BIOINFORMATICS

Vol. 20 no. 7 2004, pages 1045–1052
DOI: 10.1093/bioinformatics/bth036

Rapid 3D protein structure database searching using information retrieval techniques

Zeyar Aung* and Kian-Lee Tan

Department of Computer Science, National University of Singapore, 3 Science Drive 2, Singapore 117543

Received July 10, 2003; revised on November 20, 2003; accepted on November 24, 2003
Advance Access publication February 12, 2004

ABSTRACT

Motivation: As the sizes of three-dimensional (3D) protein structure databases are growing rapidly nowadays, exhaustive database searching, in which a 3D query structure is compared to each and every structure in the database, becomes inefficient. We propose a rapid 3D protein structure retrieval system named 'ProtDex2', in which we adopt the techniques used in information retrieval systems in order to perform rapid database searching without having access to every 3D structure in the database. The retrieval process is based on the inverted-file index constructed on the feature vectors of the relationships between the secondary structure elements (SSEs) of all the 3D protein structures in the database. ProtDex2 is a significant improvement, both in terms of speed and accuracy, upon its predecessor system, ProtDex.

Results: The experimental results show that ProtDex2 is very much faster than two well-known protein structure comparison methods, DALI and CE, yet not sacrificing on the accuracy of the comparison. When comparing with a similar SSE-based method, namely TopScan, ProtDex2 is much faster with comparable degree of accuracy.

Availability: The software is available at: http://xena1.ddns.comp.nus.edu.sg/~genesis/PD2.htm

Contact: zeyaraun@comp.nus.edu.sg

Supplementary information: Supplementary tables and figures for this paper can also be found at: http://xena1.ddns.comp.nus.edu.sg/~genesis/PD2.htm

INTRODUCTION

In the area of bioinformatics, three-dimensional (3D) structural comparison and structural database searching of proteins play important roles. In many cases, merely comparing the amino acid sequences of the proteins cannot provide sufficient information required by the biologist. In particular, we cannot detect the similarity of two remotely homologous proteins by sequence comparison alone. Instead, we need to compare their 3D structures in order to determine their similarity as the 3D structures are better preserved than the sequences throughout the evolution. Usually, a protein structure is compared against a database of other protein structures to find the structures that are similar to it.

Because of the advancements in the laboratory methods to determine the structures of the bio-molecules, the sizes of the protein structure databases such as PDB (Berman *et al.*, 2000) are growing at a very rapid rate. For example, PDB stored only about 1000 structures in 1993. But, today (as of November 2003) it stores over 23 000 structures. When the databases were of small size, in order to search a protein structure against a database, people could comfortably use exhaustive searching by comparing the query structure against each and every structure in the database. But, when the database sizes grow to the order of ten's of thousands, such an exhaustive searching approach cannot provide a satisfactory response time, however fast the pairwise comparison method used to compare the query against each of the database structures might be. Therefore, people have been starting to look at the filter-and-refine and the information retrieval (IR) approaches to cope with this database searching problem.

Our design strategy is to adopt the filter-and-refine and/or the IR approaches using secondary structure elements (SSEs) as the basic elements in order to speed up the process of database searching. We first build an inverted-file index based on the feature vectors of the relationships among the SSEs from all the protein structures in the database. When evaluating a query, we use this index to collectively determine the overall similarity of all the proteins in the database with respect to the query, and then retrieves and reports those that are most similar. We can optionally perform detailed pairwise comparison on the selected candidates upon user's request. We can use any of the existing structural alignment algorithms to carry out this refinement step.

PROBLEM DEFINITION

Pairwise 3D structure comparison

Given two 3D protein structures, we need to compare them according to a certain framework, and determine how similar they are. This process is also known as 3D structural matching or alignment. The general problem of structural comparison

*To whom correspondence should be addressed.

is known to be NP-complete. People use heuristics methods to solve this problem, and the similarity scores assigned to two given protein structures may be different from method to method.

3D structure database searching

Given a query protein structure, we have to search through the database and report the structures that are similar to the query structure. There may be a user-defined or pre-defined similarity threshold, and the structures whose similarity scores are equal to or above the threshold are reported. When using an exhaustive searching approach, searching through a database with N structures essentially means running pairwise comparison N times. If the database size becomes very large, such an exhaustive searching becomes extremely slow, because the response time grows linearly with the number of proteins in the database.

Filter-and-refine and IR approaches

Due to the inscalability of exhaustive searching approaches, researchers have started to look at more economical strategies such as filter-and-refine and IR to solve the problem of database searching.

In filter-and-refine strategy, the objects (3D protein structures in this case) that are most likely to be similar to the given query object are first selected or filtered by merely looking at an abstract structure, usually an index or a hash table, pre-constructed from the original collection of objects in the database. The objects that seem to be dissimilar to the query are discarded in the first place. Then, a detailed comparison method is applied to the selected objects in order to find those that are actually similar to the query.

In IR approach, the overall similarity measures of the objects in the database for a given query object are incrementally and simultaneously evaluated by looking at the abstract structure or an index pre-constructed from the original database. The original database of objects do not need to be processed at all.

Actually, these two strategies of filter-and-refine and IR are somehow interrelated and overlapping. Both strategies are designed to deal with huge collections of objects. ‘Indexing’ is the key concept in both of them. Both strategies may sacrifice accuracy in order to gain speed. They have to allow both false positives and false negatives to some extent during the index searching process. Both filter-and-refine and IR approaches have been widely used in the areas of document, image and spatial database searchings and retrievals (Bertino *et al.*, 1997). They have started to emerge in the area of 3D protein structure database searching recently.

RELATED WORKS

The classical pairwise comparison methods include SSAP (Orengo and Taylor, 1996), DALI (Holm and Sander, 1993), VAST (Gibrat *et al.*, 1996) and CE (Shindyalov and Bourne, 1998). These are the two-level methods, which start with finding the matching pairs of SSEs or C_α backbone fragments, and then go into the detailed finding of the matching C_α atom pairs. These methods can provide us with the good quality answers. But when performing a database search, they all have to use exhaustive searching, which results in slow response times.

TopScan (Martin, 2000) and SCALE (Chionh *et al.*, 2003) are examples of pairwise comparison methods which take SSEs as basic elements to be compared. These methods are less accurate, but much faster than the two-level methods. However, when searching against a very large database, these methods still cannot provide the required quick response time.

Guerra *et al.* (2002), PSI (Camoglu *et al.*, 2003) and ProtDex (Aung *et al.*, 2003), which is a predecessor of the system proposed in this paper, are the newly-emerged index-based methods designed for fast database searching. These methods take SSEs or relationships among them as basic elements.

INDEX CONSTRUCTION

In order to develop a rapid protein structure retrieval system for 3D structural database searching task, we adopt IR techniques, particularly inverted-file indexing and document similarity ranking mechanisms, which have been successfully used in the area of document/text retrieval for a long time. Our work is inspired by the methods such as CAFE (Williams and Zobel, 2002) that uses IR techniques to index and retrieve genome sequences, and VIPER (Müller *et al.*, 1999) that uses IR techniques in the content-based retrieval of images.

SSEs are the well-defined sub-structures within the protein structures. Two common types of SSEs are helix (H) and sheet (E). In our approach, we treat SSEs as the basic elements, as we can roughly determine the overall shape of a protein structure through the shapes and the topology of its SSEs. The number of SSEs in a protein is only an order of tens, while the number of amino acid residues is an order of hundreds. Thus, storing and handling SSEs as the basic structural elements is much more cost effective than handling the individual amino acid residues.

We perform the global similarity searching of a given query protein against the proteins in the database based on the feature vectors of the inter-SSE relationships (called the contact regions) that are indexed.

Our approach is both sequence and sequence-order independent. That is, we do not take any amino acid sequence information into account at all, and we also do not take the sequence-order information into account in finding the matching feature vectors.

The steps involved in constructing the index are described below. These steps are illustrated through Figures S1 through S4 in Supplementary information webpage.

Extracting feature vectors from contact regions

We represent a 3D protein structure as a set of feature vectors of the inter-SSE contact regions (hereafter referred to simply as contact regions). The two proteins are considered to be similar if they contain enough number of similar or compatible contact regions. So, it is important that the feature vector we choose capture sufficient information on the contact region so that we can effectively determine the similarity or compatibility between two given contact regions by comparing their representative feature vectors.

Our feature vector representation is derived from two classical concepts, namely distance matrix representation of the 3D protein structures, and vector representation of the SSEs.

Distance matrix representation. A 3D protein structure can be represented as a 2D distance matrix.

DEFINITION 1. (Distance matrix) *A distance matrix D representing a particular protein with n amino acid residues is an $n \times n$ matrix, in which each cell $D(i, j)$, where $1 \leq i, j \leq n$, stores the* Euclidean distance $Dist(i, j)$ *between the C_α atom of the ith residue and C_α atom of the jth residue. The distance matrix is symmetrical along the main diagonal, i.e. $D(i, j) \equiv D(j, i)$.*

The Euclidean distance between two points p and q is defined as

$$\mathrm{Dist}(p, q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2 + (z_p - z_q)^2} \quad (1)$$

where (x_p, y_p, z_p) and (x_q, y_q, z_q) are the 3D coordinates of point p and q, respectively.

The distance matrix is rotation and translation independent (Holm and Sander, 1993).

SSE vector representation. SSEs can be roughly approximated by their representative vectors or line segments in 3D space. We adopt the equations for calculating the start and the end points of an SSE vector from Singh and Brutlag (1997).

For a helix beginning at amino acid residue i and ending at amino acid residue j, the vector start and end points are calculated as

$$\begin{aligned}\mathbf{X}_{\text{start}} &= (0.74 \times \mathbf{X}_i + \mathbf{X}_{i+1} + \mathbf{X}_{i+2} + 0.74 \times \mathbf{X}_{i+3})/3.48\\ \mathbf{X}_{\text{end}} &= (0.74 \times \mathbf{X}_j + \mathbf{X}_{j-1} + \mathbf{X}_{j-2} + 0.74 \times \mathbf{X}_{j-3})/3.48\end{aligned} \quad (2)$$

where $\mathbf{X} = (x, y, z)$ is the coordinates of a point in 3D space.

For a sheet beginning at residue i and ending at residue j, the vector start and end points are calculated as:

$$\begin{aligned}\mathbf{X}_{\text{start}} &= (\mathbf{X}_i + \mathbf{X}_{i+1})/2\\ \mathbf{X}_{\text{end}} &= (\mathbf{X}_j + \mathbf{X}_{j+1})/2\end{aligned} \quad (3)$$

After having the start and the end points of a vector, we can represent it as a point vector $\mathbf{V}$ with respect to the origin $\mathbf{0}$.

$$\mathbf{V} = \mathbf{V}_{\text{end}} - \mathbf{V}_{\text{start}} \quad (4)$$

Given two point vectors $\mathbf{V}$ and $\mathbf{U}$, representing two SSEs, we can calculate the angle θ between them as:

$$\theta(\mathbf{V}, \mathbf{U}) = \begin{cases} 0 & \text{if } \mathbf{V} = \mathbf{U}\\ \cos^{-1}\left(\dfrac{\mathbf{V} \cdot \mathbf{U}}{\mathrm{Dist}(\mathbf{V}, \mathbf{0}) \cdot \mathrm{Dist}(\mathbf{U}, \mathbf{0})}\right) & \text{otherwise}\end{cases} \quad (5)$$

where $\mathbf{V} \cdot \mathbf{U}$ is the dot product of vectors $\mathbf{V}$ and $\mathbf{U}$, and Dist is the Euclidean distance function as defined in Equation (1).

Also, the vertex distance (VD) between two SSE vectors $\mathbf{V}$ and $\mathbf{U}$ can be calculated as:

$$VD(\mathbf{V}, \mathbf{U}) = \min \begin{cases} \mathrm{Dist}(\mathbf{V}_{\text{end}}, \mathbf{U}_{\text{start}})\\ \mathrm{Dist}(\mathbf{V}_{\text{start}}, \mathbf{U}_{\text{end}})\\ \mathrm{Dist}(\mathbf{V}_{\text{end}}, \mathbf{U}_{\text{end}})\\ \mathrm{Dist}(\mathbf{V}_{\text{start}}, \mathbf{U}_{\text{start}})\end{cases} \quad (6)$$

In our implementation, we use the STRIDE algorithm (Frishman and Argos, 1995) to identify the SSEs. We treat all α, π and 3/10 helices as a single type. For both helix and sheet, we assume the minimum length (number of amino acid residues) of an SSE to be 4. Any SSE with length shorter than 4, as annotated by STRIDE, is not regarded as an SSE. In the case of helices, if the length of a helix is exactly 4, it is extended by a single residue on either end in order to avoid a zero vector.

Feature vector construction. We have to construct a feature vector for each contact region in the distance matrix. A contact region can be formally defined as follows.

DEFINITION 2. (Contact region) *Within a distance matrix D (see Definition 1) of size $n \times n$, a contact region C^{ab} of two SSEs a and b is a sub-matrix containing the cells $D(s_a, s_b), \ldots, D(s_a + l_a - 1, s_b + l_b - 1)$, where s_a and s_b are starting amino acid residue numbers of SSEs a and b, respectively ($s_a, s_b \leq n$), and l_a and l_b are the lengths or the numbers of residues in SSEs a and b, respectively ($s_a + l_a - 1 \leq n, s_b + l_b - 1 \leq n$).*

Since the distance matrix is symmetrical, only the contact regions that are on or above the main diagonal are needed to be taken into account. For a protein with g SSEs, the total number of contact regions that need to be taken into account is $g \cdot (g + 1)/2$, which is $O(g^2)$.

For each contact region, we construct a 7D feature vector to represent it. The feature vector consists of the attributes that can effectively distinguish one contact region from another.

The feature vector of a contact region C^{ab}, which is formed by SSEs a and b, consists of seven attributes as shown below.

Dimension	Attribute	Function	Equation no.
1	Angle between a and b	θ	5
2	Vertex distance between a and b	VD	6
3	Square-root of the area of C^{ab}	SA	7
4	Aspect ratio of C^{ab}	AR	8
5	Mean C_α–C_α distance in C^{ab}	MD	9
6	Standard deviation of C_α–C_α distance in C^{ab}	SD	10
7	Type of C^{ab}	CT	11

The first two attributes are derived from SSE vector representation. Since a contact region essentially represents the inter-relationship between the two SSEs, we can logically associate it with the angle and the vertex distance between the two SSE vectors. Angle and distance are the natural and most commonly used properties for the relationship between two SSE vectors, and is also used in many other methods such as LOCK (Singh and Brutlag, 1997) and SCALE (Chionh *et al.*, 2003).

The rest of the attributes are derived directly from the distance matrix. Square-root of area and aspect ratio of contact region attributes are related to the lengths of the SSEs. These two attributes are used in order to avoid the matching of two contact regions that are formed by the SSE pairs with very different lengths. Mean and standard deviation of C_α–C_α distance attributes are related to the internal configuration of a contact region. They are used in order to avoid the matching of two contact regions which store very different C_α–C_α distance values. Contact region type is a natural property that is used for avoiding the comparison of two contact regions formed by different types of SSEs, respectively. The functions used to calculate these attribute values of the feature vector are defined as follows.

$$\mathrm{SA}(a,b) = \sqrt{l_a \cdot l_b} \tag{7}$$

$$\mathrm{AR}(a,b) = \min\left(\frac{l_a}{l_b}, \frac{l_b}{l_a}\right) \tag{8}$$

$$\mathrm{MD}(a,b) = \frac{\sum_{i=0}^{l_a-1}\sum_{j=0}^{l_b-1} D(s_a+i, s_b+j)}{l_a \cdot l_b} \tag{9}$$

$$\mathrm{SD}(a,b) = \sqrt{\frac{\sum_{i=0}^{l_a-1}\sum_{j=0}^{l_b-1} (D(s_a+i, s_b+j) - \mathrm{MD}(a,b))^2}{l_a \cdot l_b}} \tag{10}$$

$$\mathrm{CT}(a,b) = \begin{cases} 0 & \text{if } (a \text{ is H}) \wedge (b \text{ is H}) \wedge (a = b) \\ 1 & \text{if } (a \text{ is H}) \wedge (b \text{ is H}) \wedge (a \neq b) \\ 2 & \text{if } (a \text{ is E}) \wedge (b \text{ is E}) \wedge (a = b) \\ 3 & \text{if } (a \text{ is E}) \wedge (b \text{ is E}) \wedge (a \neq b) \\ 4 & \text{if } ((a \text{ is H}) \wedge (b \text{ is E})) \vee \\ & \quad ((a \text{ is E}) \wedge (b \text{ is H})) \end{cases} \tag{11}$$

where the definitions of all the symbols are the same as their previous definitions respectively. Now, let

$$\mathbf{K} = (k_\theta, k_{\mathrm{VD}}, k_{\mathrm{SA}}, k_{\mathrm{AR}}, k_{\mathrm{MD}}, k_{\mathrm{SD}}, k_{\mathrm{CT}})$$

be a feature vector. We can generate feature vector $\mathbf{K}^{ab}$ for contact region C^{ab} as follows.

$$\mathbf{K}^{ab} = (\theta(a,b), \mathrm{VD}(a,b), \mathrm{SA}(a,b), \mathrm{AR}(a,b), \mathrm{MD}(a,b), \mathrm{SD}(a,b), \mathrm{CT}(a,b)) \tag{12}$$

It should be noted that the feature vector we use is only a good approximation of the original contact region in an abstract form. There may be some cases that the two feature vectors are similar even though their original contact regions are not similar.

Building inverted-file index

For every protein structure in the database, we generate the 7D feature vectors as described above. After that, we hash these feature vectors into a hash table of seven dimensions, together with their Protein IDs. We finally build an inverted-file index based on the hash table.

Feature vector hashing. A hash table can be formally defined as follows.

DEFINITION 3. (n-dimensional hash table) *An n-dimensional hash table H is an n-dimensional array of size $(m_1+1) \times (m_2+1) \times \cdots \times (m_n+1)$ where (m_i+1) $(1 \le i \le n)$ is the length of each dimension. Each cell $H(d_1, d_2, \ldots, d_n)$ $(0 \le d_i \le m_i, 1 \le i \le n)$ in the array corresponds to a feature vector having exactly the 'discrete' attribute values of $(d_1, d_2, \ldots, d_n)$.*

We have to hash the original 7D contact region feature vector $\mathbf{K}$ with continuous attribute values into a 7D hash table with the hash function Hash. The idea is similar to that of hashing the points into the 3D grid cells in geometric hashing.

Let us define a quantized feature vector $\mathbf{T}$.

$$\mathbf{T} = (t_\theta, t_{\mathrm{VD}}, t_{\mathrm{SA}}, t_{\mathrm{AR}}, t_{\mathrm{MD}}, t_{\mathrm{SD}}, t_{\mathrm{CT}})$$

$\mathbf{T}$ can be calculated from $\mathbf{K}$ by the function Hash.

$$\mathbf{T} = \mathrm{Hash}(\mathbf{K}) \tag{13}$$

where Hash is a collection of partial quantization or discretization functions Hash_r on each continuous attribute value k_r

($r \in \{\theta, \text{VD}, \text{SA}, \text{AR}, \text{MD}, \text{SD}, \text{CT}\}$).

$$t_r = \text{Hash}_r(k_r) = \text{Round}\left(\frac{k_r \cdot m_r}{c_r}\right) \qquad (14)$$

where c_r and m_r are the maximum possible values of attribute r in the original continuous space and the new quantized space, respectively. In fact, the quantization function Hash_r performs a space-based or equal-size partitioning of the continuous data space of the attribute r.

The parameter values for c and m for each of the attributes in our implementation are (180.0, 100.0, 80.0, 1.0, 100.0, 50.0, 4) and (12, 10, 10, 10, 10, 10, 4) respectively. As a result, we have a hash table of size $(13 \times 11 \times 11 \times 11 \times 11 \times 11 \times 5)$.

Inverted-file index. The idea of inverted-file indexing is borrowed from the area of text and document retrieval. An inverted file is basically a list of ‘words’, each pointing to a posting list of ‘documents’ in which it occurs. In our case, we can treat the quantized feature vectors as our words, and the proteins in which they occur as our documents.

In our implementation, each cell in the hash table H stores a pointer to a posting list consisting of Protein IDs together with their occurrence counts. After we have hashed an original feature vector $\mathbf{K}$ into a quantized feature vector $\mathbf{T} = \text{Hash}(\mathbf{K})$, we update the posting list pointed by the cell $H(\mathbf{T})$. We insert the Protein ID, in which $\mathbf{K}$ occurs, into the posting list if it does not exist in the list yet. Otherwise, its occurrence count is increased.

After processing all the contact region feature vectors from all the proteins in the database in this way, we finally come up with our inverted-file index, in which each cell in the hash table points to the posting list of Protein IDs and their number of occurrences. Obviously, some of the cells in the hash table may have empty pointers.

QUERY EVALUATION AND DATABASE RETRIEVAL

In order to evaluate the similarity score between a query protein structure a protein structure in the database, we adopt and modify the well-known $\Sigma(tf \times idf)$ scoring scheme commonly used in document retrieval systems. Given a query protein structure Q and a protein structure P in the database, their overall similarity score ψ can be calculated as:

$$\psi(Q, P) = \frac{\sum_{\phi(\mathbf{T} \in Q, \mathbf{T}' \in P) \neq 0} [w(Q, \mathbf{T}) \cdot w(P, \mathbf{T}') \cdot \phi(\mathbf{T}, \mathbf{T}')]}{W_Q \cdot W_P} \qquad (15)$$

Given two quantized feature vectors $\mathbf{T}$ and $\mathbf{T}'$, we can determine their matching or compatibility score ϕ as

$$\phi(\mathbf{T}, \mathbf{T}') = \prod_{r \in \{\theta, \text{VD}, \text{SA}, \text{AR}, \text{MD}, \text{SD}, \text{CT}\}} \phi_r(t_r, t'_r) \qquad (16)$$

where ϕ_r is the partial matching score for attribute r ($r \in \{\theta, \text{VD}, \text{SA}, \text{AR}, \text{MD}, \text{SD}, \text{CT}\}$), which is in turn defined as

$$\phi_r(t_r, t'_r) = \begin{cases} \sigma_r \cdot e^{-(|t_r - t'_r|/\xi_r)} & \text{if } |t_r - t'_r| \leq \xi_r \\ 0 & \text{otherwise} \end{cases} \qquad (17)$$

where ξ_r is the threshold value for the allowable difference between two attribute values t_r and t'_r, and σ_r is the relative importance of attribute r. In our current implementation, we set $\xi_r = 1$ for $r \in \{\theta, \text{VD}, \text{SA}, \text{AR}, \text{MD}, \text{SD}\}$ and $\xi_{CT} = 0$, and $\sigma_r = 1$ for all $r \in \{\theta, \text{VD}, \text{SA}, \text{AR}, \text{MD}, \text{SD}, \text{CT}\}$.

$w(Q, \mathbf{T})$ is the weight of quantized feature vector $\mathbf{T}$ from query Q, which is calculated as:

$$w(Q, \mathbf{T}) = (\lg f_{Q,\mathbf{T}} + 1) \cdot \left(\lg \frac{N}{f_\mathbf{T}} + 1\right) \qquad (18)$$

and $w(P, \mathbf{T}')$ is the weight of quantized feature vector $\mathbf{T}'$ from database protein P, which is calculated as:

$$w(P, \mathbf{T}') = (\lg f_{P,\mathbf{T}'} + 1) \qquad (19)$$

where N is the total number of protein structures in the database, $f_\mathbf{T}$ is the number of proteins in which $\mathbf{T}$ occurs, $f_{Q,\mathbf{T}}$ is the number of occurrences of $\mathbf{T}$ in Q, and $f_{P,\mathbf{T}'}$ is the number of occurrences of $\mathbf{T}'$ in P.

W_x is the size of protein $x \in \{Q, P\}$ in terms of the number of quantized feature vectors it contains.

$$W_x = \sqrt{\sum_{\mathbf{T} \in x} (w(x, \mathbf{T}))^2} \qquad (20)$$

All the information required to calculate similarity score ψ can be easily extracted from the inverted-file index. We use a modified version of a textbook algorithm (Bertino *et al.*, 1997, p. 171) to calculate the similarity scores of all the proteins in the database, with respect to a query, by using the inverted-file index. The scores are then normalized into the range of 0–100. After that, all the database proteins are ranked according to their similarity scores, and are reported to the user.

Calculation of the similarity scores for all the proteins is done ‘simultaneously’ and ‘incrementally’ during the process of searching through the index. The scheme is scalable, because the index structure we need to search through is only a fixed-size hash table which will not grow with the growth of the database itself. However, the lengths of the posting lists will apparently increase in sizes with the growth of the database. There will still be some increases in cost for handling them when the database grows.

EXPERIMENTAL RESULTS

We compare our new ProtDex2 scheme against the widely-used detailed comparison schemes, namely DALI (Holm and Sander, 1993) and CE (Shindyalov and Bourne, 1998); a similar SSE-based scheme, namely TopScan (Martin, 2000);

and the current scheme's predecessor, namely ProtDex (Aung *et al.*, 2003). For DALI, CE and TopScan, we use their downloaded stand-alone versions rather than the web services.

All the experiments are conducted on Pentium 4 desktop computer with 1.6 GHz CPU and 256 MB main memory. The databases we use in our experiments are the subsets of ASTRAL v1.59 (Brenner *et al.*, 2000). Strictly speaking, the 3D structures stored in ASTRAL are not the whole proteins, but the domains within the proteins according to SCOP (Hubbard *et al.*, 1997) domain definitions. (However, we will hereafter refer to these domains as proteins for simplicity.)

We conducted two experiments: one involving a small database and a limited number of queries, and the other involving a large database and a greater number of queries.

Experiment on small database

We randomly select 10 proteins from Globins family (a.1.1.2 in SCOP) and 10 proteins from Serine/ Threonin kinases family (d.144.1.1 in SCOP) from the representative ASTRAL dataset with less than 40% sequence homology. These 20 proteins are designated as the query proteins. (Table S1 in supplementary webpage.)

We again randomly select 180 proteins, other than Globins and Serine/Threonin kinases, from four major classes (All-α, All-β, α/β and $\alpha + \beta$) of the same representative dataset. We combine these 180 proteins with the above-mentioned 20 query proteins to form the target database of 200 proteins. (Table S2 in supplementary webpage.)

We run 20 queries—taken from the Globins and Serine/ Threonin kinases families—against the target database. For DALI and CE, the similarity scores of each query protein to all the database proteins are calculated using pairwise comparisons. (Although DALI has a specialized database searching facility, it is not flexible enough to be used in our experiment.) For TopScan, the symbolic topology strings for the database proteins are pre-constructed. For each query, the similarity scores are calculated by comparing the query's topology strings to all of the database's topology strings. A comprehensive comparison mode is used taking into account the information on neighbors, accessibility, element length and loop length. For both ProtDex and ProtDex2, the indexes are pre-constructed from the database. The similarity scores of each query protein to the database proteins are calculated with the help of the index.

In all methods, for each query, all the proteins in the database are ranked according to their similarity scores with respect to the query, and are retrieved in this ranking order. If a retrieved protein and the query protein belong to the same 'family', which is the most detailed level in SCOP classification, it is regarded as a 'relevant' retrieval. For example, for a Globins family query protein, if a retrieved protein also belongs to Globins family, it is regarded as a relevant retrieval. For each query, there are 10 relevant proteins in the database of 200 proteins. If retrieved randomly, the probability of selecting a relevant protein is only 0.05.

The speed comparison of the selected methods for this experiment is shown in Table 1. The accuracy comparison is shown in Table 2.

Table 1. Running times for 20 queries on the database of 200 proteins

Method	Total time (hh : m m: ss)	Average time per query (hh : mm : ss.mm)	Average time per comparison (hh : mm : ss.mmmm)
DALI	52:36:08	02:37:48.40	00:00:47.3420
CE	10:23:03	00:31:09.15	00:00:09.3458
TopScan	00:00:59	00:00:02.95	00:00:00.0148
ProtDex	00:00:43	00:00:02.15	00:00:00.0108
ProtDex2	00:00:16	00:00:00.80	00:00:00.0040

Table 2. Accuracy comparison for 20 queries (10 form Globins family and 10 from Serine/ Threonin kinases family) on the database of 200 proteins.

No. of relevant retrievals	Average no. of retrievals required DALI	CE	TopScan	ProtDex	ProtDex2
1	1	1	1	1	1
2	2	2	2	3	2
4	4	4	5	7	4
6	6	6	8	12	6
8	8	8	14	21	9
10	10	10	29	79	16

In the table, row *i* represents the ranking under the various methods to retrieve *i* relevant answers. For example, row 2 says that when 2 answers are required, the top 2 ranked answers from DALI, CE, TopScan and ProtDex2 are the 2 relevant answers from the same family as the query; while ProtDex ranks the 3 relevant answers among the top 3 retrievals

Experiment on large database

We conduct another experiment using a large database containing 34 055 proteins which cover about 90% of the entire ASTRAL database. From them, we select 108 query proteins which belongs to 108 medium-size families (with $\geq$40 and $\leq$180 members) from four major classes, and which have less than 40% sequence homology to each other.

The lists of 34 055 database proteins and 108 query proteins are given respectively in Tables S3 and S4 in Supplementary information webpage. DALI and CE are excluded from this experiment because it is impractical to run them given their very high computational costs. It can be estimated that DALI will take over 5 years, and CE will take over 1 year, respectively, to run this experiment on the given machine! Only TopScan, ProtDex and ProtDex2 are included in this experiment.

It should be noted that the sizes of the families (40–180) are quite small with respect to the size of the entire database

Table 3. Running times for 108 queries on the database of 34 055 proteins

Method	Total time (hh : mm : ss)	Average time per query (hh : mm : ss.mm)	Average time per comparison (hh : mm : ss.mmmm)
TopScan	26:15:51	00:14:35.47	00:00:00.0257
ProtDex	05:44:35	00:03:11.46	00:00:00.0056
ProtDex2	00:14:03	00:00:07.81	00:00:00.0002

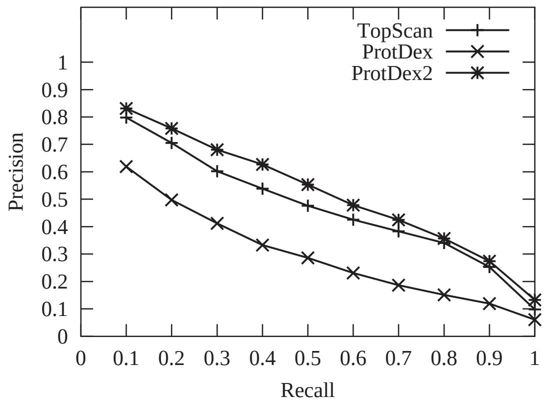

Fig. 1. Average precision–recall curves for 108 queries on the database of 34 055 proteins.

(34 055) and the probability of selecting a relevant protein by chance is quite low (0.0012–0.0053).

The speed comparison of the selected methods for this database searching task is shown in Table 3.

The accuracy comparison is shown in Figure 1. Again, a relevant retrieval is defined as an event of retrieving a protein from the database that belongs to the same ‘family’ as the query. The results are shown as average precision-recall curves, which are commonly used in the IR experiments. Precision and recall can be defined as:

$$\text{Precision} = \frac{\text{Number of relevant retrievals}}{\text{Total number of proteins retrieved}} \quad (21)$$

$$\text{Recall} = \frac{\text{Number of relevant retrievals}}{\text{Total number of proteins in the same family}} \quad (22)$$

DISCUSSION

Speed comparison

The fast speed of ProtDex2 is attributed to the conciseness of the contact region feature vector representation scheme, and the query evaluation scheme that uses the inverted-file index to collectively rank the database proteins simultaneously. The cost incurred on each virtual pairwise comparison decreases significantly (from 4 ms to 0.2 ms) as the size of the database grows (from 200 to 34 055 proteins).

Although its predecessor method, ProtDex, also uses inverted-file based query evaluation, the feature vectors are based on fixed-size overlapping sliding windows. Thus the number of feature vectors per protein is much more than that in ProtDex2, and the query evaluation is relatively slower as it involves comparisons of a huge number of feature vector pairs.

DALI and CE are apparently much slower than ProtDex2 as they are detailed two-level pairwise comparison schemes.

TopScan is much faster than DALI and CE, but still slower than ProtDex2. It has to perform exhaustive searching of each query against the whole database. The disadvantage of this exhaustive searching scheme is magnified when the database size grows. TopScan is only about 3.5 times slower than ProtDex2 for the small database of 200 proteins, but about 112 times slower for the large database with 34 055 proteins. In addition, TopScan requires 24 rotations of one structure for each pairwise comparison. Since ProtDex2 is based on SSE contact regions of the distance matrix, such rotations are not required.

Accuracy comparison

As shown in Table 2, in order to obtain all the relevant answers, ProtDex2 has to retrieve more proteins than the detailed comparison methods of DALI and CE. In this experiment, ProtDex2 needs to retrieve the top 16 answers on the average, whereas DALI and CE need to retrieve only the top 10 answers, in order to obtain all of the 10 relevant answers. However, we can achieve the same level of accuracy as DALI and CE by retrieving these top 16 answers, which is only 8% of the entrie database in this case, and refining them with DALI or CE. Given the very fast speed of ProtDex2, this filter-and-refine strategy can reduce the running time by about 12 folds while maintaining the good accuracy of the detailed comparison methods.

ProtDex2 is more accurate than its predecessor ProtDex method. In ProtDex method, the feature vectors are extracted from the fixed-size sliding windows sub-divided from the contact regions. This approach leads to the poorer results due to the cross-matchings of the sliding windows from the different contact regions. This weaknesses is avoided in ProtDex2 method by using only the feature vectors of the contact regions in their entirety.

The accuracy of ProtDex2 is comparable to that of TopScan. Both methods are based on SSEs. TopScan uses symbolic linear representation of SSE vectors using the various properties such as SSE type, direction, length, proximity, etc. On the other hand, ProtDex2 uses feature vector representation of 2D SSE contact regions using their various properties.

Interpreting similarity scores

For each query, ProtDex2 assigns a similarity score between 0 and 100 to every database protein. For the experiment

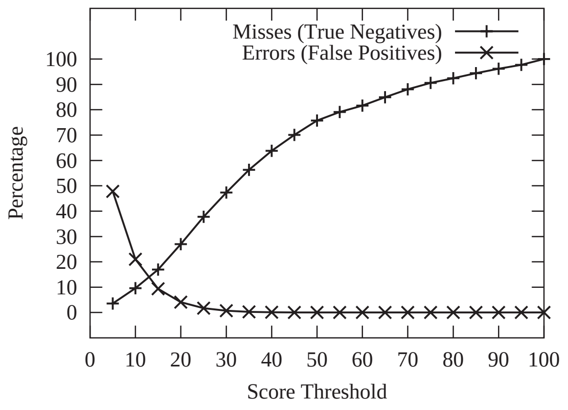

Fig. 2. Errors and Misses percentages for various score thresholds.

of 108 queries on 34 055 proteins, we conduct a frequency analysis of the scores of the relevant retrievals (intra-family matches) and those of the irrelevant retrievals (inter-family mismatches). (Table S5 in supplementary webpage.) Then, we calculate the average percentage of errors and misses for each score checkpoint yielding Figure 2. It can be observed that if we set the similarity score threshold as 15, we can have an optimal result with about 10% errors and 17% misses. If we set the score threshold as 30, we can achieve 99% accuracy (1% misses) with 47% error rate.

CONCLUSION

In this paper, we have proposed a new SSE-based indexing scheme for efficient retrieval of the protein structures from the large databases. We conducted an experiment on the retrieval efficiency and effectiveness of the scheme in comparison with the other methods by using a small database and some query proteins from the well-known Globins and Serine/Threonin kinases families. We also conducted another experiment using a larger database and the several query proteins form diverse families in order to observe the more general behaviour of the scheme.

The experimental results showed that ProtDex2 is very much faster than two well-known protein structure comparison methods, DALI and CE, yet not sacrificing on the accuracy of the comparison. When comparing with a similar SSE-based method, TopScan, ProtDex2 is much faster with comparable degree of accuracy. In filter-and-refine framework, it can be ideally used as a filtering tool to reduce the search space before running a more detailed but slower structural comparison method.

As a future work, we can further improve the accuracy of the scheme by using data or distribution-based partitioning in quantizing the feature vectors, and by using different relative importance values for different attributes in the feature vector, etc.

Finally, it can become a very useful scheme in the near future when the protein structure database sizes become too large to be searched through exhaustively.

REFERENCES

Aung,Z., Fu,W. and Tan,K.L. (2003) An efficient index-based protein structure database searching method. In *Proceedings of 8th International Conference on Database System for Advanced Applications (DASFAA'03)*, pp. 311–318.

Berman,H.M., Westbrook,J., Feng,Z., Gilliland,G., Bhat,T., Weissig,H., Shindyalov,I. and Bourne,P. (2000) The protein data bank. *Nucleic Acids Res.*, **28**, 235–242.

Bertino,E., Ooi,B.C., Sacks-Davis,R., Tan,K.L., Zobel,J., Shidlovsky,B. and Catania,B. (1997) *Indexing Techniques for Advanced Database Systems*. Kluwer Academic Publishers.

Brenner,S.E., Koehl,P. and Levitt,M. (2000) The astral compendium for sequence and structure analysis. *Nucleic Acids Res.*, **28**, 254–256.

Camoglu,O., Kahveci,T. and Singh,A.K. (2003) Psi: indexing protein structures for fast similarity search. *Bioinformatics*, **19**, 81i–83i.

Chionh,C.H., Huang,Z., Tan,K.L. and Yao,Z. (2003) Augmenting sses with structural properties for rapid protein structure comparison. In *Proceedings of 3rd IEEE Symposium on Bioinformatics and Bioengineering (BIBE'03)*, pp. 341–350.

Frishman,D. and Argos,P. (1995) Knowledge-based secondary structure assignment. *Prot. Struct. Funct. Genet.*, **23**, 566–579.

Gibrat,J.F., Madej,T. and Bryant,H. (1996) Surprising similarities in structure comparison. *Curr. Opin. Struct. Biol.*, **6**, 377–385.

Guerra,C., Lonardi,S. and Zanotti,G. (2002) Analysis of secondary structure elements of proteins using indexing techniques. In *Proceedings of 1st International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'02)*, pp. 812–823.

Holm,L. and Sander,C. (1993) Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, **233**, 123–138.

Hubbard,T.J.P., Ailey,B., Brenner,S.E., Murzin,A.G. and Chothia,C. (1997) Scop: a structural classification of proteins database. *Nucleic Acids Res.*, **25**, 236–239.

Martin,A. (2000) The ups and downs of protein topology: rapid comparison of protein structure. *Prot. Engg.*, **13**, 829–837.

Müller,H., Squire,D.M., Müller,W. and Pun,T. (1999) Efficient access methods for content-based image retrieval with inverted files. In *Proceedings of Multimedia Storage and Archiving Systems IV (VV'02)*.

Orengo,C.A. and Taylor,W.R. (1996) Ssap: sequential structure alignment program for protein structure comparison. In *Methods in Enzymology*. Academic Press, Vol. 266, pp. 617–635.

Shindyalov,I.N. and Bourne,P.E. (1998) Protein structure alignment by incremental combinatorial extension (ce) of the optimal path. *Prot. Engg.*, **11**, 739–747.

Singh,A.P. and Brutlag,D.L. (1997) Hierarchical protein structure superposition using both secondary structure and atomic representations. In *Proceedings of 5th International Conference on Intelligent Systems for Molecular Biology (ISMB'97)*, pp. 284–293.

Williams,H.E. and Zobel,J. (2002) Indexing and retrieval for genomic databases. *IEEE Trans. Knowl. Data Engg.*, **14**, 63–78.