Searching Protein 3-D Structures for Optimal Structure Alignment Using Intelligent Algorithms and Data Structures

Tomáš Novosád, Václav Snášel, Ajith Abraham, *Senior Member, IEEE*, and Jack Y. Yang

***Abstract*—In this paper, we present a novel algorithm for measuring protein similarity based on their 3-D structure (protein tertiary structure). The algorithm used a suffix tree for discovering common parts of main chains of all proteins appearing in the current research collaboratory for structural bioinformatics protein data bank (PDB). By identifying these common parts, we build a vector model and use some classical information retrieval (IR) algorithms based on the vector model to measure the similarity between proteins—all to all protein similarity. For the calculation of protein similarity, we use *term frequency* × *inverse document frequency* (tf × idf) term weighing schema and cosine similarity measure. The goal of this paper is to introduce new protein similarity metric based on suffix trees and IR methods. Whole current PDB database was used to demonstrate very good time complexity of the algorithm as well as high precision. We have chosen the structural classification of proteins (SCOP) database for verification of the precision of our algorithm because it is maintained primarily by humans. The next success of this paper would be the ability to determine SCOP categories of proteins not included in the latest version of the SCOP database (v. 1.75) with nearly 100% precision.**

***Index Terms*—Information retrieval (IR), proteins, structural classification of proteins (SCOP), similarity measure, suffix trees, vector model.**

I. Introduction

ANALYZING 3-D protein structures is a very important task in molecular biology. Nowadays, the solution for protein structures often stems from the use of the state-of-the-art technologies, such as nuclear magnetic resonance (NMR) spectroscopy techniques or X-ray crystallography, etc., as seen in the increasing number of protein data bank (PDB) [1] entries (65527 as of May 25, 2010). PDB is a database of 3-D structural data of large biological molecules, such as proteins and nucleic acids. It was proved that structurally similar proteins tend to have similar functions even if their amino acid sequences are not similar to one another. Thus, it is very important to find proteins with similar structures (even in part) from the growing database to analyze protein functions. Yang *et al.* [2] exploited machine-learning techniques including variants of self-organizing global ranking, a decision tree, and support vector machine (SVM) algorithms to predict the tertiary structure of transmembrane proteins. Hecker *et al.* [3] developed a state-of-the-art protein disorder predictor and tested it on a large protein disorder dataset created from the PDB. The relationship of sensitivity and specificity is also evaluated. Habib *et al.* [4] presented a new SVM-based approach to predict the subcellular locations based on amino acid and amino acid pair composition. More protein features can be taken into consideration to improve the accuracy significantly. Wang *et al.* [5] discussed an empirical approach to specify the localization of protein binding regions utilizing information including the distribution pattern of the detected RNA fragments and the sequence specificity of RNase digestion.

In this paper, we present a novel method for analyzing 3-D protein structure using suffix trees and classical information retrieval (IR) methods and schemes. Several studies were developed for indexing protein tertiary structure [6], [7]. These studies are targeted mainly at some kind of selection of the PDB database. The goal of this paper is to introduce new protein similarity metric and show the usage of the algorithm on the whole-current PDB database and calculate the similarity of each protein in comparison with other proteins. The suffix tree is a very useful data structure, which can discover common substructures of proteins within a reasonable time (linear or logarithmic time), depending on the implementation of the construction algorithm. In this research, the similarity between any two proteins is based on common parts (substructures) of protein main chains instead of, for example, root mean square deviation (RMSD), which calculates the protein similarity as a distance between pairs of equivalent atoms (usually C_α).

When the generalized suffix tree is constructed for all proteins that appear in the entire PDB database, we use similar methods, which were previously studied [8]–[12] for measuring the similarity of proteins based on their 3-D structure definition. Our paper arises from the relations of amino acid residues defined by its torsion angles rather than the relations just between the alpha carbon atoms. The relations between alpha carbons use Distance-matrix ALIgnment, for example, [13], when computing the distance matrix between alpha carbon atoms of a given protein. In the final stage, we build a vector space model, which is very suitable for various IR tasks and can be used for future studies of proteins relations.

Manuscript received January 26, 2010; revised June 4, 2010; accepted September 10, 2010. Date of publication September 27, 2010; date of current version November 5, 2010.

T. Novosád and V. Snášel are with Department of Computer Science, Vysoká Škola Báňská—Technical University of Ostrava, Ostrava 70833, Czech Republic (e-mail: tomas.novosad@vsb.cz; vaclav.snasel@vsb.cz).

A. Abraham is with the Machine Intelligence Research Labs, Auburn, WA 98071-2259 USA (e-mail: ajith.abraham@ieee.org).

J. Y. Yang is with the Harvard University, Cambridge, MA 02140-0888 USA (e-mail: dr.yang@jhu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITB.2010.2079939

The paper is organized as follows. In Section II, we briefly describe the proteins, vector space model, and suffix trees. Section III deals with the description of the conversion of 3-D protein backbone structures into the data, which can be indexed by suffix trees. In the following Section IV, we introduce the characterization of the proposed algorithm for measuring the similarity of proteins using the vector space model. Finally, in Section V, the evaluation of the proposed algorithm in comparison with structural classification of proteins (SCOP) database is provided.

II. Background

A. Protein Structure

Proteins are large molecules. In many cases, only a small part of the structure—*an active site*—is directly functional, the rest exists only to create and fix the spatial relationship among the active site residues [14]. Chemically, protein molecules are long polymers typically containing several thousand atoms, composed of a uniform repetitive backbone (or main chain) with a particular side chain attached to each residue. The amino acid sequence of a protein records the succession of side chains.

The polypeptide chain folds into a curve in space; the course of the chain defines a *folding pattern*. Proteins show a great variety of folding patterns. Underlying these are a number of common structural features. These include the recurrence of explicit structural paradigms—for example, $\alpha - helices$ and $\beta - sheets$ and common principles or features such as the dense packing of the atoms in protein interiors. Folding may be thought of as a kind of intramolecular condensation or crystallization [14].

1) Protein Databank: The PDB archive contains information about experimentally determined structures of proteins, nucleic acids, and complex assemblies. As a member of the wwwPDB, the RCSB PDB curates and annotates PDB data according to agreed upon standards [1].

2) Torsion Angles: Any plane can be defined by two noncollinear vectors lying in this plane; taking their cross product and normalizing yields the normal unit vector to the plane. Thus, a torsion angle can be defined by four, pairwise noncollinear vectors.

The backbone torsion angles of proteins are called ϕ (phi, involving the backbone atoms C-N-C_α-C), ψ (psi, involving the backbone atoms N-C_α-C-N), and ω (omega, involving the backbone atoms C_α-C-N-C_α). Thus, ϕ controls the C–C distance, ψ controls the N–N distance, and ω controls the C_α–C_α distance.

The planarity of the peptide bond usually restricts ω to be 180° (the typical trans case) or 0° (the rare cis case). The ϕ and ψ torsion angles tend to be from –180° to 180°.

B. Vector Space Model

The vector model [15] of documents was established in the 1970's. A document in the vector model is represented as a vector. Each dimension of this vector corresponds to a separate term appearing in document collection. If a term occurs in the document, its value in the vector is nonzero.

We use m different terms $t_1, \ldots, t_m$ for indexing N documents. Then, each document d_i is represented by a vector

$$d_i = (w_{i1}, w_{i2}, \ldots, w_{im})$$

where w_{ij} is the weight of the term t_j in the document d_i. The weight of the term in the document vector can be determined in many ways. A common approach uses the so-called term frequency $\times$ inverse document frequency (tf $\times$ idf) method, in which the weight of the term is determined by these factors: how often the term t_j occurs in the document d_i (the term frequency tf_{ij}) and how often it occurs in the whole document collection (the document frequency df_j. Precisely, the weight of the term t_j in the document d_i is as follows [16]:

$$w_{ij} = \mathrm{tf}_{ij} \times \mathrm{idf}_j = \mathrm{tf}_{ij} \times \log \frac{n}{\mathrm{df}_j} \tag{1}$$

where idf stands for the inverse document frequency. This method assigns high weights to terms that appear frequently in a small number of documents in the document set.

An index file of the vector model is represented by matrix

$$D = \begin{pmatrix} w_{11} & w_{12} & \ldots & w_{1m} \\ w_{21} & w_{22} & \ldots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \ldots & w_{Nm} \end{pmatrix}$$

where ith row matches ith document, and jth column matches jth term.

The similarity of two documents in vector model is usually given by the following formula—cosine similarity measure:

$$\mathrm{sim}(d_i, d_j) = \frac{\sum_{k=1}^{m} (w_{ik} w_{jk})}{\sqrt{\sum_{k=1}^{m} (w_{ik})^2 \sum_{k=1}^{m} (w_{jk})^2}}. \tag{2}$$

For more information, see [15], [17], [18].

C. Suffix Trees

A suffix tree is a data structure that allows efficient string matching and querying. Suffix trees have been studied and used extensively, and have been applied to fundamental string problems, such as finding the longest repeated substring [19], strings comparisons [20], and text compression [21]. Following this, we describe the suffix tree data structure—its definition, construction algorithms, and main characteristics.

1) Definitions: The following description of the suffix tree was taken from Gusfield's book *Algorithms on Strings, Trees, and Sequences* [22]. Suffix trees commonly dealing with strings as sequence of characters. One major difference is that we treat documents as sequences of words, not characters. A suffix tree of a string is simply a compact trie of all the suffixes of this string [23].

Definition 2.1: A suffix tree T for an m-word string S is a rooted directed tree with exactly m leaves numbered 1 to m. Each internal node, other than the root, has at least two children and each edge is labeled with a nonempty substring of words of S. No two edges out of a node can have edge labels beginning with the same word. The key feature of the suffix tree is that for any leaf i, the concatenation of the edge labels on the path from

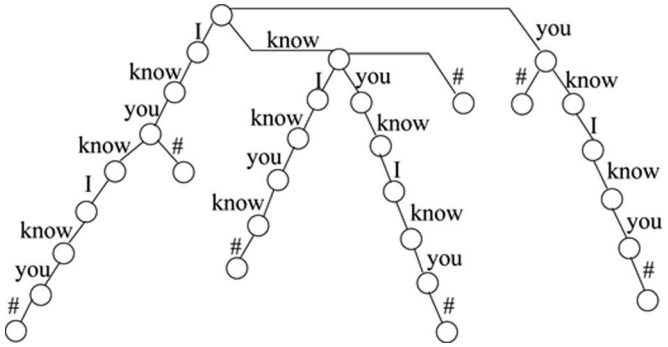

Fig. 1. Simple example of suffix trie.

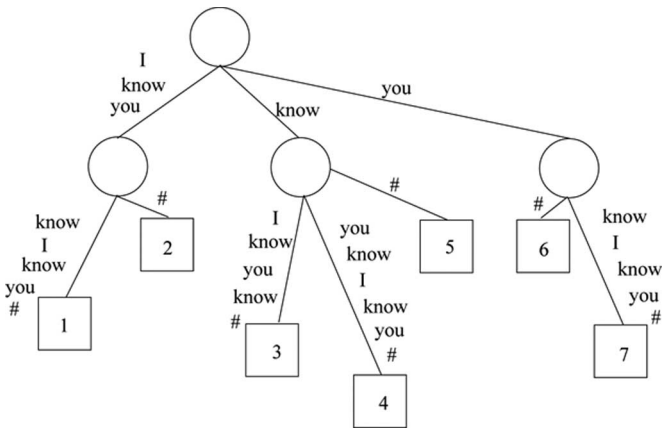

Fig. 2. Simple example of suffix tree.

the root to leaf i exactly spells out the suffix of S that starts at position i, that is it spells out $S[i \ldots m]$.

In cases, where one suffix of S matches a prefix of another suffix of S, then no suffix tree obeying the aforementioned definition is possible, since the path for the first suffix would not end at a leaf. To avoid this, we assume the last word of S does not appear anywhere else in the string. This prevents any suffix from being a prefix to another suffix. To achieve this, we can add a terminating character, which is not in the language that S is taken from, to the end of S.

Example of suffix trie of the string *"I know you know I know you#"* is shown in Fig. 1. Corresponding suffix tree of the string *"I know you know I know you#"* is presented in Fig. 2. There are seven leaves in this example, marked as rectangles and numbered from 1 to 7. The terminating characters are also shown in Fig. 2.

In a similar manner, a suffix tree of a set of strings, called a generalized suffix tree [22], is a compact trie of all the suffixes of all the strings in the set [23].

Definition 2.2: A generalized suffix tree T for a set S of n strings S_n, each of length m_n, is a rooted directed tree with exactly $\sum m_n$ leaves marked by a two number tuple (k, l), where *k* ranges from 1 to n and l ranges from 1 to m_k. Each internal node, other than the root, has at least two children and each edge is labeled with a nonempty substring of words of a string in S. No two edges out of a node can have edge labels beginning with the same word. For any leaf (i, j), the

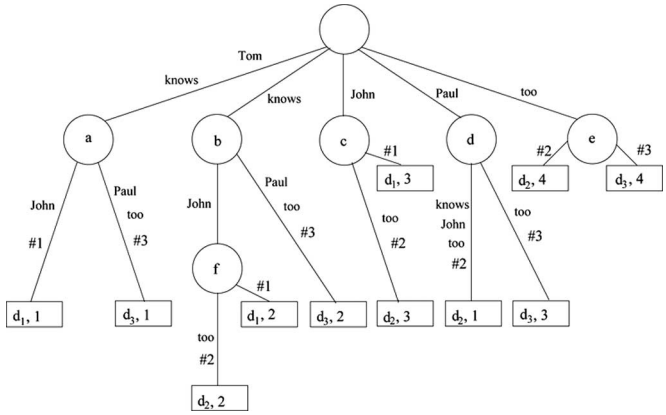

Fig. 3. Example of the generalized suffix tree.

concatenation of the edge labels on the path from the root to leaf (i, j) exactly spells out the suffix of S_i that starts at position j, i.e., it spells out $S_i[j \ldots m_i]$.

Fig. 3 is an example of a generalized suffix tree of the set of three strings—*"Tom knows John #1," "Paul knows John too #2,"* and *"Tom knows Paul too #3"* (#1, #2, and #3 are unique terminating symbols). The internal nodes of the suffix tree are drawn as circles, and are labeled from *a* to *f* for further reference. Leaves are drawn as rectangles. The first number $d_i = (d_1, \ldots, d_n)$ in each rectangle indicates the string from which that suffix originates—a unique number that identifies the string. The second number represents the position in that string d_i, where the suffix begins. Each string is considered to have a unique terminating symbol.

2) Suffix Tree Construction Algorithms: The naive, straightforward method to build a suffix tree for a string S of length L takes $O(L^2)$ time. The naive method first enters a single edge for the suffix $S[1 \ldots L]$ into the tree. Then, it successively enters the suffix $S[i \ldots L]$ into the growing tree for i increasing from 2 to L. The details of this construction method are not within the bounds of this paper. Various suffix tree construction algorithms can be found in [22] (a good book on suffix tree construction algorithms in general).

Several linear time algorithms for constructing suffix trees exist [19], [24], [25]. To be precise, these algorithms also exhibit a time dependency on the size of the vocabulary (or the alphabet when dealing with character-based trees): they actually have a time bound of $O(L \times \min(\log |V|, \log L))$, where L is the length of the string and $|V|$ is the size of the language. These methods are more difficult to implement then the naive method, which is sufficiently suitable for our purpose.

We have also made some implementation improvements of the naive method to achieve better than the O(L^2) worst case time bound. With these improvements, we have achieved constant access time for finding an appropriate child of the root (this is important because the root node has the same count of child nodes as it is the size of the alphabet—count of terms in document collection) and logarithmic time to find an existing child or to insert a new child node to any other internal nodes of the tree [11]. Then, we have also improved the generalized

suffix tree data structure to be suitable for large document collections [11].

III. Preparing the Data

We describe the process of retrieving the data for protein indexing. We used the whole-current PDB database, which consists of 65 527 known proteins, nucleic acids and complex assemblies, as of Tuesday May 25, 2010.

A. Creating Proteins Collection

In the current PDB database, we can find proteins, nucleic acids, and complex assemblies. Our study is just focused on the relations between proteins. We have filtered out all nucleic acids and complex assemblies from the entire PDB database. Then, we have filtered out proteins, which have incomplete N-Cα-C-O backbones (e.g., some of the files have C atoms in the protein backbone missing, etc.).

After this cleaning step, we have obtained a collection of 60 244 files. Each file contains a description of a specific protein and its 3-D structure and contain only amino acid residues with complete a N-Cα-C-O atom sequence.

Each retrieved file had at least one main chain (some proteins have more than one main chain) of at least one model (some PDB files contained more models of 3-D protein structure). In cases, when the PDB file contained multiple chains or models, we took into account all of those (all main chains of all models).

B. Encoding the 3-D Protein Main Chain Structure for Indexing

To be able to index proteins by IR techniques, we need to encode the 3-D structure of the protein backbone into some sequence of characters, words, or integers (as in our case). The area of protein 3-D structure encoding has been widely studied by authors in previous works, e.g., [6], [26], [27]. Since the protein backbone is the sequence of the amino acid residues (in 3-D space), we are able to encode this backbone into the sequence of integers in the following manner.

For example, let us say the protein backbone consists of four amino acid residues *M V L S* (abbreviations for methionine, valine, leucine, and serine). The relationship between the two following residues can be described by its torsion angles ϕ, ψ, and ω. Since ϕ and ψ are taking values from the interval $\langle -180^\circ, 180^\circ \rangle$, we have to do some normalization. From this interval, we have obtained 37 values (the interval was divided into 36 equal sized subintervals, by 10°), e.g., -180°, -170°,. . .,0°, 10°,. . ., 180°. Each of these values was labeled with nonnegative integers as follows: 00, 01,. . ., 36, where 00 stands for -180°. Now, let us say that ϕ is -21°, the closest discrete value is -20°, which has the label 16, so we have encoded this torsion with the string "16." The same holds for ψ. Torsion angle ω was encoded as the two characters A or B, since the ω tends to be almost in every case 0° or 180°. After concatenation of these three parts, we get a string, which looks something like this "A0102," which means that $\omega \approx 180^\circ$, $\phi \approx -170^\circ$, and $\psi \approx -160^\circ$. Concatenation was done in the following manner: $\omega\phi\psi$.

C. Indexing

The objective of this stage is to prepare the data for indexing by suffix trees. The suffix tree can index sequences. The resulting sequence in our case is a sequence of nonnegative integers. For example, let us say we have a protein with a backbone consisting of six residues, e.g., M V L S E G with its 3-D properties. The resulting encoded sequence can be, for example,

$$\{\text{A3202, A2401, A2603, A2401, A2422}\}.$$

After obtaining this sequence of five words, we create a dictionary of these words (each unique word receives its own unique nonnegative integer identifier). The translated sequence appears as follows:

$$\{0, 1, 2, 1, 3\}.$$

In this way, we encode each main chain of each model contained into one PDB file. This task is done for every protein included in our filtered PDB collection. Now, we are ready for indexing proteins using suffix trees.

IV. Protein Similarity Algorithm

We describe the algorithm for measuring protein similarity based on their tertiary structure. A brief description of the algorithm follows.

1) Prepare the data as was discussed in Section III.
2) Insert all encoded main chains of all proteins in the collection into the generalized suffix tree data structure.
3) Find all maximal substructure clusters in the suffix tree.
4) Construct a vector model of all proteins in our collection.
5) Build proteins similarity matrix.
6) For each protein, find top *N* similar proteins.

A. Inserting All Main Chains Into the Suffix Tree

At this stage of the algorithm, we construct a generalized suffix tree of all encoded main chains. As mentioned in Section III, we obtain the encoded forms of 3-D protein main chains—sequences of positive numbers. All of these sequences are inserted into the generalized suffix tree data structure (see Section II-C).

B. Finding All Maximal Substructure Clusters

To be able to build a vector model of proteins, we have to find all maximal phrase clusters. Recall the example given in Section III: the phrases can be, e.g., *"Tom knows John #1," "knows John #1," "John #1,"* etc. (just imagine that *"Tom knows John #1"* is equal to *"A3202 A2401 A2603 #1"*). The *phrase* in our context is an encoded protein main chain or any of its parts. The document in our context can be seen as a set of encoded main chains of the protein. Now, we can define a maximal phrase cluster (the longest common substructure) [8].

Definition 4.1: A phrase cluster is a phrase that is shared by at least two documents, and the group of documents that contain the phrase. A maximal phrase cluster is a phrase cluster, whose phrase cannot be extended by any word in the language without changing (reducing) the group of documents that contain it. Maximal phrase clusters are those we are interested in.

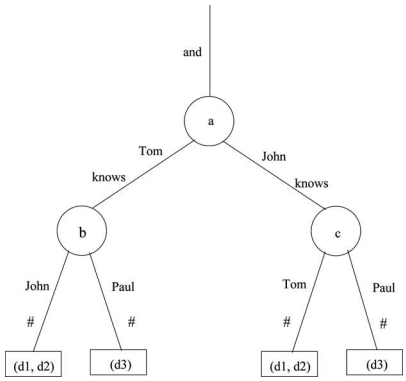

Fig. 4. Example of maximal phrase cluster—node *b* and *c*.

Now, we simply traverse the generalized suffix tree and identify all maximal phrase clusters (i.e., all of the longest common substructures). Fig. 4 displays the example of maximal phrase cluster.

C. Building a Vector Model

We describe the procedure for building the matrix representing the vector model index file (see Section II-B). In a classical vector space model, the document is represented by the terms (which are words), respectively, and by the weights of the terms. *In our model, the document is represented not by the terms but by the common phrases (maximal phrase clusters)!*—the term in our context is a common phrase, i.e., maximal phrase cluster.

In the previous stage of the algorithm, we have identified all maximal phrase clusters—all the longest common substructures. From the definition of the phrase cluster, we know that the phrase cluster is the group of the documents sharing the same phrase (group of proteins sharing the same substructure). Now, we can obtain the matrix representing the vector model index file directly from the generalized suffix tree. Each document (protein) is represented by the maximal phrase clusters in which it is contained. For computing the weights of the phrase clusters, we are using a tf $\times$ idf weighting schema, as given by (1).

Simple example: Let us say that we have a phrase cluster containing documents d_i. These documents share the same phrase t_j. We compute w_{ij} values for all documents appearing in a phrase cluster sharing the phrase t_j. This task is done for all phrase clusters identified by the previous stage of the algorithm.

Now, we have a complete matrix representing the index file in a vector space model (see Section II-B).

D. Building a Similarity Matrix

In the previous stage of the algorithm, we have constructed a vector model index file. To build a protein similarity matrix, we use standard IR techniques for measuring the similarity in a vector space model. As mentioned in Section II-B, we have used cosine similarity, which looks quite suitable for our purpose. The similarity matrix is given by

Documents (proteins) similarity matrix

$$S = \begin{pmatrix} 1 & \mathrm{sim}(d_1, d_2) & \dots & \mathrm{sim}(d_1, d_n) \\ \mathrm{sim}(d_2, d_1) & 1 & \dots & \mathrm{sim}(d_2, d_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{sim}(d_n, d_1) & \mathrm{sim}(d_n, d_2) & \dots & 1 \end{pmatrix}$$

where the ith row matches the ith document (protein, respectively), and the jth column matches the jth document (protein). The similarity matrix is diagonally symmetrical.

E. Finding Similar Proteins

This step is quite simple. When we have computed the similarity matrix S, we simply sort the documents (proteins) on each row, according to their similarity scores. The higher the score, the more similar the two proteins are. This is done for each protein in our protein collection.

V. Evaluation and Testing

A. Structural Classification of Proteins

In order to evaluate the accuracy and effectiveness of our algorithm, we applied it to well-known SCOP [28] database, which is maintained by humans in contrast with, for example, CATH database, which uses automated methods. We used the current version of the SCOP database (v. 1.75 released on June 2009), which contains 38 221 of classified proteins. We have chosen SCOP, because we wanted to evaluate our algorithm on manually classified proteins, rather than on automated ones.

There is also another structural classification system called CATH. CATH is an hierarchical classification of protein-domain structures, which clusters proteins at four major levels: class (C), architecture (A), topology (T), and homologous superfamily (H). The boundaries and assignments for each protein domain are determined using a combination of automated and manual procedures, which include computational techniques, empirical and statistical evidence, literature review, and expert analysis [29]. CATH uses the DALI algorithm to find similarities between proteins.

B. Evaluation

For each protein P in our collection C, we did the following.

1) For protein P, determine the class, folding pattern group, superfamily, family, and domain [28].
2) Based on the similarity matrix, find N most similar proteins P_S according to their score of similarity to protein P.
3) For each protein P_S, determine the class, folding pattern group, superfamily, family, and domain.
4) For all proteins in our collection, compute the percentage of correctly classified proteins P_S to protein P.

We did this for each protein in our collection and computed the overall percentage accuracy over our filtered collection. We did not classify approximately 25 000 proteins, since they did not appear in the SCOP database. The highest possible score for two proteins is 1.0, which means the two proteins are completely similar. The score of 0.0 means the two proteins have

TABLE I
TEN THE MOST SIMILAR PROTEINS TO A PROTEIN LABELED *2ENG* AND THE SCOP CLASSIFICATIONS

Rank	PDB	Score	Class	Fold	SuperF	Family	Domain
	2eng	**1.0000**	**48724**	**50684**	**50685**	**50686**	**50687**
1	3eng	0.7277	48724	50684	50685	50686	50687
2	4eng	0.7152	48724	50684	50685	50686	50687
3	1hd5	0.6783	48724	50684	50685	50686	50687
4	1l8f	0.3355	48724	50684	50685	50686	50687
5	1oa7	0.3289	48724	50684	50685	50686	50687
6	1oa9	0.3283	48724	50684	50685	50686	50687
7	1du5	0.1107	48724	49869	49870	49871	49874
8	1aun	0.1058	48724	49869	49870	49871	49872
9	2r01	0.1057	0	0	0	0	0
10	3hbc	0.1051	0	0	0	0	0

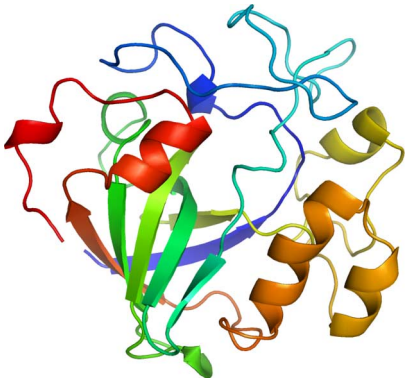

Fig. 5. (2ENG) Endoglucanase V.

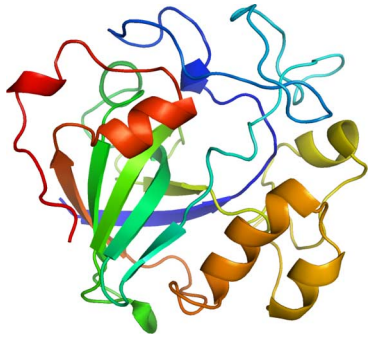

Fig. 6. (1st—3ENG) Structure of endoglucanase V cellobiose complex.

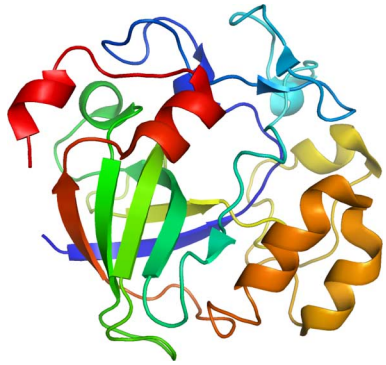

Fig. 7. (4th—1l8f) Structure of 20K-endoglucanase from *Melanocarpus albomyces* at 1.8A.

Fig. 8. (8th—1aun) Pathogenesis-related protein 5-D from *Nicotiana tabacum*.

no structural similarity discovered by our algorithm. The score can also be seen as percentage of similarity if it is multiplied by 100.

In more precise terms: if we have protein P, based on the calculated similarity matrix, we sort all other proteins P_S in our protein collection in descending order according to their scores. The greater the score the more similar the protein is to protein P. We take only the top N highest scoring proteins (the top N most similar proteins to the given protein). We set N to the value of ten. After this, we obtain a list such that the similar proteins for every protein in our collection, we have determined the SCOP classification of those proteins. Table I depicts the algorithm result for the protein marked with the label *2ENG* (endoglucanase V). The class with id = 48 724 means *all beta proteins*, the fold with id = 50 684 means *double psi beta-barrel*, the superfamily with id = 50 685 means *Barwin-like endoglucanases*, the family with id = 46 463 means *Eng V-like*, the fomain with id = 50 687 means *endoglucanase V (Eng V)*, and finally, the species with id = 50 688 means *Humicola insolens*.

Note that, e.g., class id = 0 means that the protein is not classified by the SCOP database.

Table I displays the ten most similar proteins to protein labeled *2ENG* and it is generated for every protein in our filtered protein collection. Fig. 5 shows the picture of secondary structure of protein marked *2ENG* in PDB database. Figs. 6–8 represent the three structurally similar proteins to protein labeled *2ENG*, corresponding to Table I. Images were generated using PyMOL software.

C. Experiments

Here, we present the results with the proposed method of measuring protein similarity based on their tertiary structure and in the comparison with the SCOP database. All experiments were run on computer with 32 GB of RAM and 4 AMD 64-bit Opteron dual core 1.8 GHz CPUs. The whole PDB database

TABLE II
CLASS CLASSIFICATION PERCENTAGE ACCURACY

	Similarity Score Cutoff					
Rank	**0.00**	**0.14**	**0.28**	**0.42**	**0.56**	**0.70**
1	93.02	96.44	98.36	98.64	99.04	99.19
2	88.61	91.89	94.65	95.63	97.02	98.25
3	85.14	88.30	91.32	92.37	94.64	97.39
4	82.15	85.07	88.16	89.44	92.16	96.10
5	79.74	82.64	85.74	87.15	89.96	94.93
6	77.43	80.17	83.39	84.83	87.83	93.75
7	76.30	78.94	81.74	83.17	86.44	92.63
8	74.37	76.98	79.86	81.36	84.25	91.08
9	73.32	75.82	78.68	80.09	83.21	90.46
10	71.76	74.27	77.37	78.86	81.97	89.67
Count	35908	33011	29018	26756	22049	12291

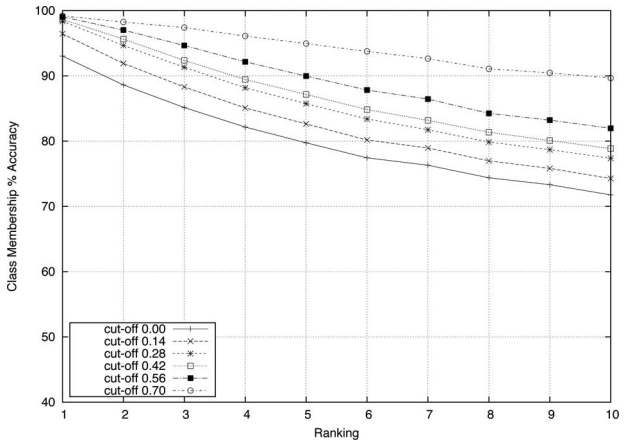

Fig. 9. Protein class membership percentage accuracy.

indexed by our version of the suffix tree construction algorithm, which takes about 3.5 GB of RAM and about 40 min of time (see Section IV-A). The calculation of the similarity matrix in Section IV-D takes about 10 h of time and 10 GB of RAM, since the similarity matrix is computed in memory. Total time needed to compute all pairwise similarities is about 12 h for the whole current PDB database. All calculations were done on one 1.8 GHz CPU using only one thread. All parts of the application is written in C++ programming language.

First, we have computed a percentage accuracy of all proteins in the entire SCOP database (35 908 proteins classified), next we have computed the accuracy only for proteins for which our algorithm found proteins with at least some given score of similarity (e.g., we have protein A and for this protein exists at least one protein, which has a score of similarity with protein A of at least 0.14—we cutoff all proteins, which do not satisfy this assumption)—this is some kind of threshold or cutoff.

The description of the following Table II is as follows. Figs. 9–13 show these results in a graph representation. Column *rank* means the ordering of similar proteins (e.g., Rank 1 means the most similar protein to a given protein, and Rank 10 means the tenth most similar protein to a given protein).

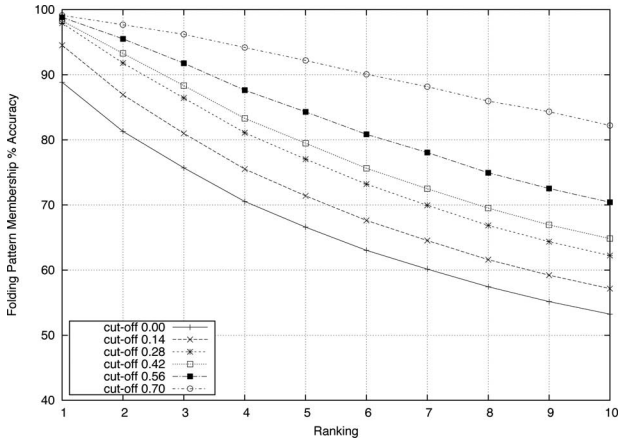

Fig. 10. Protein folding pattern membership percentage accuracy.

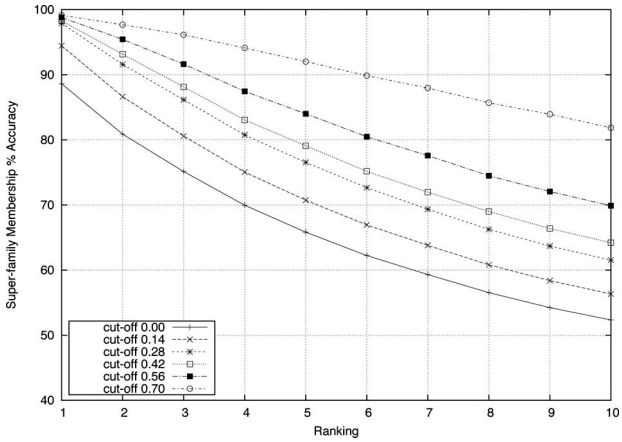

Fig. 11. Protein superfamily membership percentage accuracy.

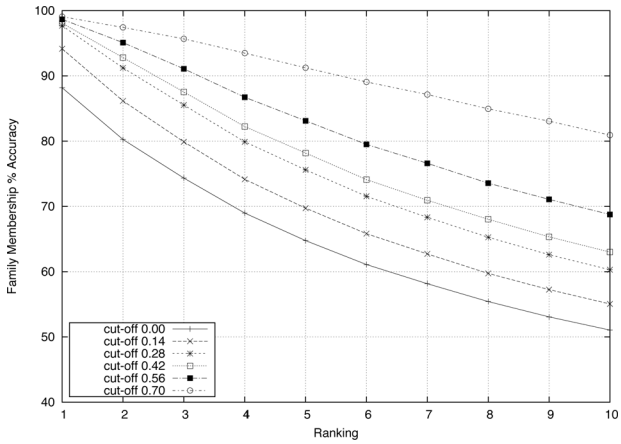

Fig. 12. Protein family membership percentage accuracy.

Column sim stands for minimal similarity (i.e., sim 0.0 means that there exists at least one protein to given protein, which has score of similarity at least 0.0). Line *count* means for how many proteins with this cutoff (minimal similarity) were found in our collection.

In more precise terms, e.g., line 1 of the Table II (not considering the header of the table) means that all the proteins placed

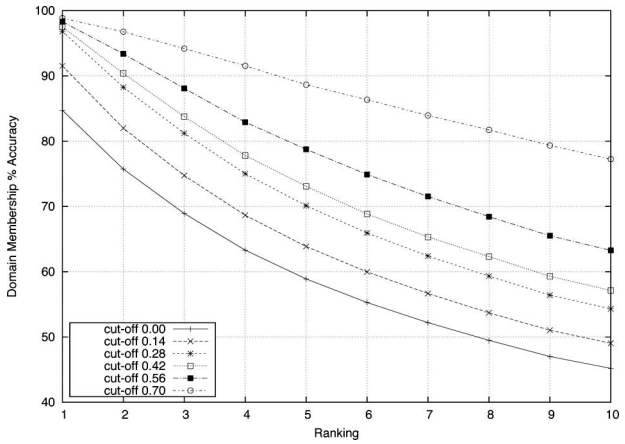

Fig. 13. Protein domain membership percentage accuracy.

TABLE III
PROTEINS UNCLASSIFIED BY USING SCOP FOUND BY OUR ALGORITHM AND THEIR MPA TO A GIVEN CLASS, FOLD, SUPERFAMILY, FAMILY, AND DOMAIN

sim	mpa_C	mpa_{Fo}	mpa_{SF}	mpa_{Fa}	mpa_D	**UPC**	**TPC**
0.00	93.02	88.83	88.61	88.17	84.72	4403	35908
0.14	96.44	94.52	94.44	94.14	91.51	3491	33747
0.28	98.36	97.91	97.89	97.67	96.80	2143	29526
0.42	98.64	98.28	98.28	98.07	97.51	1813	27163
0.56	99.04	98.81	98.80	98.66	98.30	1356	22374
0.70	99.19	99.12	99.12	99.06	98.85	639	12462

sim—Similarity score cutoff, mpa—membership percentage accuracy, UPC—count of proteins unclassified by SCOP, and TPC—total proteins count with given cutoff.

in the first place (i.e., the most similar protein to given a protein, see the Table I) have a 93.02% accuracy in the classification of class with no cutoff, a 96.44% accuracy with the cutoff of proteins scoring less than 0.14, etc.

We have also identified class, fold, superfamily, family, and domain of proteins, which are not classified by the SCOP with almost 100% membership accuracy. Table III shows these results. Let us examine line 5 of this table. Column sim $= 0.56$ means that we have chosen only proteins, which have at least one structurally similar protein with a score of similarity of at least 0.56. Column mpa_C means *membership percentage accuracy to the scop protein class* (same for fold—mpa_{Fo}, superfamily—mpa_{SF}, family—mpa_{Fa}, and domain—mpa_D). Column UPC—*unclassified proteins count* is the count of proteins, which are not classified by SCOP and which appear in the first place in the list of similar proteins to a given protein. Column TPC—*total proteins count* is the total count of proteins, which have at least one structurally similar protein with a score of similarity of at least 0.56. In summary, this means that we have found 1356 unclassified proteins by using SCOP out of 22 374, such that proteins have a 99.04% class membership accuracy, a 98.81% fold membership accuracy, a 98.80% superfamily membership accuracy, etc.

VI. CONCLUSION

In this paper, we have presented a novel method for measuring protein similarity using suffix tree data structure and IR techniques. The method is fully automated and in comparison with the human maintained database SCOP, our approach has achieved very good results. We have also demonstrated that we can use common IR models and methods for measuring similarity of proteins. With these methods, we have achieved very good results. The method can compute all pairwise similarities in about 12 h on one 1.8 GHz CPU.

We can also identify classes, folds, superfamilies, families, and domains of many unclassified proteins contained in the current SCOP database with almost 100% membership accuracy. By the simple observation that when the unclassified protein is most similar to the protein, which is classified and have at least some given score, then in 99% of the cases, the unclassified protein has a similar SCOP category as the known protein.

We now have a similarity matrix computed for all the proteins included in the current PDB database. In our future work, we wish to use the similarity matrix for other IR tasks, such as clustering, application of statistical, or graph methods. We will also investigate the protein relations based on these methods for using our algorithm.

REFERENCES

[1] RCSB Protein Databank—PDB, [Online]. Available: http://www.rcsb.org (last access July 10, 2009).
[2] J. Y. Yang, M. Q. Yang, A. K. Dunker, Y. Deng, and X. Huang, "Investigation of transmembrane proteins using a computational approach," in *Proc. BMC Genomics 2008*, vol. 9, (suppl 1):s7.
[3] J. Hecker, J. Y. Yang, and J. Cheng, "Protein disorder prediction at multiple levels of sensitivity and specificity," in *Proc. BMC Genomics 2008*, vol. 9, (suppl 1):s9.
[4] T. Habib, C. Zhang, J. Y. Yang, M. Q. Yang, and Y. Deng, "Supervised learning method for the prediction of subcellular localization of proteins using amino acid and amino acid pair composition," in *Proc. BMC Genomics 2008*, vol. 9, (suppl 1):s16.
[5] X. Wang, G. Wang, C. Shen, L. Li, X. Wang, S. D. Mooney, H. J. Edenberg, J. R. Sanford, and Y. Liu, "Using RNase sequence specificity to refine the identification of RNA-protein binding regions," in *Proc. BMC Genomics 2008*, vol. 9, (suppl 1):s17.
[6] F. Gao and M. J. Zaki, "PSIST: Indexing protein structures using suffix trees," in *Proc. IEEE Comput. Syst. Bioinf. Conf.*, 2005, pp. 212–222.
[7] T. Shibuya, "Geometric suffix tree: A new index structure for protein 3D structures," in *Proc. Combinatorial Pattern Matching, LNCS 4009*, 2006, pp. 84–93.
[8] O. Zamir and O. Etzioni, "Web document clustering: A feasibility demonstration," in *Proc. SIGIR1998*, pp. 46–54.
[9] K. M. Hammouda and M. S. Kamel, "Efficient phrase-based document indexing for web document clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 10, pp. 1279–1296, Oct. 2004.
[10] H. Chim and X. Deng, "A new suffix tree similarity measure for document clustering," in *Proc. 16th Int. Conf. World Wide Web 2007*. New York: ACM, pp. 121–130.
[11] J. Martinovič, T. Novosád, and V. Snášel, *Vector Model Improvement Using Suffix Trees,* Pictaway, NJ: IEEE ICDIM, 2007, pp. 180–187.
[12] M. Lexa, V. Snášel, and I. Zelinka, "Data-mining protein structure by clustering, segmentation and evolutionary algorithms," in *Data Mining: Theoretical Foundations and Applications*, Studies in Computational Intelligence, vol. 204. Berlin, Germany: Springer-Verlag, 2009, pp. 221-248, ISBN 978-3-642-01087-3.
[13] L. Holm and C. Sander, "Dali: A network tool for protein structure comparison," *Trends Biochem. Sci.*, vol 20, no. 11, pp. 478–480, 1995.
[14] A. M. Lesk, *Introduction to Bioinformatics*. New York: Oxford Univ. Press, 2008.
[15] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Reading, MA: Addison-Wesley, 1999.
[16] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.
[17] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval,* Cambridge, U.K.:Cambridge Univ. Press, Jan. 2008.
[18] C. J. van Rijsbergen, *Information Retrieval*, 2nd ed. London, U.K.: Butterworths, 1979.

[19] P. Weiner, "Linear pattern matching algorithms," in *Proc. 14th Annu. Symp. Found. Comput. Sci.*, 1973, pp. 1–11.
[20] A. Ehrenfeucht and D. Haussler, "A new distance metric on strings computable in linear time," *Discrete Appl. Math*, vol. 20, no. 3, pp. 191–203, 1988.
[21] M. Rodeh, V. R. Pratt, and S. Even, "Linear algorithm for data compression via string matching," *J. ACM*, vol. 28, no. 1, pp. 16–24, 1981.
[22] D. Gusfield, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge, U.K.: Cambridge Univ. Press, 1997.
[23] O. Zamir, "Clustering web documents: A phrase-based method for grouping search engine results," Doctoral dissertation. Univ. Washington, Seattle, Washington, DC, 1999.
[24] E. McCreight, "A space-economical suffix tree construction algorithm," *J. ACM*, vol. 23, pp. 262–272, 1976.
[25] E. Ukkonen, "On-line construction of suffix trees," *Algorithmica*, vol. 14, pp. 249–60, 1995.
[26] J. Ye, R. Janardan, and S. Liu, "Pairwise protein structure alignment based on an orientation-independent backbone representation," *J. Bioinformat. Comput. Biol.*, vol. 2, pp. 699–718, 2004.
[27] L. P. Chew, D. Huttenlocher, K. Kedem, and J. Kleinberg, "Fast detection of common geometric substructure in proteins," *J. Comput. Biol.*, vol. 6, no. 3/4, pp. 313–326, 1999.
[28] SCOP: A structural classification of proteins database for the investigation of sequences and structure. (1994–2009). [Online]. Available: http://scop.mrc-lmb.cam.ac.uk/scop/ (last access July 10, 2009).
[29] CATH: Protein structure classification. (1999–2008). [Online]. Available: http://www.cathdb.info/ (last access July 10, 2008).

Tomáš Novosád received the B.Sc. and M.Sc. degrees in computer science from Faculty of Natural Science, Department of Computer Science, Palacky University, Olomouc, Czech Republic. He is currently working toward the Ph.D. degree from the Faculty of Electrical Engineering and Computer Science, Vysoká Škola Báňská—Technical University of Ostrava, Ostrava, Czech Republic.

He is also an Active Senior Software Developer in a commercial company. His development experience includes more than 12 years in developing enterprise applications for large international and Czech companies. He has coauthored several publications on international conferences including best paper award. His current research interests include bioinformatics, machine learning, information retrieval, evolutionary computing, and data mining.

Václav Snášel received the M.Sc. degree from Palacky University, Olomouc, Czech Republic, and the Ph.D. degree in algebra and geometry from Masaryk University, Brno, Czech Republic.

In 2001, he was a Visiting Scientist at the Institute of Computer Science, Academy of Sciences, Czech Republic. Since 2003, he has been a Vice Dean for research and science at Faculty of Electrical Engineering and Computer Science, Vysoká Škola Báňská—Technical University of Ostrava, Ostrava, Czech Republic, where he has been a Dean since 2010 and a Full Professor since 2006. He was also with industrial company, where he was involved in different industrial research and development projects for more than 10 years. He has been engaged in research and development for more than 27 years in the industry and academia. He has given more than 12 plenary lectures and conference tutorials in various areas. He has authored/coauthored several refereed journal/conference papers, book chapters, and more than 350 papers (138 are recorded at Web of Science). He was also the Supervisor for many Ph.D. students from Czech Republic, Jordan, Yemen, Slovakia, Ukraine, and Vietnam. He is the Editor-in-Chief of two journals. He has also been engaged with the editorial board of some reputed international journals. His research interests include multidisciplinary environment involving artificial intelligence, multidimensional data indexing, bioinformatics, information retrieval, semantic Web, knowledge management, data compression, machine intelligence, neural network, Web intelligence, data mining and applied to various real-world problems.

Ajith Abraham (M'96–SM'07) received the M.S. degree from Nanyang Technological University, Singapore, and the Ph.D. degree in computer science from Monash University, Melbourne, Australia.

He is currently the Coordinator of the Machine Intelligence Research Labs, Scientific Network for Innovation and Research Excellence, Auburn, WA, which has members from more than 70 countries. He has a worldwide academic experience with formal appointments in Monash University, Oklahoma State University, Stillwater, OK, Chung-Ang University, Seoul, Korea, Jinan University, Jinan, China, Rovira i Virgili University, Tarragona, Spain, Dalian Maritime University, Dalian, China, Yonsei University, Seoul, Korea, the Open University of Catalonia, Barcelona, Spain, the National Institute of Applied Sciences, Lyon, France, and the Norwegian University of Science and Technology, Trondheim, Norway. He is engaged with the editorial board of several reputed international journals. He has been Guest Editor of more than 35 special issues on various topics. He has authored or coauthored more than 600 publications. He has given more than 40 plenary lectures and conference tutorials in various areas. His research and development experience includes more than 20 years in the industry and academia. His research interests include multidisciplinary environment involving machine intelligence, network security, sensor networks, e-commerce, Web intelligence, Web services, computational grids, data mining, and their applications to various real-world problems.

Dr. Abraham is a Senior Member of the IEEE Systems Man and Cybernetics Society, the IEEE Computer Society, the Institution of Engineering and Technology, U.K., the Institution of Engineers Australia, Australia, etc. He was the recipient of several best paper awards at international conferences. He is a Co-Chair of the IEEE Systems Man and Cybernetics Society Technical Committee on Soft Computing. He is actively involved in the Hybrid Intelligent Systems, Intelligent Systems Design and Applications, Information Assurance and Security, and Next Generation Web Services Practices series of international conferences, in addition to other conferences.

Jack Y. Yang received the M.S. and Ph.D. degrees from Purdue University, West Lafayette, IN, the postdoctoral training from Harvard Medical School, Boston, MA, and Indiana University School of Medicine, Indianapolis, IN, and training in biostatistics and bioinformatics from Johns Hopkins University, Baltimore, MD and in computer science from University of Illinois at Urbana-Champaign.

He is currently a Scientist with the Harvard University, Cambridge, MA. He has been a Faculty Member of Indiana University. He was an experimental and computer scientist for more than 15 years in teaching, research and engineering practice experience in biomedical engineering and computational science. He was engaged in both engineering practice and translational medicine. He is the Editor-in-Chief of *International Journal of Functional Informatics and Personalized Medicine* and an Honorary Consulting Editor of *International Journal of Computational Biology and Drug Design*. He has also been an Editor of more than a dozen journals and proceedings books, such as *International Journal of Pattern Recognition* and *Artificial Intelligence (World Scientific)*, *Journal of Supercomputing (Springer Science)*, Proceedings of IEEE Computer Society Bioinformatics and Biomedicine International Workshops, and Proceedings of International Conference on Bioinformatics and Computational Biology. He has authored or coauthored more than 100 peer reviewed papers and book chapters. He is also Specialist in cancer biology and artificial intelligence.

Dr. Yang was the General Chair of the IEEE 7th International Conference on Bioinformatics and Bioengineering at Harvard Medical School and Co-PI of the National Science Foundation grant. He is also a Consultant to International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing and International Conference on Bioinformatics and Computational Biology. He has also been involved in many invited talks including a number of keynote lectures to promote the emerging field of functional informatics and personalized medicine. He is the Chair of Board of Directors of International Society of Intelligent Biological Medicine.