = n+1,\ 0 < j < n, \\ 0, & i = n+1,\ j = n, \\ b_{n,n}, & i = n+1,\ j = n+1. \end{cases} \quad (1)$$

  Join(A, B) = {D}

According to Algorithm 1, two outer matrices $m \times m$ matrix A and $n \times n$ matrix B can be joined to obtain a matrix with the same size $n \times n$ (by Case 1a). Similarly, an $(m+1) \times (m+1)$ matrix can also be obtained (case 1b). Algorithm 1 compares all the diagonal entries of the two matrices n times and fills the elements of new matrix runs $\frac{n^2+n}{2}$ times. Joining an "inner" matrix and an "outer" matrix can generate the resulting matrix by executing Algorithm 2. It compares all the diagonal entries of the two matrices $n+1$ times and fills the elements of the new matrix runs $\frac{n^2+3n+1}{2}$ times; thus, the time complexity of the Join Operation is $\Theta(n^2)$. Figs. 6 and 7 illustrate Algorithms 1 and 2.

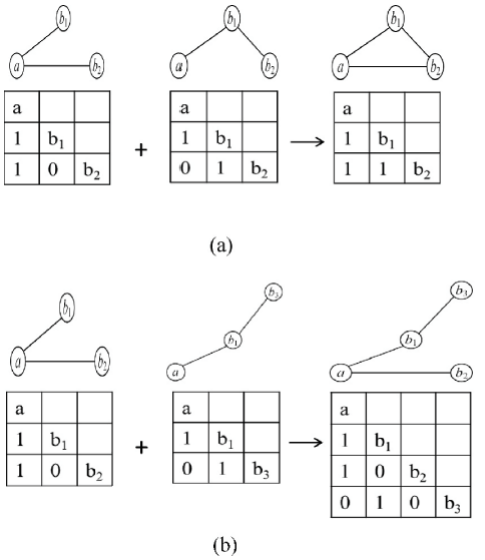


FIGURE 6. Example of Algorithm 1 CAM_TREE_JOIN_CASE_1 operation: (a) Case 1a, and (b) Case 1b.

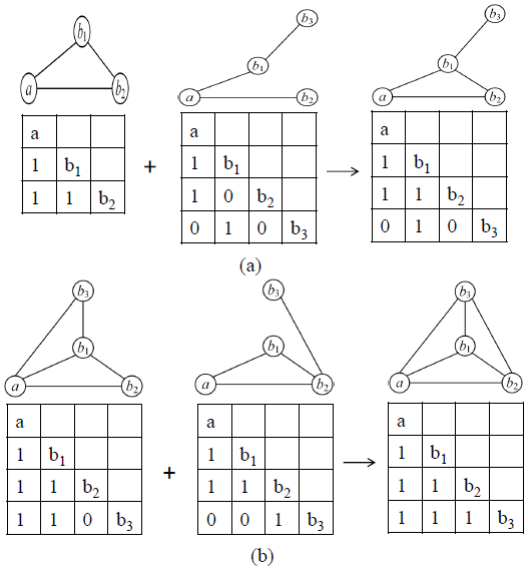


FIGURE 7. Example of Algorithm 2 CAM_TREE_JOIN_CASE_2 operation: (a) Case 2a, and (b) Case 2b.

2) CAM-TREE-EXTENSION

An additional technique for enumeration is the extension operation executed by Algorithm 3. From a graph G_k with k edges, it generates a candidate graph G_{k+1} with $(k+1)$ edges.

This algorithm extends the graph G_{origin} by adding the node u, which is adjacent to v, where $v \in G_{origin}$. Therefore, the algorithm fills the elements $\frac{n^2+3n+1}{2}$ times; thus, the extension operation has complexity $\Theta(n^2)$. Fig. 8 indicates how Algorithm 3 operates. If b_3 is adjacent to a, b_1, and b_2, all of them can use the EXTENSION operation to generate a new candidate graph.

Algorithm 2 CAM_TREE_JOIN_CASE_2

Input: Inner matrix $m \times m$ matrix A, and a outer matrix $n \times n$ matrix B, where the last edge of A is $a_{m,f}$ and the last edge of B is $b_{n,k}$.

Output: A CAM C.

if $(m < n)$ **then**

  /* Case 2a */

  C is an $n \times n$ matrix, where

$$c_{i,j} = \begin{cases} a_{i,j} \vee b_{i,j}, & 0 < i,\ j \le m, \\ b_{i,j}, & otherwise. \end{cases} \quad (2)$$

  $\text{Join}(A, B) = \{C\}$

if $(m = n) \wedge (f \neq k)$ **then**

  /* Case 2b */

  D is an $n \times n$ matrix, where

  $c_{i,j} = a_{i,j} \vee b_{i,j}$

  $\text{Join}(A, B) = \{D\}$

Algorithm 3 CAM_TREE_EXTENSION

Input: An outer matrix $A(n \times n)$ and a protein test graph $G = (V, E)$,

Output: A set of $(n+1) \times (n+1)$ adjacency matrices B.

$S \leftarrow \emptyset$

for *each edge* $(u, v) \in E$ *with* $(u \in (V \setminus A)) \wedge (v \in A)$ **do**

$$b_{i,j} = \begin{cases} a_{i,j}, & 0 < i,\ j \le m,\ i \neq n \text{ or } j \neq k, \\ b_{i,j}, & i = n+1,\ 0 < j < n, \\ 1, & i = n+1,\ j = n, \\ u, & i = n+1,\ j = n+1. \end{cases} \quad (3)$$

  $S \leftarrow S \bigcup \{B\}$

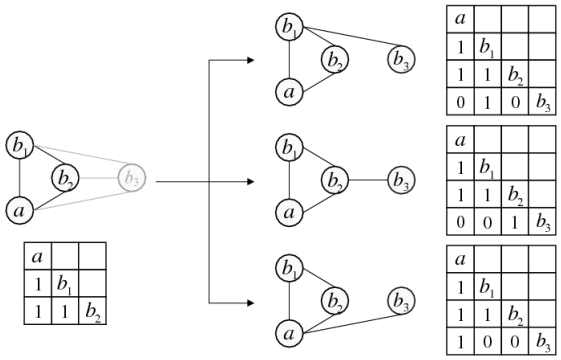
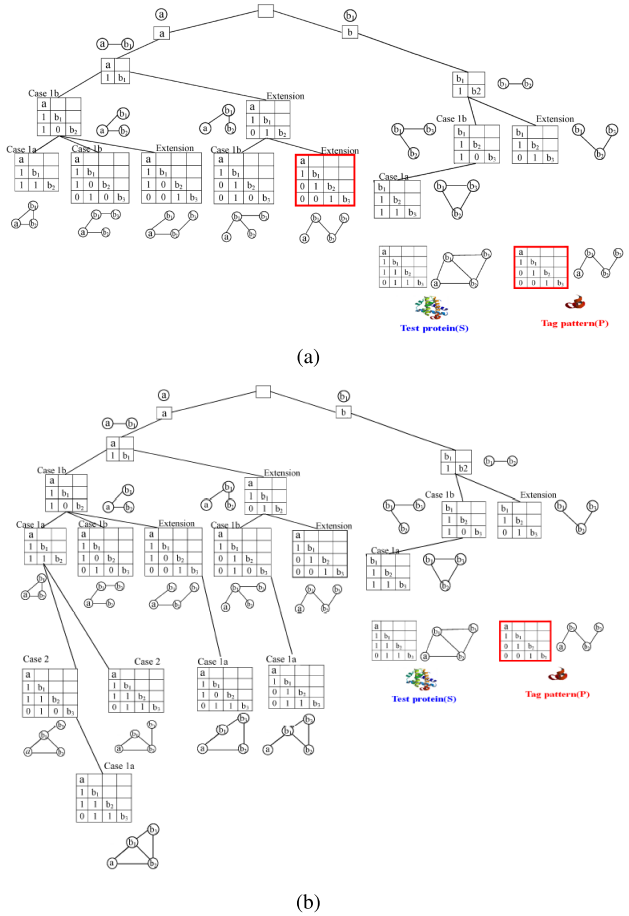


FIGURE 8. Example of Algorithm 3 CAM_TREE_EXTENSION operation.

D. CLASSIFICATION OF A PROTEIN FAMILY

Fig. 9 illustrates the classification procedure using Algorithm 4. If the candidate graph is isomorphic to the subgraph of the test protein graph, then the procedure



(a)

(b)

FIGURE 9. CAM tree for the test protein graph *G*. (a) Example of a CAM tree where the candidate subgraph can be identified in graph *G*. (b) Example of a CAM Tree where the candidate subgraph cannot be identified in graph *G*.

Algorithm 4 CLASSIFICATION_OF_PROTEIN

Input: G is the test protein graph; P is the conserved sequence represented by a graph.
Output: Whether a candidate subgraph belongs to subgraphs of the test protein graph.

```
X ← ∅ /* X is mining test protein subgraph so far */
while X is a subgraph of G do
    X ← CAM_TREE_JOIN_CASE
    X ← CAM_TREE_EXTENSION
    if X is isomorphic to P then
        X belongs to the characteristic substructural family
        break
    else
        X does not belong to the characteristic substructural family
```

is terminated; otherwise, it enumerates all of the subgraphs of the protein graph. In other words, the structure of candidate graph will affect the execution time for classification.

TABLE 2. Tested data sets.

Data Set	Protein Name	X-ray	NMR	Test
DS1	PKD-Like	4	1	5
DS2	Globin	7	1	8
DS3	BCL-XL	11	8	19
DS4	Immunoglobulin	7	6	13
DS5	Histone	8	0	8
DS6	HSP	9	2	11
DS7	serpin	11	0	11
DS8	serine proteases	9	3	12
DS9	E2F	5	4	9
DS10	Argonaute	10	4	14

TABLE 3. Different proximity edge lengths for Bcl-xl protein.

Proximity Length (Å)	B-factor	Average Time (sec.)	Accuracy(%)
5	60	0.0156	53
10	60	0.0623	94
15	60	0.3192	94
20	60	3.0232	100
25	60	7.9514	100

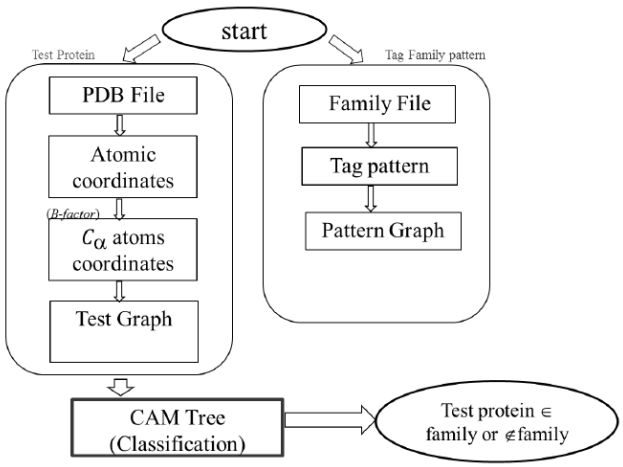


FIGURE 10. Experimental design flow chart.

V. EXPERIMENTAL RESULT

A. IMPLEMENTATION AND TEST PLATFORM

The proposed algorithm is compared with the following methods, BLAST [2],[1] BLAST-like alignment

[1]BLAST is used to find regions of local similarity between sequences.

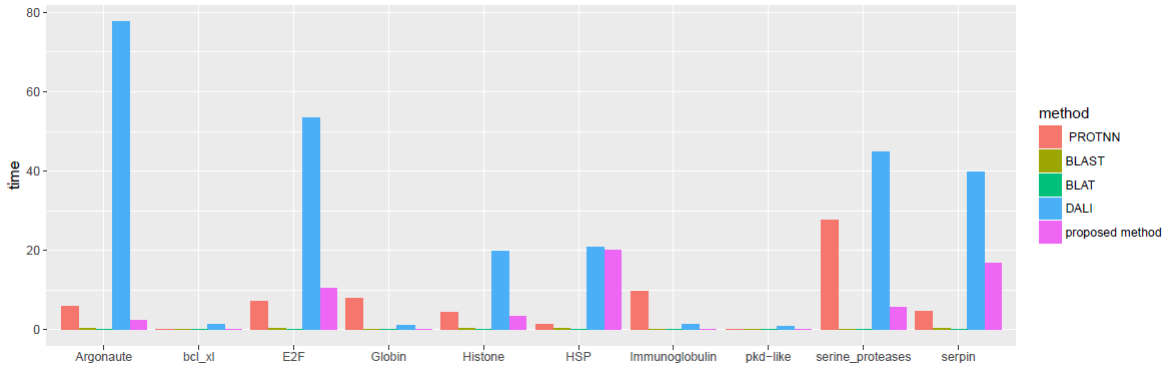


FIGURE 11. Average execution time of four algorithms for each protein family when *B*-factor = 50 and *threshold* = 10.

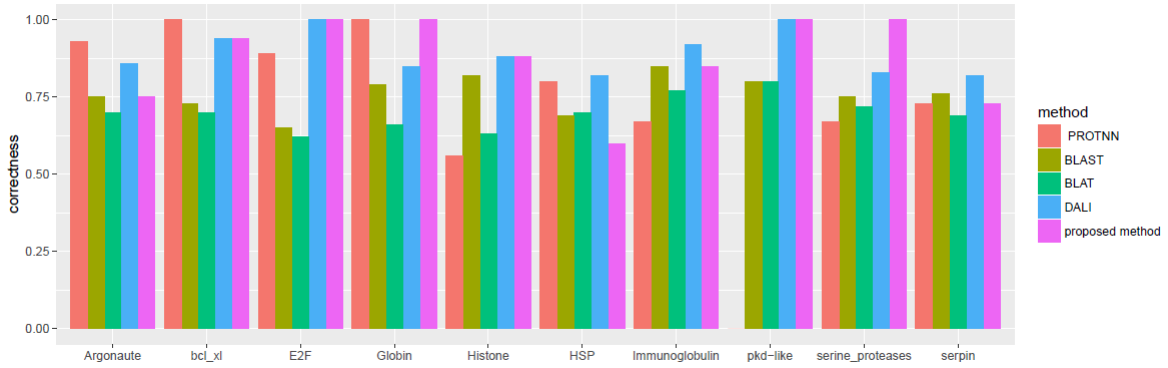


FIGURE 12. Average accuracy of four algorithms for each protein family when *B*-factor=50 and *threshold* = 10.

tool (BLAT) [18],[2] distance alignment (DALI) [12], and ProtNN [7].[3]

The data sets used for this experiment are transformed into a format that is accepted by these computer programs. In the experiment, we selected ten data sets from the PDB (Table 2), and we tested compressive BLAST, BLAT, DALI, and ProtNN on a real-world identical protein data set from the protein data bank. Experimental environment was a 2.67 GHz PC with 8 GB of memory. We used the C++ programming language to implement the proposed algorithm. The source codes of the BLAST, BLAT, DALI, and ProtNN were downloaded at http://www.ncbi.nlm.nih.gov/, http://genome.ucsc.edu/, http://ekhidna.biocenter.helsinki.fi/dali_lite/downloads/v3/index_html, and https://sites.google.com/site/wajdidhifli/softwares/protnn, respectively.

Table 3 shows the execution time and classification accuracy for various proximity lengths. For the distances 10 Å and 15 Å, each vertex and edge could be displayed on the graph. When the distance threshold grows, the execution time rapidly increases because the extra vertices and edges of each graph added. Moreover, due to the fact that the distinct B-factor also affects the executing time and accuracy of classification, we selected 40 and 50 [6] as the B-factors, as shown in Table 4.

TABLE 4. Different B-factors for Bcl-xl protein.

B-factor	Proximity length (Å)	Average Time (sec.)	Accuracy(%)
10	20	1.486	44
20	20	1.963	56
30	20	1.774	83
40	20	2.507	94
50	20	2.862	94
60	20	3.023	100

B. ACCURACY OF CLASSIFICATION OF EACH PROTEIN FAMILY

A flowchart of the experimental design is shown in Fig. 10. The proposed classification was compared with BLAST, BLAT, DALI, and ProtNN. The alignment results generated an *e*-value. On the basis of the *e*-value=$1 - e^{-10}$ (e=2.718), we evaluated the classification results. In the first experiment,

[2]This tool is a pairwise sequence alignment algorithm that was developed by Jim Kent at the University of California Santa Cruz (UCSC) in the early 2000s to help in the assembly and annotation of the human genome.

[3]ProtNN is a classification approach for protein 3D-structures that was developed by Wajdi Dhifli and Abdoulayé Diallo in 2016.

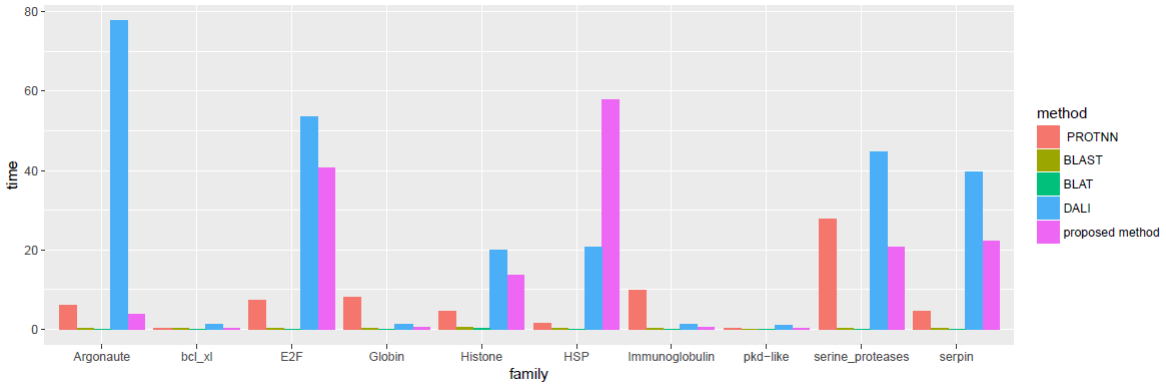


FIGURE 13. Accuracy and execution time of four algorithms for each protein family when *B*-factor = 60 and *threshold* = 15.

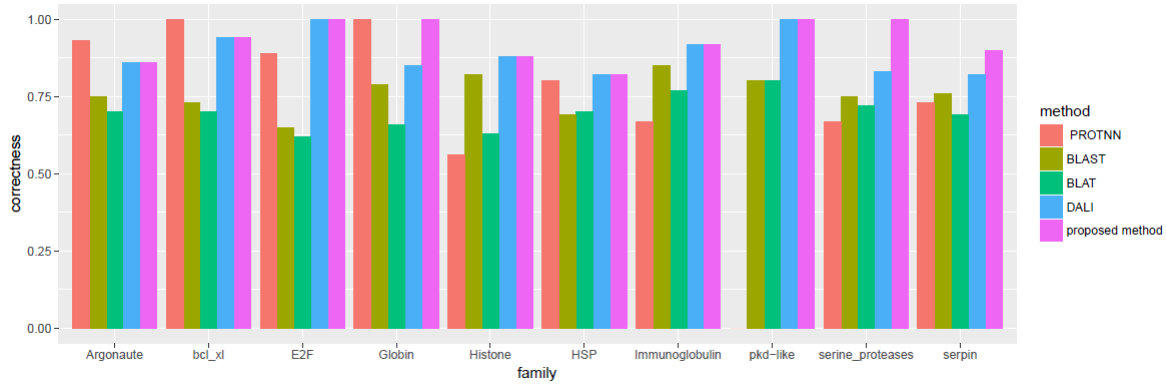


FIGURE 14. Accuracy and execution time of four algorithms for each protein family when *B*-factor = 60 and *threshold* = 15.

TABLE 5. Average execution time for classifying protein when *B*-factor = 50 and *threshold* = 10.

Data set	B-factor	Threshold	Proposed algorithm	BLAST	BLAT	DALI	ProtNN
DS1	50	10	0.135s	0.1095s	0.07s	0.96s	0.121s
DS2	50	10	0.169s	0.137s	0.087s	1.202s	8s
DS3	50	10	0.159s	0.178s	0.095s	1.331s	0.151s
DS4	50	10	0.1924s	0.156s	0.0991s	1.368s	9.789s
DS5	50	10	3.517s	0.375s	0.196s	19.862s	4.47s
DS6	50	10	19.9941s	0.249s	0.115s	20.713s	1.452s
DS7	50	10	16.703s	0.248s	0.039s	39.676s	4.652s
DS8	50	10	5.653s	0.156s	0.116s	44.726s	27.66s
DS9	50	10	10.422s	0.268s	0.077s	53.505s	7.308s
DS10	50	10	2.368s	0.259s	0.028s	77.738s	5.947s

TABLE 6. Average accuracy for classifying protein when *B*-factor = 50 and *threshold* = 10.

Data set	B-factor	Threshold	Proposed algorithm	BLAST	BLAT	DALI	ProtNN
DS1	50	10	100%	80%	80%	100%	0%
DS2	50	10	100%	79%	66%	85%	100%
DS3	50	10	94%	73%	70%	94%	100%
DS4	50	10	85%	85%	77%	92%	67%
DS5	50	10	88%	82%	63%	88%	56%
DS6	50	10	60%	69%	70%	82%	80%
DS7	50	10	73%	76%	69%	82%	73%
DS8	50	10	83%	75%	72%	83%	67%
DS9	50	10	100%	65%	62%	100%	89%
DS10	50	10	75%	75%	70%	86%	93%

we set the value of the *B*-factor and *threshold* at 50 and 10, respectively. Table 5 and Fig. 11 present the execution time on the mining graph. Table 6 and Fig. 12 present the accuracy of classification of each protein family. Because BLAST and BLAT use sequence searching to identify specific patterns, the proposed algorithm uses graph mining to solve this problem. Although the execution time is higher, the accuracy of the proposed algorithm is higher than those of BLAST and BLAT.

Compared with ProtNN, the accuracy of some protein families is higher than proposed algorithm. However, if ProtNN cannot classify the protein family correctly, ProtNN requires extensive execution time. In the proposed algorithm, the execution time does not increase even if the proposed algorithm classified the protein family error.

The results are illustrated in Fig. 11 and Fig. 12. However, the performance was dissatisfactory for some data sets. Consequently, we increased the value of the *B*-factor and *threshold* to 60 and 15, respectively, for an additional experiment, as shown in Tables 7 and 8, and in Figs. 13 and 14.

In the additional experiment, we created artificial data sets to test various numbers of nodes. Fig. 15 demonstrates that the proposed algorithm can be applied to protein graphs with large numbers of nodes.[4] The experimental results also imply that the proposed method can be applied to efficiently classify protein structures.

[4]In general, the number of nodes of the candidate subgraph is between 10 and 15.

TABLE 7. Average execution time for classifying protein when *B*-factor = 60 and *threshold* = 15.

Data set	B-factor	Threshold	Proposed algorithm	BLAST	BLAT	DALI	ProtNN
DS1	60	15	0.324s	0.11s	0.07ss	0.96s	0.121s
DS2	60	15	0.506s	0.137s	0.087s	1.202s	8s
DS3	60	15	0.319s	0.178s	0.095s	1.331s	0.151s
DS4	60	15	0.494s	0.156s	0.099s	1.368s	9.789s
DS5	60	15	13.69s	0.375s	0.196s	19.862s	4.47s
DS6	60	15	57.733s	0.249s	0.115s	20.713s	1.452s
DS7	60	15	22.208s	0.248s	0.039s	39.676s	4.652s
DS8	60	15	20.819s	0.156s	0.116s	44.726s	27.66s
DS9	60	15	40.538s	0.268s	0.077s	53.505s	7.308s
DS10	60	15	3.819s	0.259s	0.028s	77.738s	5.947s

TABLE 8. Average accuracy for classifying protein when *B*-factor = 60 and *threshold* = 15.

Data set	B-factor	Threshold	Proposed algorithm	BLAST	BLAT	DALI	ProtNN
DS1	60	15	100%	80%	80%	100%	0%
DS2	60	15	100%	79%	66%	85%	100%
DS3	60	15	94%	73%	70%	94%	100%
DS4	60	15	92%	85%	77%	92%	67%
DS5	60	15	88%	82%	63%	88%	56%
DS6	60	15	82%	69%	70%	82%	80%
DS7	60	15	90%	76%	69%	82%	73%
DS8	60	15	83%	75%	72%	83%	67%
DS9	60	15	100%	65%	62%	100%	89%
DS10	60	15	86%	75%	70%	86%	93%

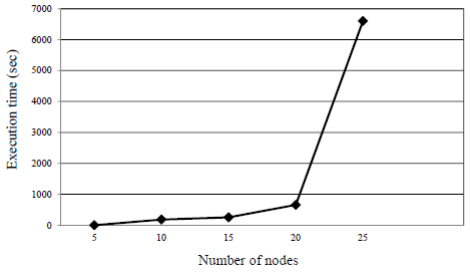


FIGURE 15. Number of nodes versus execution time.

VI. CONCLUSION

In this paper, a graph mining algorithm for classifying protein families was proposed. The proposed algorithm retains high accuracy and efficient execution time. Moreover, with the aids of using proximity edge length and adopting the threshold of B-factors, the proposed algorithm can substantially reduce the searching space. The experimental results demonstrate that our method can be efficiently applied to classify protein structures.

VII. ACKNOWLEDGMENT

This paper was presented at the Proceedings of the International Conference on Intelligent Computing, ICIC 2015: Intelligent Computing Theories and Methodologies, Lecture Notes in Computer Science, vol. 9225, pp. 723–729.

REFERENCES

[1] P. Aloy, E. Querol, F. X. Aviles, and M. J. E. Sternberg, "Automated structure-based prediction of functional sites in proteins: Applications to assessing the validity of inheriting protein function from homology in genome annotation and to protein docking," *J. Mol. Biol.*, vol. 311, no. 2, pp. 395–408, 2001.

[2] S. F. Altschul *et al.*, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *Nucleic Acids Res.*, vol. 25, no. 17, pp. 3389–3402, 1997.

[3] Z. Aung and K.-L. Tan, "Automatic protein structure classification through structural fingerprinting," in *Proc. 4th IEEE Symp. Bioinf. Bioeng. (ICBBE)*, Mar. 2004, pp. 508–515.

[4] D. Bandyopadhyay, J. Huan, J. Liu, W. Wang, J. Prins, and J. Snoeyink, "Using fast subgraph isomorphism checking for protein functional annotation using SCOP and gene ontology," Dept. Comput. Sci., Univ. North Carolina, Chapel Hill, NC, USA, Tech. Rep. TR04-031, 2004.

[5] D. Bandyopadhyay *et al.*, "Functional neighbors: Inferring relationships between nonhomologous protein families using family-specific packing motifs," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 5, pp. 1137–1143, Sep. 2010.

[6] *B-Factors and Coordinate Error Obtained From*. [Online]. Available: http://www.proteopedia.org/wiki/index.php/Temperature_value

[7] W. Dhifli and A. B. Diallo, "ProtNN: Fast and accurate protein 3D-structure classification in structural and topological space," *BioData Mining*, vol. 9, p. 30, Sep. 2016.

[8] D. Fischer, H. Wolfson, and R. Nussinov, "Three-dimensional, sequence order-independent structural comparison of a serine protease against the crystallographic database reveals active site similarities: Potential implications to evolution and to protein folding," *Protein Sci.*, vol. 3, no. 5, pp. 769–778, 1994.

[9] T. G. Gligoris *et al.*, "Closing the cohesin ring: Structure and function of its Smc3-kleisin interface," *Science*, vol. 346, no. 6212, pp. 963–967, Nov. 2014.

[10] S. Henikoff, J. G. Henikoff, and S. Pietrokovski, "Blocks+: A non-redundant database of protein alignment blocks derived from multiple compilations," *Bioinformatics*, vol. 15, no. 6, pp. 471–479, 1999.

[11] L. B. Holder, D. J. Cook, and S. Djoko, "Substructure discovery in the SUBDUE system," in *Proc. Assoc. Advancement Artif. Intell. Workshop Knowl. Discovery Database (AAAI)*, Jul. 1994, pp. 169–180.

[12] L. Holm and P. Rosenström, "DALI server: Conservation mapping in 3D," *Nucleic Acids Res.*, vol. 38, pp. 545–549, Jul. 2010.

[13] S.-Y. Hsieh, C.-W. Lee, Z.-Y. Yang, H.-W. Wang, and J.-H. Yu, "A novel algorithm for classifying protein structure familiar by using the graph mining approach," in *Proc. Int. Conf. Intell. Comput. (ICIC), Intell. Comput. Theories Methodol.*, in Lecture Notes in Computer Science, vol. 9225, 2015, pp. 723–729.

[14] J. Huan, D. Bandyopadhyay, W. Wang, J. Snoeyink, J. Prins, and A. Tropsha, "Comparing graph representations of protein structure for mining family-specific residue-based packing motifs," *J. Comput. Biol.*, vol. 12, no. 6, pp. 657–671, 2005.

[15] 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 *Proc. 8th Annu. Int. Conf. Res. Comput. Mol. Biol. (RECOMB)*, Mar. 2004, pp. 308–315.

[16] J. Huan, W. Wang, and J. Prins, "Efficient mining of frequent subgraphs in the presence of isomorphism," in *Proc. 3rd IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2003, pp. 549–552.

[17] A. Inokuchi, T. Washio, and H. Motoda, "An Apriori-based algorithm for mining frequent substructures from graph data," in *Proc. 4th Eur. Conf. Princ. Pract. Knowl. Discovery Data Mining (PKDD)*, 2000, pp. 13–23.

[18] W. J. Kent, "BLAT—The BLAST-like alignment tool," *Genome Res.*, vol. 12, no. 4, pp. 656–664, 2000.

[19] V. Krishna, N. N. R. R. Suri, and G. Athithan, "A comparative survey of algorithms for frequent subgraph discovery," *Current Sci.*, vol. 100, no. 25, pp. 190–198, 2011.

[20] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," in *Proc. 1th IEEE Conf. Data Mining (ICDM)*, Nov. 2001, pp. 313–320.

[21] M. Laberge and T. Yonetani, "Common dynamics of globin family proteins," *Int. Union Biochem. Molecular Biol.*, vol. 59, nos. 8–9, pp. 528–534, 2007.

[22] W. W. M. Lam and K. C. C. Chan, "A graph mining algorithm for classifying chemical compounds," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Nov. 2008, pp. 321–324.

[23] D. Landsman and A. P. Wolffe, "Common sequence and structural features in the heat-shock factor and Ets families of DNA-binding domains," *Trends Biochem. Sci.*, vol. 20, no. 6, pp. 225–226, 1995.

[24] J. A. Lees *et al.*, "The retinoblastoma protein binds to a family of E2F transcription factors," *Mol. Cellular Biol.*, vol. 13, no. 12, pp. 7813–7825, 1993.

[25] M. Lenoir *et al.*, "Structural basis of dynamic membrane recognition by trans-golgi network specific FAPP proteins," *J. Mol. Biol.*, vol. 427, no. 4, pp. 966–981, Feb. 2015.

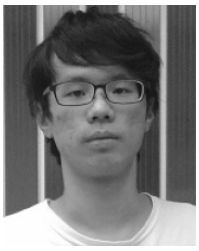
[26] D. S. Lieber, O. Elemento, and S. Tavazoie, "Large-scale discovery and characterization of protein regulatory motifs in eukaryotes," *PLoS ONE*, vol. 5, no. 12, p. e14444, 2010.

[27] S. Nijssen and J. N. Kok, "A quickstart in frequent structure mining can make a difference," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Aug. 2004, pp. 647–652.
[28] S. Nijssen and J. N. Kok, "Frequent graph mining and its application to molecular databases," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, vol. 5, Oct. 2004, pp. 4571–4577.
[29] S. Parthasarathy and M. R. N. Murthy, "Protein thermal stability: Insights from atomic displacement parameters (B values)," *Protein Eng.*, vol. 13, no. 1, pp. 9–13, 2000.
[30] *Protein Data Obtained From Protein Data Bank*. [Online]. Available: http://www.rcsb.org/pdb/home/home.do
[31] A. M. Petros, E. T. Olejniczak, and S. W. Fesik, "Structural biology of the Bcl-2 family of proteins," *Biochimica et Biophysica Acta, Mol. Cell Res.*, vol. 1644, nos. 2–3, pp. 83–94, 2004.
[32] E. Remold-O'Donnell, "The ovalbumin family of Serpin proteins," *Fed. Eur. Biochem. Soc. Lett.*, vol. 315, no. 2, pp. 105–108, 1993.
[33] D. E. Tronrud and E. Dale, "Knowledge-based B-factor restraints for the refinement of proteins," *J. Appl. Crystallogr.*, vol. 29, no. 2, pp. 100–104, 1996.
[34] S. Vishveshwara, K. V. Brinda, and N. Kannan, "Protein structure: Insights from graph theory," *J. Theor. Comput. Chem.*, vol. 1, no. 1, pp. 187–211, 2002.
[35] B. Wackersreuther, P. Wackersreuther, A. Oswald, C. Böhm, and K. M. Borgwardt, "Frequent subgraph discovery in dynamic networks," in *Proc. 8th Workshop Mining Learn. Graphs (MLG)*, Aug. 2010, pp. 155–162.
[36] B. Wang, D.-S. Huang, and C. Jiang, "A new strategy for protein interface identification using manifold learning method," *IEEE Trans. Nanobiosci.*, vol. 13, no. 2, pp. 118–123, Jun. 2014.
[37] S.-L. Wang, X. Li, S. Zhang, J. Gui, and D.-S. Huang, "Tumor classification by combining PNN classifier ensemble with neighborhood rough set based gene reduction," *Comput. Biol. Med.*, vol. 40, no. 2, pp. 179–189, 2010.
[38] N. Weskamp, D. Kuhn, E. Hüllermeier, and G. Klebe, "Efficient similarity search in protein structure databases by k-clique hashing," *Bioinformatics*, vol. 20, no. 10, pp. 1522–1526, 2005.
[39] D. B. West, *Introduction to Graph Theory*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.
[40] D. W. Williams, J. Huan, and W. Wang, "Graph database indexing using structured graph decomposition," in *Proc. 23rd Int. Conf. Data Eng. (ICDE)*, Apr. 2007, pp. 975–976.
[41] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proc. 3rd IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2002, pp. 721–724.
[42] H.-J. Yu and D.-S. Huang, "Novel 20-D descriptors of protein sequences and it's applications in similarity analysis," *Chem. Phys. Lett.*, vol. 531, pp. 261–266, Apr. 2012.
[43] Z. Yuan, T. L. Bailey, and R. D. Teasdale, "Prediction of protein B-factor profiles, large-scale discovery and characterization of protein regulatory motifs in eukaryotes," *Proteins*, vol. 58, no. 4, pp. 905–912, 2005.
[44] K. Zhang *et al.*, "Connor, "Common functional genetic variants in catecholamine storage vesicle protein promoter motifs interact to trigger systemic hypertension," *J. Amer. College Cardiol.*, vol. 55, no. 14, pp. 1463–1475, 2010.

SUN-YUAN HSIEH received the Ph.D. degree in computer science from National Taiwan University, Taipei, Taiwan, in 1998. He then served the compulsory two-year military service. From 2000 to 2002, he was an Assistant Professor at the Department of Computer Science and Information Engineering, National Chi Nan University. In 2002, he joined the Department of Computer Science and Information Engineering, National Cheng Kung University, where he is currently a Distinguished Professor. His current research interests include design and analysis of algorithms, fault-tolerant computing, bioinformatics, parallel and distributed computing, and algorithmic graph theory. He is a fellow of the British Computer Society. He received the 2007 K. T. Lee Research Award, the President's Citation Award (American Biographical Institute) in 2007, the Engineering Professor Award of the Chinese Institute of Engineers (Kaohsiung Branch) in 2008, the National Science Council's Outstanding Research Award in 2009, and the IEEE Outstanding Technical Achievement Award (IEEE Tainan Section) in 2011.

CHIA-WEI LEE received the B.S. and M.S. degrees from the Department of Computer Science and Information Engineering, National Chi Nan University, Taiwan, in 2003 and 2005, respectively, and the Ph.D. degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, in 2009. He then served the compulsory 11-month military service. From 2011 to 2017, he was an Assistant Research Fellow at the Department of Computer Science and Information Engineering, National Cheng Kung University. He is currently an Assistant Professor at the Department of Computer Science and Information Engineering, National Taitung University. His current research interests include graph theory and algorithms, interconnection networks, and system-level diagnosis.

ZONG-YING YANG received the M.S. degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, in 2014. His current research interests include design and analysis of algorithms and bioinformatics.



HENG-WEI WANG received the B.S. degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, in 2014. His current research interests include design and analysis of algorithms and bioinformatics.

JUN-HAN YU received the B.S. degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, in 2014. His current research interests include design and analysis of algorithms and bioinformatics.

BO-CHENG CHAN received the B.S. degree from the Department of Computer Science and Information Engineering, National Taiwan Ocean University, Taiwan, in 2014. He is currently pursuing the M.S. degree with the Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include wireless personal area network communication and bioinformatics.

TAI-LING YE received the B.S. degree from the Department of Computer Science and Information Engineering, National Chi Nan University, Taiwan, in 2008, and the M.S. and Ph.D. degrees from the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, in 2010 and 2017, respectively. His research interests include graph theory, interconnection networks, system-level fault diagnosis, and bioinformatics.