World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:8, 2009

Using Spectral Vectors and M-Tree for Graph Clustering and Searching in Graph Databases of Protein Structures

Do Phuc, and Nguyen Thi Kim Phung

***Abstract*—**In this paper, we represent protein structure by using graph. A protein structure database will become a graph database. Each graph is represented by a spectral vector. We use Jacobi rotation algorithm to calculate the eigenvalues of the normalized Laplacian representation of adjacency matrix of graph. To measure the similarity between two graphs, we calculate the Euclidean distance between two graph spectral vectors. To cluster the graphs, we use M-tree with the Euclidean distance to cluster spectral vectors. Besides, M-tree can be used for graph searching in graph database. Our proposal method was tested with graph database of 100 graphs representing 100 protein structures downloaded from Protein Data Bank (PDB) and we compare the result with the SCOP hierarchical structure.

***Keywords*—**Eigenvalues, m-tree, graph database, protein structure, spectra graph theory.

I. Introduction

IN protein structure databases, it is common to have similarity query, such as asking for proteins in the database which have similar structure to a given protein structure. In this paper, we use graph to represent protein structure. The similarity query of protein structures will become the graph searching. The challenge of this problem is the problem of graph comparison. Subgraph isomorphism is known to be NP complete. As a result, the graph comparison is a NP hard problem. Several research approaches for this problem have been proposed recently. In [7], the authors proposed a method for graph indexing called Closure-tree. Closure-tree is a hierarchical index. The graphs (database objects) are in the leafs and graph closures are in the internal nodes of closure tree. Graph closures can be considered as a cluster representation that aggregates structural and annotation information of the underlined graphs. In [4], the authors proposed a method for graph indexing based on listing all of short paths. This approach provided the additionally frequency information and built up the distinct short paths as key-table and used it for filtering when a query is processed.

Do Phuc, Associate Professor, is with Department of Information Systems, University of Information Technology. His research interests are data mining, bioinformatics (e-mail: phucdo@uit.edu.vn).

Nguyen Thi Kim Phung, PhD-student, is with Department of Information Systems, University of Information Technology. Her research interests are database theory, text classification, bioinformatics (e-mail: phungntk@uit.edu.vn).

In [19],[20] the authors used frequent and discriminate graphs as features to index objects. In [15], the authors used frequent trees to index entities. The above methods require a candidate subgraph isomorphism in verification step and not all frequent features are good for indexing the graphs.

In this paper, instead of using the subgraph isomorphism for graph comparison, we approach to spectral graph theory by using spectral vector to represent graph. The normalized Laplacian representation of the adjacency matrix of graph is used to calculate the eigenvalues by using Jacobi rotation algorithm. From a set of eigenvalues, we create spectral vector for graph. The Euclidean distance between two spectral vectors is used to measure the similarity between two graphs. The similarity searching efficiency of M-tree in multimedia databases has been proven in [5],[13],[14]. We used M-tree for clustering and similarity search in graph databases. The M-tree is used to build a hierarchical cluster of graphs representing protein structures. The graph database of protein structures downloaded from PDB and SCOP are used for testing our proposed method. The rest of this papers is as follows 2) Using graph to represent the protein structure 3) Similarity between two graphs 4) Using M-tree for clustering and similarity search in graph database 5) Experiment and discussion 6) Conclusion.

II. Using Graph to Represent Protein Structure

Protein is considered as polypeptide chains linked together. A polypeptide chain is a chain of amino acids (amino acid residues) linked together by peptide bonds. An amino acid consists of a central carbon atom (usually alpha Carbon C_α) and an amino group (NH2), a hydrogen atom (H), a Carboxy group (COOH) and a side chain (R) bound to the C_α. The backbone of the polypeptide is given by the repeated sequence of three atoms of each residue in the chain: the amide N, the alpha Carbon C_α and the Carbonyl C.

Fig. 1 is a schematic representation of spatial neighbors of a residue "i" in a polypeptide chain [17]. A distance cut off is taken and the residues which fall within the radius are the spatial neighbors of residue "i". Residues A,B,C and D are spatial neighbors of "i". The spatial neighbors are indicated using dotted lines shown in segment of the polypeptide chain.

Based on the data of protein structure in PDB (Protein Data Bank), we have data of the protein structures. The following is

the information of protein named 1ARB (PROTEASE).
HEADER HYDROLASE(SERINE PROTEASE) 15-APR-93 1ARB
TITLE THE PRIMARY STRUCTURE AND STRUCTURAL CHARACTERISTICS OF
TITLE 2 ACHROMOBACTER LYTICUS PROTEASE I, A LYSINE-SPECIFIC SERINE
TITLE 3 PROTEASE

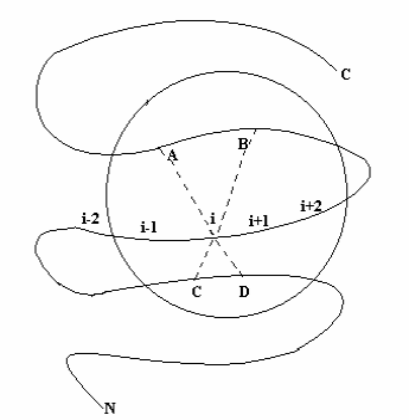

Fig. 1 Schematic representation of the spatial neighbors of amino acids in protein structure

The information of a protein structure is as follows (The meaning of each columns is shown in Table I):

ATOM 1 N GLY A 1 11.726 -10.369 10.598 1.00 12.32 N
ATOM 2 CA GLY A 1 11.567 -9.015 10.090 1.00 11.91 C
ATOM 3 C GLY A 1 11.280 -8.099 11.303 1.00 12.02 C
ATOM 4 O GLY A 1 11.256 -8.584 12.493 1.00 12.20 O
ATOM 5 N VAL A 2 11.060 -6.876 11.020 1.00 12.56 N
ATOM 6 CA VAL A 2 10.798 -5.882 12.075 1.00 15.09 C
ATOM 7 C VAL A 2 9.497 -5.127 11.777 1.00 14.36 C
ATOM 8 O VAL A 2 9.248 -4.650 10.670 1.00 14.42 O
ATOM 9 CB VAL A 2 12.004 -4.895 12.060 1.00 17.17 C

TABLE I
MEANING OF COLUMNS

1 – 6	Record name	*MASTER*	Meaning
11 – 15	Integer	numRemark	Number of REMARK records
16 – 20	Integer	″0″	
21 – 25	Integer	numHet	Number of HET records
26 – 30	Integer	numHelix	Number of HELIX records
31 – 35	Integer	numSheet	Number of SHEET records
36 – 40	Integer	numTurn	Deprecated
41 – 45	Integer	numSite	Number of SITE records
46 – 50	Integer	numXform	Number of coordinate transformation records (ORIGX + SCALE +MTRIX)
51 – 55	Integer	NumCoord	Number of atomic coordinate records records (ATOM +HETATM)
56 – 60	Integer	numTer	Number of TER records
61 – 65	Integer	numConect	Number of CONECT records
66 – 70	Integer	numSeq	Number of SEQRES records

The protein database is shown in Fig. 2.

proteinid	ATOM	serial	atomname	resiname	typ	resiseq	X_cord	Y_cord	Z_cord	tempf	element
1dua	ATOM	63	N	LYS	A	9	-5.046	15.782	33.105	0	N
1dua	ATOM	73	CA	HIS	A	10	-3.524	13.972	29.607	0	C
1dua	ATOM	103	O	LYS	A	13	-1.843	7.287	30.51	0	O
1dua	ATOM	107	CE	LYS	A	13	-8.159	9.609	28.626	0	C
1dua	ATOM	123	N	ASN	A	15	0.594	8.538	31.34	0	N
1dua	ATOM	147	CD2	TYR	A	17	-1.288	2.958	27.027	0	C
1dua	ATOM	152	N	TYR	A	18	3.134	4.828	29.942	0	N
1dua	ATOM	155	O	TYR	A	18	6.297	5.509	31.643	0	O
1dua	ATOM	156	CB	TYR	A	18	4.925	6.246	29.143	0	C
1dua	ATOM	173	CG1	VAL	A	20	3.139	0.417	33.572	0	C
1dua	ATOM	177	C	ASN	A	21	6.84	-1.342	31.945	0	C
1dua	ATOM	182	ND2	ASN	A	21	9.043	-3.759	32.823	0	N
1dua	ATOM	194	CD1	PHE	A	23	9.037	-7.676	29.619	0	C
1dua	ATOM	220	CG	PRO	A	26	-2.434	-2.699	31.803	0	C
1dua	ATOM	221	CD	PRO	A	26	-0.981	-2.522	32.148	0	C
1dua	ATOM	239	OE2	GLU	A	28	-6.924	-7.533	30.337	0	O
1dua	ATOM	254	CD1	PHE	A	30	2.423	-3.335	23.834	0	C
1dua	ATOM	277	O	VAL	A	33	2.287	-13.724	19.181	0	O
1dua	ATOM	303	CG	LYS	A	36	-3.177	-21.296	18.166	0	C
1dua	ATOM	308	CA	ASP	A	37	-3.022	-15.835	16.524	0	C
1dua	ATOM	309	C	ASP	A	37	-2.226	-15.242	15.384	0	C

Fig. 2 Protein database

We use the graph vertices to represent amino acid residues of protein [16]. Since the protein backbone defines the overall protein conformation, we choose the Cα atom to represent residue (amino acid).

link_id	graph_id	Amino_1	Amino_2	Amino_1_resiserial	Amino_2_resiserial	Distance
83810	1dua	HIS	ASN	1	43	8.309
83811	1dua	HIS	HIS	1	127	8.486
83812	1dua	ASP	TYR	2	3	6.745
83813	1dua	ASP	ASN	2	4	8.233
83814	1dua	ASP	TYR	2	8	7.903
83815	1dua	ASP	VAL	2	45	6.2
83816	1dua	TYR	ASP	3	2	6.745
83817	1dua	TYR	GLN	3	15	6.911
83818	1dua	TYR	VAL	3	45	8.029
83819	1dua	TYR	SER	3	49	7.668
83820	1dua	TYR	ILE	3	211	7.147
83821	1dua	TYR	MET	3	222	8.399
83822	1dua	ASN	ASP	4	2	8.233
83823	1dua	ASN	GLY	4	9	7.542
83824	1dua	ASN	PRO	4	94	6.935
83825	1dua	ASN	ILE	4	191	7.179
83826	1dua	VAL	GLY	5	26	6.741
83827	1dua	VAL	ASN	5	72	6.949
83828	1dua	VAL	PHE	5	74	8.448
83829	1dua	VAL	THR	5	81	6.437
83830	1dua	VAL	ALA	5	174	8.309
83831	1dua	VAL	ARG	5	177	6.369
83832	1dua	VAL	LEU	5	178	7.662
83833	1dua	THR	ASP	6	30	6.675

Fig. 3 Graph database of protein structures

Two vertices called residues are connected by a bond edge when these residues are consecutive in the primary sequence. Starting from this simplified protein model, we compute proximity edges (spatial neighbors of residues). Two vertices will be connected by an edge if the distance between them does not exceed a threshold δ. Since we are interested in neighboring residues (spatial neighbors) within a physical interaction radius, we chose δ to vary over values ranging from 6.5–8.5°A [6],[8]. The graph created from the proximity of Cα atoms of protein is shown in Fig. 4 and a portion of graph database of protein structures is shown in Fig. 3.

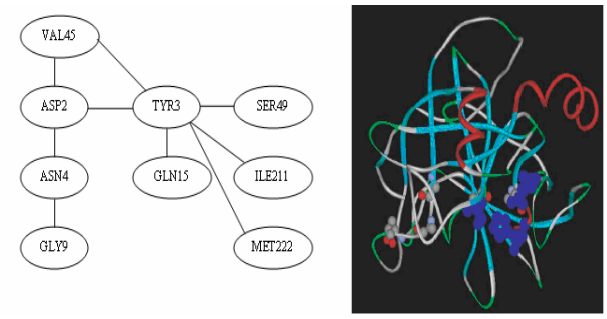

Fig. 4 A Protein structure to be represented by graph

III. Similarity between Two Graphs

Spectral graph theory is the study of the eigenvalues of matrix representing graphs [2],[3],[16]. There are several matrix representations of graphs. The first representation is the adjacency matrix as follows:

$$A_G(u,v) = \begin{Bmatrix} 1 : u \text{ is adjacent to } v \\ 0 : \text{otherwise} \end{Bmatrix}$$

Another representation of the normalized Laplacian as defined in Biggs [12] is as follows:

$$L_G(u,v) = \begin{Bmatrix} \text{degree}(v) : \text{if } u = v \\ -1 : \text{if } u \neq v \\ 0 : \text{otherwise} \end{Bmatrix}$$

Given graph G1 and G2 (Fig. 5) as follows:

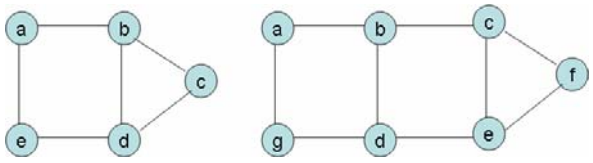

Fig. 5 Two compared graphs

The adjacency matrix of graph G1 is:

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	1	0
C	0	1	0	1	0
D	0	1	1	0	1
E	1	0	0	1	0

The normalized Laplacian representation of graph G1:

	A	B	C	D	E
A	2	-1	0	0	-1
B	-1	3	-1	-1	0
C	0	-1	2	-1	0
D	0	-1	-1	3	-1
E	-1	0	0	-1	2

A. Eigenvalues

Consider a square matrix A. λ is called as an eigenvalue of A if there exists a non-zero vector x such that $Ax = \lambda x$. In this case, x is called an eigenvector (corresponding to λ).

An example of eigenvalue and eigenvector are as follows. If

$$x = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

is an eigenvector corresponding to eigenvalue $\lambda = 0$ for

$$A = \begin{bmatrix} 6 & 3 \\ -2 & -1 \end{bmatrix}$$

we could find out the equation $Ax = \lambda x$:

$$Ax = \lambda x$$

$$\begin{bmatrix} 6 & 3 \\ -2 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -2 \end{bmatrix} = 0\begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Therefore, λ and x are eigenvalues and eigenvectors, respectively, for A.

We use Jacobi rotation algorithm to calculate eigenvalues of the normalized Laplacian matrix. The time complexity of Jacobi rotation algorithm is polynomial degree, then we sort eigenvalues in descendant order. The spectral vector of graph G1 is as follows:

{4.62; 3.62; 2.38; 1.38; 0.00}

The adjacency matrix of graph G2:

	A	B	C	D	E	F	G
A	0	1	0	0	0	0	1
B	1	0	1	1	0	0	0
C	0	1	0	0	1	1	0
D	0	1	0	0	1	0	1
E	0	0	1	1	0	1	0
F	0	0	1	0	1	0	0
G	1	0	0	1	0	0	0

The normalized Laplacian matrix:

	A	B	C	D	E	F	G
A	2	-1	0	0	0	0	-1
B	-1	3	-1	-1	0	0	0
C	0	-1	3	0	-1	-1	0
D	0	-1	0	3	-1	0	-1
E	0	0	-1	-1	3	-1	0
F	0	0	-1	0	-1	2	0
G	-1	0	0	-1	0	0	2

Sort the eigenvalues in descendant order, we have the spectral vector of graph G2

{5.25; 3.80; 3.55; 2.45; 2.20; 0.75; 0.00}

B. Eigen Distances

The distance calculation itself is simply the Euclidean distance between two spectral vectors. To account for the different number of eigenvalues in different sized graphs, a

Pad (v, x) function was defined which appended a number of x to the end of the shorter vector v until the appropriate length was reached for the longer vector [10]. In these above two graphs, the spectra vectors are as follows:

λ_{G1}= {4.62; 3.62; 2.38; 1.38; 0.00}
λ_{G2}= {5.25 ; 3.80; 3.55; 2.45; 2.20; 0.75; 0.00}

Since G1 has fewer eigenvalues, the corresponding spectral vector will be padded with 0 until the length 7 of G2:

λ_{G1}={4.62; 3.62; 2.38; 1.38; 0.00, 0.00; 0.00}
λ_{G2}= {5.25; 3.80; 3.55; 2.45; 2.20; 0.75; 0.00}

The eigen distance is the Euclidean distance between λ_{G1} and λ_{G2} is 2.89.

IV. Using M-Tree for Clustering and Similarity Search in Graph Databases

Suppose that we have a graph database with n protein structures. Each protein structure is represented by a graph. The normalized Laplacian representation of adjacency matrix for every graph is calculated. Then we use Jacobi rotation algorithm to calculate spectral vectors. For representing n protein structures, we use n spectral vectors. Let Lmax be the maximum number of components of n spectral vectors. We lengthen all spectral vectors with the number of components smaller than Lmax by padding with 0 at the right of vectors. A spectral vector is a data object of database. The M-tree is used for clustering spectral vectors representing the protein structure [4]. The M- tree partitions objects on the basis of their relative distances (Euclidean distance between spectral vectors). The M-tree organizes the data objects into fixed nodes. Each nodes can store up to M entries (the capacity of M-tree nodes). The data objects are stored in leaf nodes, whereas internal nodes store the so-called routing objects. A routing object is a database object to which a routing role is assigned by a specific promotion algorithm. An entry for a routing object Or also includes a pointer denoted ptr(T(Or)), which refers the root of a sub-tree, T(Or), called the covering tree of Or (see Fig. 6). A routing object Or hence defines a region in the metric space M, centered on Or and with radius r(Or). For each (ground) database object, one entry having the format entry(Oj)={Oj; oid(Oj); d(Oj); P(Oj))} is stored in a leaf node, where oid(Oj) is the identifier of the object, which is used for providing the access to the whole objects resident on a separate file (protein structure).

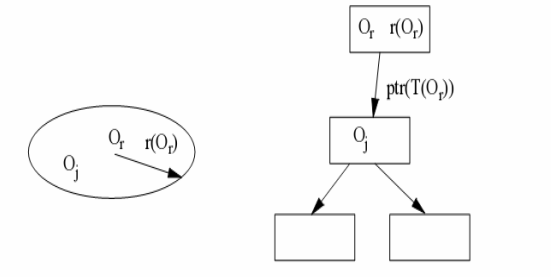

Fig. 6 The covering tree of Or.

All objects in the covering tree of Or are within the distance r(Or) from Or, r(Or) > 0 and r(Or) is called the covering radius of Or. Finally, a routing object O is associated with a distance to P(Or), its parent object, that is the routing object which references the node where the O entry is stored. To build M-tree Paolo used a batch bulk loading algorithm [5], [13].

The similarity query SimQuery(Q,rQ,C) requests for all database objects such that d(Oj,Q)<r(Q). Algorithm SimQuery starts from the root node and recursively traverses all the paths which can not be excluded from leading to objects satisfying d(Oj,Q) < r(Q) [13],[14].

V. Experiment and Discussion

A. Discussion of the Accuracy of M-tree Clustering

Classification of protein domains based on their tertiary structure provides a valuable resource that can be used to understand protein function and evolutionary relationships. SCOP is the most widely used database. SCOP is hierarchically organized and protein domains are used as basic units of classification. We recognize that a manual method was used to produce SCOP, so the quality of classification is very high. We use SCOP as a gold standard for measuring the quality of our proposed method. We also evaluated the quality of clusters using a measure called "cluster purity"[1]. It is 1 when all domains in the same cluster have perfect agreement in their class labels, and it is defined as:

$$ClusterPurity(M,S) = \frac{1}{N}\sum_{A\in M} \max_{B\in S} \left| \frac{A \cap B}{A} \right|$$

In the above equation, A is a cluster in the set of clusters M created by M-Tree, B is a cluster in the set of S families of SCOP and *N* is the cardinality of M. The SCOP structure is very large, so we select only the proteins and the clusters belonging to the test set. It means that, we have:

$$\bigcup_{A\in M} A = \bigcup_{B\in S} B$$

Cluster Purity is a value between 0 and 1. If cluster value is 1 then all the cluster A of M will be the subset of a cluster B of S. The higher value of Cluster Purity, the higher quality of cluster.

We use the proteins structural classification from SCOP database (see more at http://scop.mrc-lmb.cam.ac.uk/scop/data/scop.b.html) as gold standard to evaluate our method. We used both prokaryotic and eukaryotic serine proteases in the super family 'Trypsin-like serine proteases' (see more at http://scop.mrc-lmb.cam.ac.uk/scop/data/scop.b.c.hh.b.A.A.html) to test M-tree method. SCOP hierarchical structure [11] in the Superfamily 'Trypsin-like serine proteases' is described as follows:

Trypsin-like serine proteases [50494] (4)
1. Prokaryotic proteases [50495]
1.1 Achromobacter protease [50496]

1.1.1. Achromobacter lyticus[50497] (2)
1. 1arb
2. 1arc
1.2 alpha-Lytic protease [50498]
1.2.1. Lysobacter enzymogenes [50499] (41)
1. 2h5c
2. 1ssx
.....
1.3. Protease A [50500]
1.3.1.Streptomyces griseus [50501] (5)
1. 2sga
2. 3sga
.....
2. Eukaryotic proteases [50514]
2.1 Trypsin(ogen) [50515]
2.1. Cow (Bos taurus) [50516] (273)
1. 1hj9
2. 1utn
.....
2.2 Pig (Sus scrofa) [50517] (26)
1. 2a31
2. 2a32
.....
3. Viral proteases [50596]
…….
4. Viral cysteine protease of trypsin fold [50603]

According to SCOP, we have these proteins following arrangement into groups ([50497], [50499], [50501], etc..). We chose randomly some proteins belongs to groups and downloaded the accessible codes of proteins (the PDB files). The next, we built a database of protein graphs in DBMS SQL Server 2005 and generated links between nodes based on the proximity of amino acids (residues) in peptide chain to represent protein structure by graph. Neighboring residues within a physical interaction radius, we chose the threshold to vary over values ranging from 6.5–8.5°A and then used Jacobi rotation algorithm to calculate the eigenvalues of normalized Laplacian representation of adjacency matrix. Some eigenvalues of protein 1dua are shown in Fig. 7.

Proteinid	eigenvalue
1dua	16.64
1dua	15.85
1dua	15.49
1dua	15.07
1dua	14.55
1dua	14.46
1dua	14.22
1dua	14.12
1dua	13.97
1dua	13.85
1dua	13.64
1dua	13.54
1dua	13.26
1dua	13.13
1dua	12.89
1dua	12.74
1dua	12.65
1dua	12.51
1dua	12.45

Fig. 7 The result of calculating egienvalues of graphs

We chose a test data set with 100 proteins. Let S={S1,S2,…, S9} be a set of clusters of SCOP containing these proteins. Then, we use M-tree for clustering these proteins. Let M={M_1, M_2, M_3,…, M_8} be a set of clusters of M-Tree. We use the formula of Cluster Purity to calculate the values in Table II. In this table, the value of each cell is ratio between the cardinality of the intersection between cluster M_i and S_j and the cardinality of cluster M_i. The last row contains the maximum value for each column.

TABLE II
THE CLUSTER PURITY

MTREE / SCOP	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
S_1	0.25	0.25	1	1	0	0	0	0
S_2	0.75	0	0	0	0	0	0	0
S_3	0	0.25	0	0	0	0	0.33	0
S_4	0	0	0	0	0	0	0	1
S_5	0	0	0	0	0	1	0	0
S_6	0	0	0	0	0	0	0	0
S_7	0	0.5	0	0	0	0	0.33	0
S_8	0	0	0	0	0	0	0.33	0
S_9	0	0	0	0	0	0	0	0
Max	**0.75**	**0.5**	**1**	**1**	**1**	**1**	**0.33**	**1**

Using this measure, the cluster purity of the M-Tree clusters is as follows:

$$\frac{1}{8}(0.75+0.5+1+1+1+1+0.33+1)=0.8225$$

This high cluster purity value shows that our clustering method produces clusters that have a high degree of agreement with the SCOP classes. When processing a similarity search, a query is submitted to the M-tree. A graph representing for this protein will be created and the spectral vector for this matrix will be generated by Jacobi rotation algorithm. After padding this spectral vector with 0 to lengthen the length of vector to LMax, this vector will be used as a query for searching in M-tree.

B. Discussion of Processing Time

The time complexity of Jacobi rotation algorithm for calculating eigenvalues is polynomial degree. The time duration of creating the graphs from PDB data and the spectral vectors for 100 protein structures is about 30 minutes. The time duration of creating M-trees for 100 spectral vectors is 2 minutes. We believe that this time duration is reasonable. Moreover, M-tree is a fast structure for indexing a large volume of data [12],[13]. So, we chose M-tree for clustering and similarity search in protein structure databases. Besides, M-tree is an incremental algorithm, so we can add more protein structure incrementally without running M-tree for whole new data.

VI. CONCLUSION

In this paper, we propose a method of graph clustering and

searching in protein structure database. Jacobi rotation algorithm is used to calculate the eigenvalues of the normalized Laplacian representation of adjacency matrix of graph representing protein structure**.** We use Euclidean distance between two spectral vectors to measure similarity between graphs. A M-tree structure is used to index the spectral vectors and to increase efficiency of similarity searching in protein structure graph database. Experiment results encourage us to develop more applications of graph indexing and graph searching.

ACKNOWLEDGMENT

We would like to express our thanks to the Department of Information Systems, University of information technology and School of Computing, NUS for their support and the valuable comments of the reviewers.

REFERENCES

[1] Brew C, Schulte im Walde (2002). Spectral Clustering for German Verbs, In Proc of the Conf in Natural Language Processing, PA, USA, pp 117-124
[2] Chris Godsil (2006). Graph Spectra and Graph Isomorphism. Aveiro Workshop on Graph Spectra, University of Aveiro, Mathematics Department, April 2006.
[3] David McWherter, William C. Regli (2001). An Approach to Indexing Database of Solid Models. Technical Report DU-MCS-01-02.
[4] D. Shasha, J. T. L. Wang, and R. Giugno (2002). Algorithmics and Applications of Tree and Graph Searching. In Proc. PODS'02 Proceeding of the International Conference in Pattern recognition (ICPR), Quebec, Canada, August 2002.
[5] Do Phuc, Hoang Kiem (2006) . Using M-tree for similarity search in biological sequence databases, Journal of Bio-Technology, VAST, ISBN 1811-4899, pp151-158
[6] Huan, J., Wang, W., Washington, A., Prins, J., Shah, R., Tropsha, A. (2004). Accurate classification of proteins tructural families based on coherent subgraph analysis. In Proc. Pacific Symposium on Biocomputing 9:411-422.
[7] Huahai He, Ambuj K. Singh (2006). Closure-Tree: An Index Structure for Graph Queries, ICDE.
[8] Jun Huan, Deepak Bandyopadhyay, Wei Wang, Jack Snoeyink, Jan Prins, Alexander Tropsha (2005). Comparing Graph Representations of Protein Structure for Mining Family-Specific Residue-Based Packing Motifs. Journal of Computational Biology. July 2005, 12(6): 657-671.
[9] James Cheng, Yiping Ke, Wilfred Ng, An Lu (2007). FG-Index: Towards Verification-Free Query Processing on Graph Databases, SIGMOD, 2007.
[10] Mitchell Peabody (2002). Finding Groups of Graphs in Databases. Master thesis in Computer Science, Drexel University, August, 2002.
[11] Murzin, A.G, Brenner, S., Hubbard, T. & Chothia., C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. Journal of MolecularBiology, 247, 536–40.
[12] Norman Biggs. Algebraic Graph Theory. Cambridge University Press, 2nd edition, 1993.
[13] Paolo C., Marco P., Pavel Z (1997): M-tree: An efficient Access Method for Similarity Search In Metric Spaces, VLDB 1997, pp:426-435.
[14] Patella M (1998). Similarity search in multimedia database, PhD dissertation, University of Bolgona, Italia.
[15] Peixiang Zhao, Jeffrey Xu Yu, Philip S. Yu (2007). Graph Indexing: Tree + Delta >= Tree, VLDB, 2007
[16] Steffen Lang (2007). Protein domain decomposition using spectral graph partitioning.
[17] Saraswathi Vishveshwara et al (2002). Protein structure insights from graph theory, Journal of Theoretical and Computational Chemistry, Vol 1, No 1.
[18] X. Yan, J. Han (2002). gSpan: Graph-based substructure pattern mining, ICDM, 2002
[19] Xiafeng Yan, Philip S. Yu, Jiawei Han (2004). Graph indexing: A Frequent Structure-based Approach, SIGMOD 2004
[20] Yun Chi, Yirong Yang, Richard Muntz, Mining Frequent Rooted Trees and Free Trees Using Canonical Forms, UCLA Technical Report, 2003