International Journal on Artificial Intelligence Tools
© World Scientific Publishing Company

Towards SCALEable Protein Structure Comparison and Database Search

Chern-Hooi Chionh Zhiyong Huang Kian-Lee Tan Zhen Yao
Dept Computer Science, National University of Singapore
3 Science Drive 2, Singapore 117543
{jchionh,huangzy,tankl,yaozhen}@comp.nus.edu.sg

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Comparing protein structures in three dimensions is a computationally expensive process that makes a full scan of a protein against a library of known protein structures impractical. To reduce the cost, we can use an approximation of the three dimensional structure that allows protein comparison to be performed quickly to filter away dissimilar proteins. In this paper, we present a new algorithm, called SCALE, for protein structure comparison. In SCALE, a protein is represented as a sequence of secondary structure elements (SSEs) augmented with 3D structural properties such as the distances and angles between the SSEs. As such, the comparison between two proteins is reduced to a sequence alignment problem between their corresponding sequences of SSEs. The 3-D structural properties of the proteins contribute to the similarity score between the two sequences. We have implemented SCALE, and compared its performance against existing schemes. Our performance study shows that SCALE outperforms existing methods in terms of both efficiency and effectiveness (measured in terms of precision and recall). To avoid exhaustive search, an index based on the structural properties is also proposed. The index prunes away a considerable amount of dissimilar proteins given a query protein.

Keywords: Protein structure, similarity search, secondary structure elements, sequence alignment, structural properties.

1. Introduction

Protein structure comparison is a very important tool that has wide applications in protein structure analysis, structure-based drug design, molecular modelling and knowledge-based protein structure predictions. Whenever a new protein structure is solved, it is often necessary to determine whether the structure has been seen before. This is typically done by searching a library of known protein structures. However, automatic 3D structure comparison at the atomic coordinate level[23,25,30] is so time-consuming for it to be of practical use. Even with the use of sequence screening and a hierarchical expansion scheme to reduce the search space, it is estimated that approximately two weeks of computer time would be required to scan a large protein domain such as a TIM barrel[3,4,6]. The problem is exacerbated by the rapid increase in the sizes of databases for 3D protein structures as new structure data are increasingly being discovered. Thus, novel methods for structural bioinformatics are required. In particular, approximate techniques that provide rapid protein structure comparison are desirable as they can serve as a pre-screen for the computationally expensive SSAP-type algorithms.

One promising approach is to represent the 3D structure of protein by its secondary structure elements (SSEs). However, SSE alone is not adequate in distinguishing (dis)similar protein since two proteins with the same SSE can be folded in different ways. Instead, most of the existing works augment the SSE with additional information[3,9,17,22,24,32]. For instance, TOPSCAN[3] uses symbolic linear representation of SSE vectors augmented with properties such as SSE type, direction, length, proximity, etc, and applies dynamic programming to align two linear representations. Another recently proposed method[32] represents the various relationships, such as distances, angles, types and lengths, among the SSEs in a protein as matrices. Then it finds the maximum common parts of the corresponding matrices of two proteins to detect their similarity. Unfortunately, TOPSCAN is not rotation invariant and requires the same comparison to be performed 24 times between two sequences of SSEs. As such, it is not expected to scale for large protein databases. The method in Ref. 32 is also computationally expensive in determining common submatrices between proteins. To keep the processing overhead low, the scheme restricts its search space to a single connected SSE at the expense of the sensitivity of the method.

In this paper, we present a new algorithm, called SCALE[a] (Structure-Conscious ALignment of secondary structure Elements), for protein structure comparison. In SCALE, a protein is represented as a sequence of secondary structure elements (SSEs) augmented with 3D structural properties. As a first cut, we have used the distances of closest approach and the dihedral packing angles between the SSEs as the 3D structural properties. The similarity between two proteins can be determined based on the SSE alignment and the 3D structural properties. As such, the comparison between two proteins is reduced to a sequence alignment problem between their corresponding sequences of SSEs. The 3-D structural properties of the proteins contribute to the similarity score between the two sequences.

We have implemented SCALE, and compared its performance against TOPSCAN[3] and BLAST[23,25] (using primary sequence alignment) on real datasets obtained from the Protein Data Bank[29]. Two sets of experiments are performed. The first evaluates the effectiveness of SCALE on a small dataset of 90 proteins, and the second evaluates the efficiency of the proposed algorithm on a large dataset of 1000 proteins. Our performance study shows that BLAST is not effective to detect proteins with similar structure if their primary sequences are dissimilar. Our study also shows that the proposed method is not only efficient compared to TOPSCAN, but is also more effective (in terms of minimizing false drops and maximizing the number of true hits).

[a]A SCALE is a measuring instrument - in our case, a measure of similiarity between proteins.

However, even with a rapid comparison algorithm, exhaustive database search is still computationally expensive. Thus, we also develop an index scheme based on protein SSE structural properties. Experiments show that this index scheme is able to prune away considerable number of dissimilar structures in the database so that only a small number of proteins which are presumed to be similar to the query need to be examined by the comparison algorithm.

A preliminary version of this paper appears in Ref. 14. There, we presented the proposed structure comparison scheme, SCALE, and reported some preliminary results. In this version, we have extended the work to facilitate database searching more efficiently using a newly designed index structure.

The rest of this paper is organized as follows. In the next section, we present our scheme to represent a protein. Section 3 discusses the proposed protein structure comparison algorithm. We introduce the index structure for database searching in Section 4. In Section 5, we present an experimental study and report our findings. Section 6 reviews some existing work on protein structure comparison, and finally, we conclude in Section 7 with directions for future work.

2. Representation of a protein

Since the 3D structure of a protein is more conserved, we expect two similar proteins to share common substructure(s). As such, in this work, each protein is represented by its sequence of secondary structure elements (SSEs) and an *angle-distance (AD)* matrix. There are two major types of SSEs, namely, α helix and β strand. A protein SSE sequence is thus a string of α and β. For example, the protein `1cuk`[b] has 14 SSEs β-β-β-α-β-β-α-α-α-α-α-α-α-α.

The AD matrix captures the 3D information between the SSEs of a protein. It is a $n \times n$ matrix that represents all pair-wise closest distances and dihedral angles between the SSEs, where n is the number of SSEs the protein has. The upper triangle part of the matrix stores the *distance of closest approach* between two SSEs, while the lower triangle part contains the *dihedral packing angle* of two SSEs. Figure 1 illustrates how the closest distance (d) and the dihedral angle (Ω) are determined, as described in Ref. 12.

Mathematically, AD is defined as:

$$AD_{i,j} = \begin{cases} d_{i,j} & \text{if } i < j, \\ 0 & \text{if } i = j, \\ \Omega_{i,j} & \text{if } i > j. \end{cases} \tag{1}$$

where, 1$\leq$ i,j $\leq$n, $d_{i,j}$ and $\Omega_{i,j}$ are the distance of closest approach and the dihedral packing angle between the i-th and j-th SSEs in the protein, respectively. Figure 2 shows the AD matrices for several proteins. In the figure, the darker a cell is, the smaller is the real number in that cell.

[b]Throughout this paper, we shall use the PDB code when naming protein.

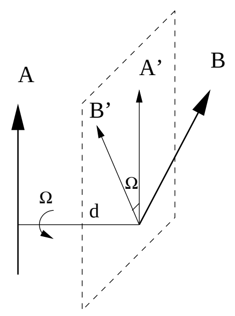
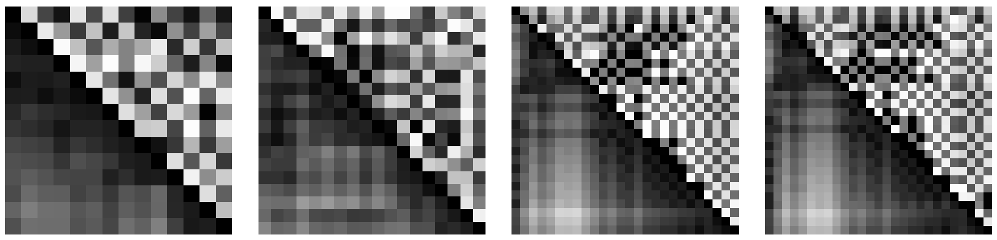


Fig. 1. Computation of the closest distance and dihedral angle between two SSEs. Each SSE is represented by a vector with length and direction obtained from the N- and C-terminal C_α atoms. Let A and B be two vectors corresponding to two SSEs, d is the closest distance between A and B. Moreover, let A' and B' be the projected vectors onto the plane that is normal to d. The dihedral angle (Ω) is the angle between A' and B' measured along the plane.



(a) 1cuk. (b) 1a5t. (c) 1kap. (d) 1sat.

Fig. 2. Examples of angle-distance matrix AD.

3. Protein Structure Comparison

In this section, we look at how to measure the similarity between two proteins. We first discuss the similarity measure, and then present an algorithm to compute the similarity measure between two proteins.

3.1. *Similarity Measure*

Before we look at the similarity measure between two proteins, let's refer to Figure 2 again. From the figure, it is clear that `1cuk` is more similar to `1a5t` than `1kap` and `1sat`. Likewise, `1kap` has a higher degree of similarity to `1sat` than `1cuk` and `1a5t`. This is not surprising as, according to CATH[5,19], `1cuk` and `1a5t` belong to the same class "main alpha" and architecture "orthogonal bundle". Similarly, proteins `1kap` and `1sat` both belong to class "main beta" and architecture "2 solenoid". More interestingly, we also note that the matrices from different categories (or families) are quite dissimilar (compare Figure 2(a) and (c)).

With our representation of proteins, two proteins, p_1 and p_2, are similar if

- p_1 and p_2 share common SSE subsequences.
- The pairwise distances between the common SSE subsequences are similar.
- the pairwise dihedral angles between the common SSE subsequences are similar.

Our basic approach is to align the SSE sequences and to use the information in the AD matrices in computing the degree of similarity based on the alignment. We shall defer the discussion on the alignment algorithm to the next subsection. Suppose p_1 and p_2 are two proteins with SSE lengths l_1 and l_2 respectively. We shall denote the SSE sequences of the two proteins as $p_1[1, l_1]$ and $p_2[1, l_2]$ respectively. Suppose p_1 and p_2 have been aligned, and the length of the final alignment is L ($L > l_1$, $L > l_2$). Let $p_1^a[1, L]$ and $p_2^a[1, L]$ denote the sequences of p_1 and p_2 in the final alignment respectively. Two subsequences in the alignment, $p_1^a[i, j]$ and $p_2^a[i, j]$ $(i < j)$, are said to be *common* if

$$\forall\ k \quad p_1^a[k] = p_2^a[k] \quad (i \le k \le j). \tag{2}$$

The subsequences are said to be *maximally common* if Eq.(2) holds and there does not exist $i' < i$ and $j' > j$ for which the equation holds.

Let $p_1^a[i, j]$ and $p_2^a[i, j]$ $(i < j)$ be a maximally common subsequence (MCS). Moreover, let $p_1[i_1, i_1 + j - i + 1]$ and $p_2[i_2, i_2 + j - i + 1]$ be the corresponding subsequences in proteins p_1 and p_2, i.e., $p_1[k_1] = p_2[k_2]$ for $i_1 \le k_1 \le i_1 + j - i + 1,\ i_2 \le k_2 \le i_2 + j - i + 1$. Let T_d be a threshold value used to determine the tolerance for the absolute difference between the distances of a matching pair of SSE is acceptable. T_a is defined for angles in a similar way. The similarity value obtained from the MCS is given by:

$$\begin{aligned} &Score(MCS) = \\ &w_d \sum_{1 \le k \le j-i+1} \frac{T_d - |d(p_1, i_1 + k - 1, i_1 + k) - d(p_2, i_2 + k - 1, i_2 + k)|}{T_d} + \\ &w_a \sum_{1 \le k \le j-i+1} \frac{T_a - |a(p_1, i_1 + k - 1, i_1 + k) - a(p_2, i_2 + k - 1, i_2 + k)|}{T_a} \end{aligned} \tag{3}$$

where w_d and w_a are weights used to determine the relative importance of distance and angle, and $d(p, j, j + 1)$ denote the distance between the jth and $(j + 1)$th SSE of protein p, and $a(p, j, j + 1)$ denote the angle between the jth and $(j + 1)$th SSE of protein p.

The score of the alignment between two proteins is thus given by the sum of all MCSs. Assuming that there are k MCSs, then

$$Score(p_1, p_2) = \sum_{1 \le i \le k} Score(MCS_i) \tag{4}$$

We note that in the above formulation, we have only utilized the distance and angle information for connected SSEs. In other words, we have not fully utilized the information on SSEs that are not connected.

We also note that it is possible for Score(MCS) and Score(p_1, p_2) to be 0. This occurs when the distance difference and angle difference of a pair in the alignment are larger than their respective thresholds. In these cases, by only looking at the alignment score, we are unable to differentiate between two proteins that contain similar SSEs and those that do not. To address this problem, similarity of two proteins is measured by a 2×1 vector:

$$Sim(p_1, p_2) = (Score(p_1, p_2), \sum_{1 \le i \le k} \frac{Len(MCS_i)}{L}) \tag{5}$$

where $Len(MCS_i)$ denote the length of the ith MCS. The additional second component is used to recover the “lost type information” due to large distance and angle differences.

3.2. *The Alignment Scheme: Algorithm SCALE*

As mentioned, our approach is to align the SSE sequences, and use the AD matrices to compute the degree of similarity. We call our algorithm SCALE (Structure-Conscious ALignment of secondary structure Elements). Since sequence alignment problem is so well studied, there are many good existing algorithms that we can employ to find the optimal alignment efficiently. We adapted the Needleman-Wunsh dynamic programming algorithm[10] for our work. The algorithmic description of our alignment scheme, SCALE, is shown in Figure 3. We note that in the figure, we only show the computation of the scores, and have omitted the backtracking steps to obtain the best alignment.

As shown, the algorithm is very similar to the basic dynamic programming algorithm of Ref. 10. The main differences between the proposed scheme and the basic dynamic programming scheme are the followings:

- Since the score is computed for pairs of connected SSEs, the matrix size is smaller by 1 – given k SSEs, there are $k - 1$ connected SSE pairs. This explains the “-1” in the algorithm (lines 3, 5, 7, 8).
- Since we are dealing with a 2D similarity vector, in our algorithmic description, we have essentially “overloaded” the addition operation, i.e., “+” in line 9. In our work, each individual component is added and cumulated accordingly.
- The score to be added to cell $[i, j]$ is determined by the similarity between the $S[i, i + 1]$ and $T[j, j + 1]$, i.e., the pair of connnected SSEs starting at i and j.

We note that by performing alignment on SSE sequences, our proposed scheme is efficient. With respect to existing algorithms that operate at atom levels, we not only reduce a problem from 3D to 1D, but also reduce the complexity from $\Theta(NM)$ to $\Theta(nm)$, where N and M are the number of atoms in the proteins and n and m are

Algorithm SCALE.
Input: SSE sequences S and T, angle distance matrices AD_S and AD_T, gap penalty g
Output: dynamic programming matrix A with alignment scores tabulated in $A(m,n)$.
$A(i,j)$ provides the score for the best alignment of $S[1..i]$ and $T[1..j]$.

```
1.  $m = \text{Length}(S)$ // find length of $S$
2.  $n = \text{Length}(T)$ // find length of $T$
3.  for $i = 0$ to $m-1$ do
4.      $A[i,0] = i * g$ // initialize first column
5.  for $j = 0$ to $n-1$ do
6.      $A[0,j] = j * g$ // initialize first row
7.  for $i = 1$ to $m-1$ do // now iterate over every element
8.      for $j = 1$ to $n-1$ do
9.          $A[i,j] = \max(A[i-1,j] + g,$
                        $A[i-1,j-1] + Sim(S[i,i+1], T[j,j+1]),\ A[i,j-1] + g)$
10. return $A$
```

Fig. 3. A dynamic programming based algorithm.

the number of SSEs in the proteins. Clearly, $N >> n$ and $M >> m$. Matrix-based approaches (e.g., Ref. 32), on the other hand, has complexity of $\Theta(n^2m^2)$.

4. Protein Structure Database Search

As we shall see in our experimental study, SCALE is a promising structure comparison algorithm. To search a database of protein structures, SCALE can be employed to compare the query protein against all protein structures in the database. However, usually, only the top few proteins with the relative larger similarity are of biologists' interest. It is thus a waste of time and resources to rank the whole database. Therefore, we design an index to speed up the search process by pruning away very dissimilar proteins in the database before running a comparison algorithm. Given a query protein, the index would be traversed first which returns a list of proteins. Instead of feeding the comparison method with the entire database, now, only a portion of it (obtained from the index search), which hopefully contains a superset of the top few most similar proteins, is actually compared. Any structure comparison method can be used for the comparison step; in this work, we use SCALE. The following subsections will introduce the index structure and how it is used to speed up searching.

4.1. *Index structure*

Similar structures must share some common substructures, and hence common SSE-multiplets (e.g., three SSEs form an SSE-triplet). Our idea is, for each kind of connected SSE-multiplet, to record all the proteins containing it. Given a query protein, its similar proteins must share many common SSE-multiplets with it, thus, they must be in the protein list(s) containing certain SSE-multiplet(s) that the query protein contains.

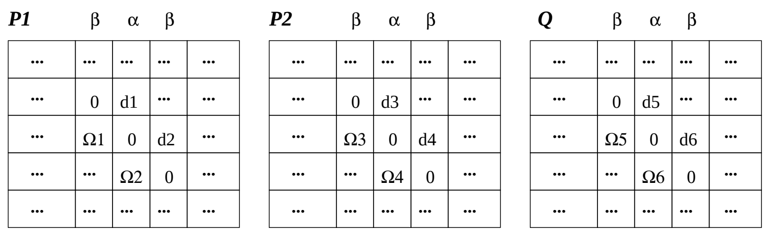
Our index is a multi-attribute-tree based on connected SSE-triplets. A protein with n SSEs generates $n-2$ connected SSE-triplets, from 1-2-3, 2-3-4, to $(n-2)-(n-1)-n$. For example, the protein `1bik` whose SSE sequence is β-β-α-α-β-β-α has the following connected SSE-triplets in order: β-β-α, β-α-α, α-α-β, α-β-β and β-β-α.

The first level of the index is based on type. There are eight possibilities of SSE type for a triplet, therefore, this level has eight nodes. The second level of the index is based on dihedral angle. The range of dihedral angle is divided into m intervals. Since each SSE-triplet has two dihedral angles, there are m^2 nodes under each node of the type level. The lowest level is based on closest distance. The range of closest distance is divided into n intervals, and hence there are n^2 nodes under each angle node. Each distance node points to a leaf page, which contains the IDs of all the proteins in the database that have at least one SSE-triplet whose type, angle and distance values matching with the values in the path from the root to that leaf page. The overall structure is illustrated in Figure 4.

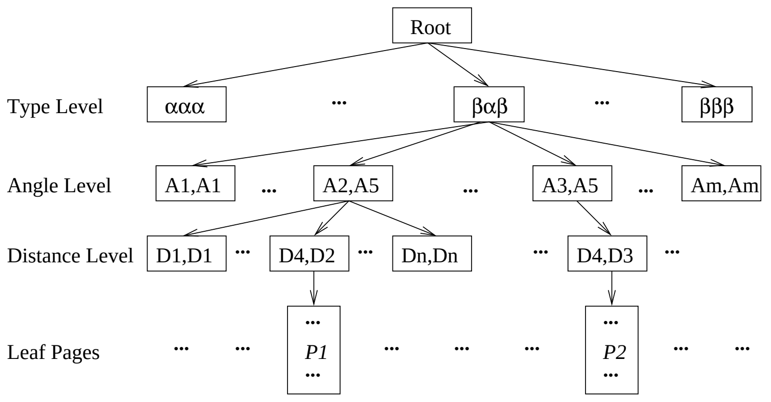
4.2. *Searching the index structure*

Given a query protein, we need to go through the following steps to get the might-be-similar protein ID list:

(1) Identify all the connected SSE-triplets in the query protein.
(2) For each SSE-triplet identified, get its type, angle and distance values. Then, traverse the index in the following way:
    (a) Move to the corresponding type node from the root.
    (b) Extend each angle and distance value by E_a and E_d on each side, respectively. Suppose originally the angles are a_1 and a_2, distances are d_1 and d_2, now we have ranges: $[a_1 - E_a, a_1 + E_a]$, $[a_2 - E_a, a_2 + E_a]$, $[d_1 - E_d, d_1 + E_d]$ and $[d_2 - E_d, d_2 + E_d]$.
    (c) From the type node in step (a), move to all the angle nodes whose two angle intervals intersecting with the range $[a_1 - E_a, a_1 + E_a]$ and $[a_2 - E_a, a_2 + E_a]$ correspondingly.
    (d) From all the nodes reached at the angle level, move to all the distance nodes whose two distance intervals intersecting with the range $[d_1 - E_d, d_1 + E_d]$ and $[d_2 - E_d, d_2 + E_d]$ correspondingly.
    (e) Union and return the protein ID lists from all the leaf nodes pointed by a distance node reached in the above step.
(3) Union the protein ID lists returned by all SSE-triplets from the previous step. Keep a count of how many lists a protein ID belongs to. This provides a lower



(a) AD matrices



(b) Index structure

Fig. 4. An example of the proposed index structure. Suppose $P1$ and $P2$ are two proteins in the database, and both have an SSE-triplet whose type is β-α-β. The portion of their AD matrices corresponding to that SSE-triplet are shown in (a). As shown in (b), the index consists of type, angle, distance level of internal nodes and leaf pages. Ai and Dj (for $1 \leq i \leq m$ and $1 \leq j \leq n$) are the ith and jth interval of angle and distance, respectively. Suppose $\omega 1 \in A2$, $\omega 3 \in A3$, $\omega 2, \omega 4 \in A5$, $d1, d3 \in D4$, $d2 \in D2$, and $d4 \in D3$, then, $P1$ and $P2$ are inserted in the leaf pages shown in (b).

bound of how many common SSE-triplets shared in this protein and the query protein. The larger it is, the more similar these two proteins are presumed to be.

(4) Return all the protein IDs associated whose count is at least Th, where Th is a threshold controlling how stringent we want the pruning to be—the larger it is, the more common SSE-triplets we expect to be shared between the query and the picked protein, and thus, more proteins with less common SSE-triplets are pruned.

Figure 4 also illustrates the query process of one SSE-triplet of the query protein Q. Suppose Q has an SSE-triplet of type β-α-β and its AD matrix is shown in (a). Assume before processing this SSE-triplet, Q has already hit $P1$ twice, but no hit on $P2$, thus the protein list $PL_Q = \{\ldots, (P1, 2), \ldots\}$. Suppose $\omega 5 - E_a \in A2$, $\omega 6 \in A5$, $d5 \in D4$ and $d6 + E_d \in D3$. Therefore, to process this SSE-triplet of Q, we shall follow the paths Root-$\beta\alpha\beta$-$(A2, A5)$-$(D4, D2)$ and Root-$\beta\alpha\beta$-$(A3, A5)$-$(D4, D3)$. The first path leads to the leaf page containing $P1$ while the second one points to the leaf page of $P2$. Hence, the counter for $P1$ is increased by 1 and $P2$ is inserted into the protein list: $PL'_Q = \{\ldots, (P1, 3), \ldots (P2, 1), \ldots\}$.

5. A Performance Study

In this section, we present a performance study to evaluate the SCALE algorithm and the proposed index. As references, we compare SCALE against the following two methods:

- BLAST[16]. BLAST is the standard tool that molecular biologists used to search for sequence similarity in protein databases. Comparing with it allows us to see the benefits of incorporating structural information.
- TOPSCAN[3]. Like SCALE, TOPSCAN captures some structural information (e.g., proximity, size, direction, etc.). However, it represents protein structure as a sequence of symbols and employs sequence alignment technique to determine the similarity between proteins. By comparing with TOPSCAN, we see how critical are the distance and angle information, and how much can be gained by these information.

The SCALE algorithm is implemented in the C++ programming language. We have also obtained the source code of TOPSCAN from the authors. The BLAST version that we used is NCBI BLAST[27]. The threshold values of Eq.(3) are set as follows: T_d is set to 10 Armstroms and T_a is set to 30 degrees. We consider both the angle and distance information to be equally important by setting both w_a and w_d to be equal to 1. These are also values used in Ref. 32. For TOPSCAN and BLAST, we used the default settings suggested by the respective packages.

We compare the various schemes on their *effectiveness* and *efficiency*. The former determines how accurately a method is in identifying structually similiar proteins given a query protein. The latter reflects how fast a scheme is, and indicates how well the scheme will scale in large databases.

We conducted two sets of experiments. The first study examines the accuracy of the various methods. In this set of experiments, we followed the approach adopted in Ref. 32 and used the same dataset of 90 proteins selected from the SCOP database[31]. The second study evaluates the efficiency of the various schemes. The dataset involves over 1000 proteins.

To examine the effectiveness of the index, we switched to use CATH[4] protein categorization scheme since large-scale structure-based classification is not available in SCOP. We selected all the proteins from Protein Data Bank[29] that comprise either single domain, or multiple domains that belong to the same CATH group, and wherein we are able to generate SSE information from WebMol program successfully.

Finally, 8223 such proteins form the protein database for our index performance study. All the experiments are preformed on SUN's E450 running solaris.

5.1. *Effectiveness of SCALE*

As mentioned, we have used 90 proteins to study the effectiveness of the various methods. These proteins are obtained from the SCOP database[18] and have been carefully selected with the following properties[32]:

- *Proteins that have less than 40% homology.* We only want to consider less homologous proteins since homologous proteins have less significance in estimating the performance of the method.
- *Proteins belonging to All alpha, All beta, Alpha and Beta in SCOP.* Other categories in SCOP are not always classified from the structural viewpoint.
- *Proteins that have more than 7 SSEs and fewer than 41 SSEs.* Proteins that have few SSEs have less structural information to work on.

The selected 90 proteins are shown in Tables 1-4.

Table 1. 18 proteins in All Alpha class

PDB Code	Classification in SCOP (All Alpha)
1eca, 1hlb, 2hbg, 2lhb, 3sdh	Globin-like
1bbh, 2ccy, 2tmv	Ferritin-like
1rtp, 1scm, 2sas, 2scp	EF Hand-Like
2gst, 1glp	Gluthathione S-transferases, C-terminal domain
1ezm, 8tln	Thermolysin-like metallopeoteases, C-terminal doman
1cpt, 1phg	Cytochrome P450

Table 2. 31 proteins in All Beta class

PDB Code	Classification in SCOP (All Beta)
1cd8, 1cid, 1hsb, 1tlk, 1cfb, 2mcm	Immunoglobulin-like beta-sandwich
2cas, 2stv	Viral coat and capsid proteins
4fgf, 1ilb, 1tie, 1hce	beta-Trefoil
1arb, 2sga, 3sgb, 4sgb, 4htc, 1ppf, 3rp2	Trypsin-like serine proteases
2hpe, 2rsp, 2er7, 1mpp	Acid proteases
1hbq, 1epa, 1mup, 1ftp, 1icn, 1crb	Lipocalins
1sri, 1avd	Streptavidin-like

For all the proteins, the sequence of SSEs and the angle-distance matrix (AD) are computed from PDB files of the Protein Data Bank using a modified version of the WebMol[15] program.

Essentially, we perform all-against-all comparison of the 90 proteins. The following are the steps of the experiment for one protein comparison:

(1) Pick one protein out of the 90 proteins and compute similarity values against all other proteins (using one of the methods studied), including itself. This will

Table 3. 35 proteins in Alpha and Beta class

PDB Code	Classification in SCOP (Alpha and Beta)
1byb, 1ghr, 1nar, 2acq, 2mnr, 4enl, 1oyc	beta/alpha (TIM)-barrel
3cox, 1pbe, 3grs	FAD/NAD(P)-binding domain
3chy, 2fcr, 2fx2, 1ovf, 1cus	Flavodoxin-like
1dhr, 1ldm	NAD(P)-binding Rossmann-fold domains
5p21, 2reb	P-loop containing small nucleotide triphosphate hydrolases
1cse, 2sic	Subtilases
3cla, 1eaf	CoA-dependent acetyltransferases
1ede, 1tca, 3tgl	alpha/beta-Hydrolases
4cpa, 2ctc, 1amp	Phosphorylase/hydrolase-like
8abp, 1gca, 2lbp	Periplasmic binding protein-like I
1pda, 1sbp, 1omp	Periplasmic binding protein-like II

Table 4. 6 proteins in Alpha and Beta class

PDB Code	Classification in SCOP (Alpha + Beta)
7rsa, 1onc	Ribonuclease A-like
1frd, 1lgr	beta-Grasp
1aya, 2pnb	SH2-like

generate a list of similarity values of the picked protein compared against each of the 90 proteins. Each element of the list is a triple, (*proteinA, proteinB, sim*), where *proteinA* is the protein name of the picked protein, *proteinB* is the name of the protein compared against and *sim* is a similarity score (in the case of SCALE, it is a vector described in Eq.(5)).

(2) Sort this list (according to the ordering described in Section 4 for SCALE).

(3) Pick the top n proteins from the sorted list, and count the number of these top n proteins that belong to *proteinA*'s SCOP category. Let this number be m. Let the number of the proteins in the relevant SCOP category be N. We then compute the *precision* and *recall* as follows:

$$Precision = \frac{m}{n}$$

$$Recall = \frac{m}{N}$$

The above steps are performed for every protein within each of the SCOP category, namely, All Alpha, All Beta, Alpha / Beta and Alpha + Beta. Therefore, all-against-all comparison is performed on the 90 proteins. For each SCOP category, the average count in step 3 for the category is computed and recorded.

We have performed this experiment with SCALE, TOPSCAN and BLAST algorithms to see how accurate they are. The results of the experiments are shown in Figure 5 for the All Alpha category, Figure 6 for the All Beta category, Figure 7 for the Aplha / Beta category and Figure 8 for the Alpha + Beta category.

From the figures, we can make several interesting observations. First, as expected, all the schemes' precision decreases with increasing n, and their recall increases as n increases. Second, we observe that BLAST did not perform well in terms of recall and precision. This is expected as the proteins used in our study have little sequence similarity. Since structural similarity are not captured by the sequence similarity, BLAST is unable to pick them out. This clearly shows the need for 3D structural information in detecting structural similarity.

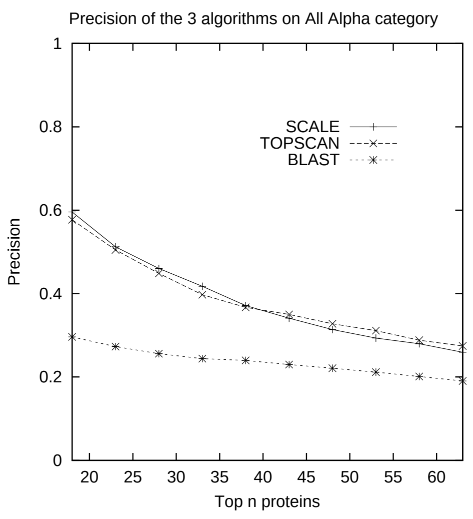
Third, SCALE performs better than TOPSCAN for the All Beta and Alpha/Beta categories. This shows that the angle and distance measures used by SCALE is an important contribution to determining structural similarity between proteins. For the All Alpha category, both schemes perform equally well.

Finally, all the schemes do not perform well for the Alpha + Beta category. But as we increase n, we see that SCALE's precision and recall improves as compared to those of TOPSCAN and BLAST. The Alpha + Beta category groups together the proteins that have mainly anti-parrallel beta sheets with alpha regions and beta regions segregated. The SCALE and TOPSCAN experiments for this category ranked proteins from the All Alpha and All Beta categories higher than proteins from its own category. This is due to large segregated alpha or beta regions within the protein getting aligned better with proteins that contain only aplha helices or proteins that contain only all beta sheets as compared to other proteins with segegrated alpha and beta regions. Both algorithms SCALE and TOPSCAN produce this result because both algorithms only consider information from connected SSEs. The protein from the Aplha + Beta category may be structurally similiar to a All Beta category protein, as the whole segregated beta region may match the entire All Beta category protein. But, they are categorized by SCOP in different categories. In order to be able to identify them as belonging to different SCOP categories, we may require the use of extra information of angles and distances of non-connected SSEs.

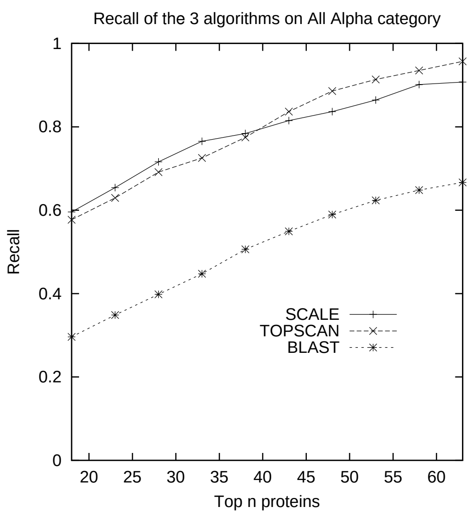
5.2. *Efficiency of SCALE*

To evaluate the efficiency of SCALE, we perform all-against-all comparison against a extended dataset of 1000 proteins instead of 90 proteins. The average run-time of the implemented algorithm is recorded. The results of this experiement is shown in Table 5. As shown in the table, the run-time performance of SCALE is much faster than TOPSCAN. From Table 5, we can see that the run-time of SCALE is about 20 times faster than TOPSCAN. This result is expected because the representation of proteins used for TOPSCAN is not rotation invariant. As a result, TOPSCAN needs to compare two proteins 24 times, each with a different orientation. For SCALE, angles between SSEs and closest distance of approach between SSEs are rotation invariant. Therefore, comparison bewteen two proteins only need to be performed once.

To summarize, the proposed scheme, SCALE, is an effective and efficient scheme for approximate protein structure comparison. The results also showed that the angle and distance information provide critical structural information.

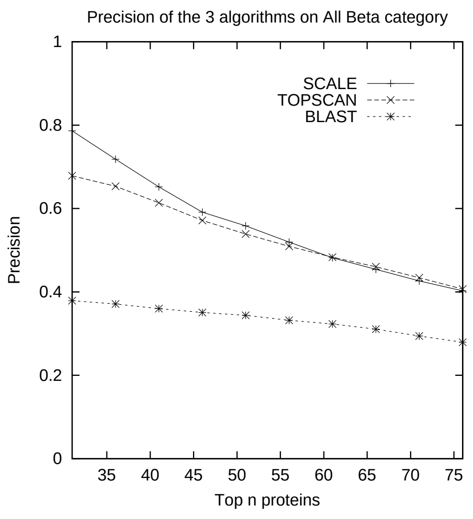


(a) Precision.

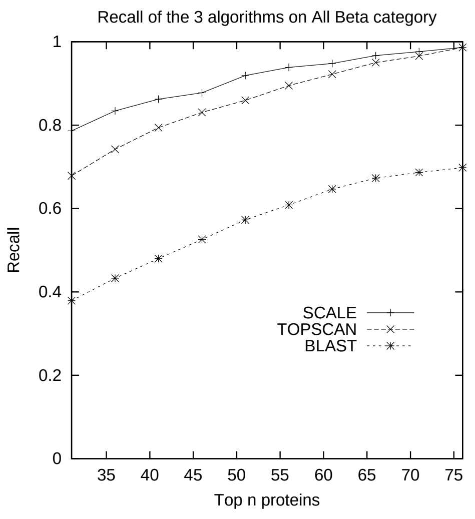


(b) Recall.

Fig. 5. All Alpha category.



(a) Precision.

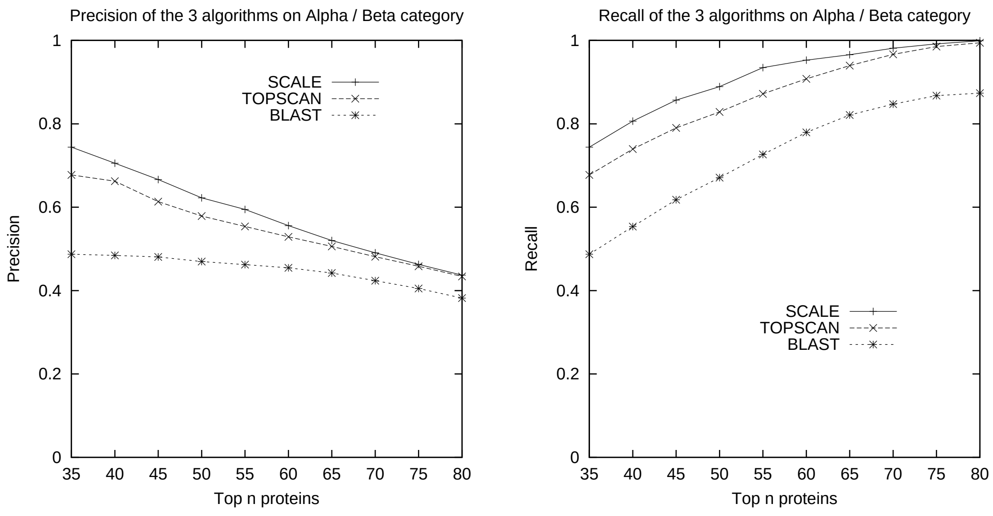


(b) Recall.

Fig. 6. All Beta category.

Table 5. Run-time performance of SCALE and TOPSCAN.

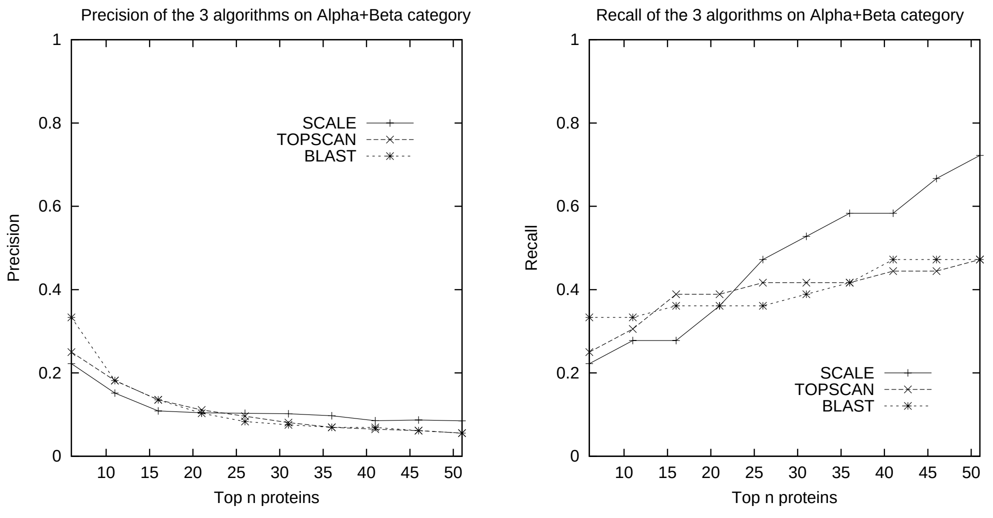
Size of dataset	SCALE run-time	TOPSCAN run-time
90 vs 90	0h 00m 10.26s	0h 02m 32.14s
1000 vs 1000	0h 16m 51.44s	4h 58m 16.59s



(a) Precision.

(b) Recall.

Fig. 7. Alpha / Beta category.



(a) Precision.

(b) Recall.

Fig. 8. Alpha + Beta category.

5.3. *Effectiveness of the index*

Recall that in the index, the ranges of dihedral angle and closest distance are divided into m and n intervals, respectively. The interval division is critical to the effectiveness of the index. Thus, we studied the distribution of angle and distance of neighboring SSEs in all proteins in our database (please refer to Figure 9. Since the angle distribution is not too skew, we simply divided its range into three equal-length intervals: $(-180°, -60°], (-60°, 60°], (60°, 180°]$.

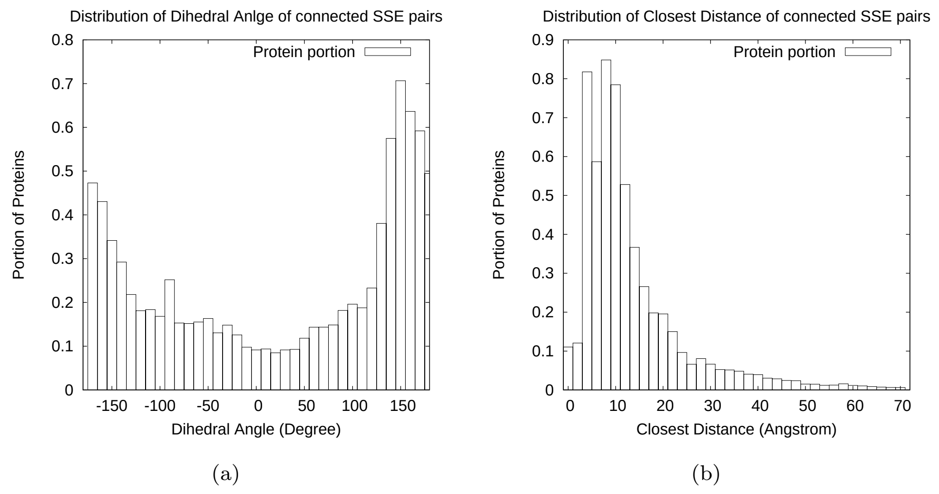


Fig. 9. The range of dihedral angle (a) and closest distance (b) of connected SSE pairs are evenly divided into 36 intervals. The y-axis is the portion of proteins in the entire dataset that has at least one SSE pair whose dihedral angle value and closest distance value fall into the corresponding range define on x-axis in (a) and (b) respectively.

16 proteins (please refer to Table 6) which are the intersection of the representative-protein-set defined by CATH[4] and our database are selected to evaluate the performance of our index scheme. Since biologists are usually only interested in the few most similar structures, as long as the protein list returned by the index contains most of the top few proteins in the SCALE result, and if the protein list is much smaller than the database size, the index will be considered as a good scheme for a pre-scanning step. The average size of protein list when varying Th value is shown in Figure 10 (a).

For each query protein, we are interested in the number of misses[c] and true drops[d]. The averages of misses and true drops for threshold Th from 1 to 5 are plotted in Figure 10 (b).

By observing Figure 10, we notice that the larger the Th is, i.e., the more stringent the protein picking criteria is, the fewer the number of proteins will be

[c]A *miss* is defined as a protein which is inside the top 100 in SCALE output, but is not retrieved when the index is used.

[d]*True drop* refers to a protein which belongs to the same CATH group as the query protein, but is not picked by the index

Table 6. 16 representative proteins from CATH

CATH group code/name	Representative protein PDB id
1.10 (Mainly alpha, Orthogonal Bundle)	1gab
1.40 (Mainly alpha, Alpha solenoid)	1ppr
1.50 (Mainly alpha, Alpha/alpha barrel)	1cem
2.10 (Mainly beta, Ribbon)	2fnz
2.20 (Mainly beta, Single Sheet)	1atx
2.40 (Mainly beta, Barrel)	1hav
2.50 (Mainly beta, Clam)	4bcl
2.80 (Mainly beta, Trefoil)	1hcd
2.90 (Mainly beta, Orthogonal Prism)	1jpc
2.102 (Mainly beta, 3-layer Sandwich)	1rie
2.110 (Mainly beta, 4 Propeller)	1hxn
2.120 (Mainly beta, 6 Propeller)	1mwe
2.160 (Mainly beta, 3 Solenoid)	1thj
3.70 (Alpha beta, Box)	1plq
3.75 (Alpha beta, 5-stranded Propeller)	4jdw
3.80 (Alpha beta, Horseshoe)	2bnh

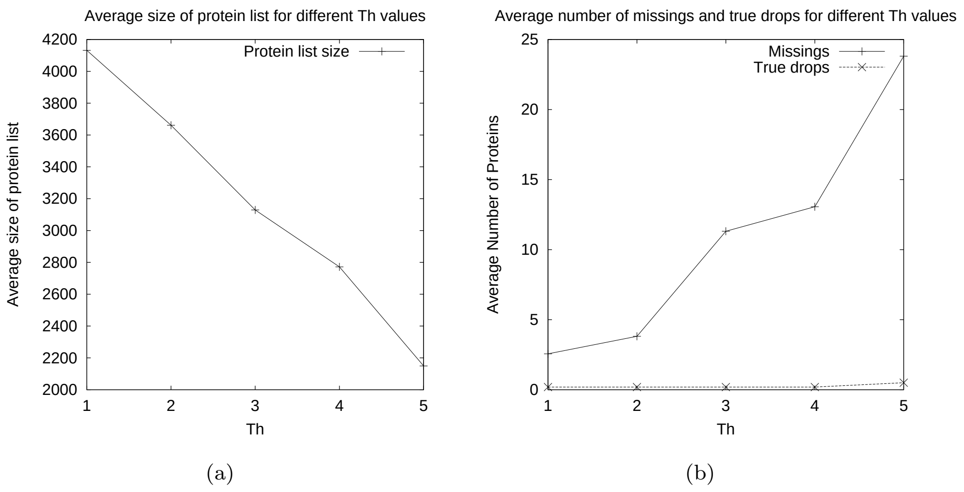


Fig. 10. (a) Size of the protein list returned by the index and (b) Number of misses and true drops when varying the value of Th. Note that the total size of the database is 8223, and the numbers are averaged on 16 representative query proteins.

returned from the index—which is desirable. The best case for protein list size (wrt the dataset we used) is at $Th = 5$, where three quarter of the database is pruned. The size of the protein list decreases almost linearly with the increase in Th. The number of true drops is always close to zero with very little increment as Th becomes larger, though the number of misses increases dramatically. However, the number of true drops is more of our interest because it compares against the true answer. From the difference between the number of misses and true drops, we found that many misses are actually dissimilar proteins, and are beneficial to be missed by using the index scheme. When Th is less than 5, the average number of true drops is always 0.2. Note that we assume a biologist is looking at the top 100 proteins, thus, the probability of missing an ought-to-be-picked protein due to the index is only 0.2%. This shows the index scheme almost does not sacrifice the accuracy of the search at all.

We shall also note that, it usually takes 14-20 minutes for SCALE to process one query against a full database with 8223 proteins, while the average time to traverse the index is less than 2 seconds (24.52 seconds for the 16 representatives). Therefore, the overhead of traversing the index is almost negligible. If we can prune away half of the database (which is the worst case in Figure 10, where $Th = 1$), we are already able to save 10 minutes. Finally, we set our Th to be 4, where the average size of protein list is 2772, and the average number of true drops is 0.2. Thus, the index prunes away 2/3 of the database, while keeping 99.8% of the most relevant proteins. Based on such an index, the time to search for a similar protein in a database can be reduced to only 1/3 of that of an exhaustive search with almost the same accuracy.

6. Related work

The problem of structural comparison or alignment is known to be NP-hard. A number of approximation methods have been developed to solve the problem of structural comparison.

DALI[23,25] is based on aligning 2D intra-molecular distance matrices. It computes the best subset of corresponding residues from the two proteins such that the similarity between the 2D distance matrices is maximized. Searches through all possible alignments of residues are performed using Monte-Carlo and branch-and-bound algorithms. *DALI* is known to be one of the most biologically sensitive methods. However, it is computationally expensive.

In *VAST*[22], all pairs of SSEs that have the same type are represented as nodes of a graph, and the nodes (SSE pairs) that share similar distance and angle properties are connected by edges. Then, clique detection algorithm is applied to find the maximal subgraph and hence the initial alignment between two structures. Next, iterative superposition is applied to get the detailed alignment at the residue level. It also takes the statistical significance of an alignment into account in calculating the best superposition. *TOP*[17] represents SSEs as vectors, and uses least square-fitting approach to superimpose them. Detailed matching at the residue level is performed based on various criteria. *CE*[26] algorithm involves a combinatorial extension of an alignment path defined by aligned fragment pairs (AFPs), which are based on local geometry of the structure. Combinations of AFPs that represent possible continuous alignment paths are selectively extended leading to a single optimal path. *LOCK*[9] represents SSEs as vectors, and superimposes them using dynamic programming to get the initial alignment. Then, atomic and core superpositionings are performed using a greedy algorithm.

Although fine-grain methods usually result in high accuracy, they are very slow. Given the rapid growth rates of the protein structural databases in the near future, there is increasing motivation to design coarse-grain approaches that are faster and reasonably accurate.

Coarse-grain approach is used in Ref. 3, 24, 32, and they treat only SSEs as the basic elements. *3D-Lookup*[24] also uses vectors to represent the SSEs, and uses geometric hashing to hash them into the search space. Ref. 32 represents the various relationships, such as distances, angles, types and lengths, among the SSEs in a protein as matrices. Then it finds the maximum common parts of the corresponding matrices of two proteins to detect their similarity. *TOPSCAN*[3] uses symbolic linear representation of SSE vectors using the properties such as SSE type, direction, length, proximity, etc, and applies Needleman and Wunsch dynamic programming algorithem to align two linear representations.

Some survey reports[1,21,28] are available for reviewing the background theories and the various methods in this area.

Our method shares some common features with *TOPSCAN*[3], for instance, both work on SSE level, both represent a protein linearly so that a 3D problem is reduced to 1D, both employ dynamic programming algorithm. But there are two significant differences. The spatial relationship between SSEs is captured much more precisely in our method than in *TOPSCAN*. *TOPSCAN* divides the 3D space into 6 directions, namely, forward and backward on each coordinate. Thus, a β sheet with 1° to x axis will get the same representation as a β sheet with 40° to x axis. But in our method, there is even no approximation at all. All the real numbers are used in the representation as well as the alignment. Besides the loose capture of spatial information, in *TOPSCAN*, since the structure is based on a 3D coodinate system, the representation is not orientation-invariant. In order to compare proteins in all the different orientations, each comparison needs to rotate one of the proteins 24 times. The rotation is done via permutation of the string. But our method concerns the differences in a pairwise manner, and it is orientation-invariant. This contributes significantly to its efficiency.

There are also some works on indexing protein structures[33,8,20,13,7]. All of these methods deduce feature vectors from protein structures, and then index these feature vectors using existing indexing techniques. For example, ProtDex2[33] employs inverted-file, Ref. 8 uses histograms, while Ref. 20, 7 build classical tree structures —R-tree[2], R*-tree[11] and B+-tree, and Entropy Balanced Statistical (EBS) k-d tree for Ref. [13]. The major difference among these methods resides in where and how to get the feature vectors. Ref. 8 decomposes the 3D space into sectors where each sector corresponds to a bin in the histogram; Ref. 13 applies computer vision techniques to extract the predominant information encoded in each 2D distance matrix which is generated from 3D coordinates of protein chains; In Ref. [20], each amino acid is represented as a vector, and the four distances among the four end-points of two vectors are used to index; ProDex2[33] extracts seven properties (such as angle, average C_α-C_α distance) for each SSE pair, while Ref. [7] works on SSE-triplets.

7. Conclusion

In this paper, we have revisited the problem of protein structure comparison. We have proposed a novel scheme called SCALE that approximates the 3D structural information of a protein by its SSE sequence and pairwise distance and angle information. In this way, SCALE can compare two proteins by aligning their SSE sequences and using the distance and angle information to augment the similarity score. We have implemented SCALE, and evaluated its performance against TOPSCAN and BLAST. Our performance study shows that SCALE outperforms TOPSCAN and BLAST in terms of precision and recall. It is also more efficient than TOPSCAN.

We also proposed an index structure based on protein SSE structural properties. Experiments showed that the index is able to retrieve similar proteins from a database so that only a small portion of the database has to be explicitly compared with the query protein using structural comparison tool, such as SCALE.

We plan to extend this work in several directions. First, in our current work, we have not fully exploited the information in the *AD* matrix. In fact, we are only using the information on connected SSEs. We believe a more precise similarity measure that incorporates the distances and angles between non-connected SSEs can further improve the precision and recall by pruning away false drops. Second, we have also not tapped into information that TOPSCAN has used, e.g., size of SSEs and proximity. Studying how much this information contributes to similarity comparison between proteins is also in our agenda.

Acknowledgements

The authors would like to thank Andrew C.R. Martin of the School of Animal and Microbial Sciences, University of Reading, for providing the source code of TOPSCAN and Dirk Walther of the Department of Cellular and Molecular Pharmacology, University of California, San Francisco, for providing the source code of WebMol.

References

1. Godzik A. The structural alignment between two proteins: is there a unique answer? *Protein Science*, 5(7):1325–1338, 1996.
2. Guttman A. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 47–57, Boston, MA, June 1984.
3. Martin A. The ups and downs of protein topology: rapid comparison of protein structure. *Protein Engineering*, 13:829–837, 2000.
4. Orengo C. A, Michie A. D., Jones, Swindells M. B., and Thornton J. M. Cath: A hierarchic classification of protein domain structures. *Structure*, 5:1093–1108, 1997.
5. Orengo C. A., Michie A. D., Jones S., Jones D. T., Swindells M. B., and Thornton J.

M. Cath- a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108, 1997.

6. Orengo C. A., Jones S., Martin A. C. R., Swindells M.B., Michie A. D., Jones D. T., and Thornton J. M. Classifying a protein fold in the cath hierarchic database. *Acta Cryst.*, D54:1155–1167, 1998.
7. amoglu O., Kahveci T., and Singh A. Towards index-based similarity search for protein structure databases. In *IEEE Computer Society Bioinformatics Conf.*, pages 148–158, 2003.
8. Kriegel H. P. Ankerst M., Kastenmuller G. and Seidl T. Nearest neighbor classification in 3d protein databases. In *Int Conf Intell Syst Mol Biol*, pages 34–43, 1999.
9. Singh A.P. and Brutlag D.L. Hierarchical protein structure superposition using both secondary structure and atomic representations. In *5th International Conference on Intelligent Systems for Molecular Biology*, pages 284–293, AAAI Press, 1997.
10. Needleman S. B. and Wunsch C. D. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.
11. Schneider R. Beckmann N., Kriegel H.-P. and Seeger B. The r*-tree: An efficient and robust access method for points and rectangles. In *ACM SIGNOD Int. Conf. on Management of Data*, pages 23–25, Atlantic City, NJ, May 1990.
12. Chothia C., Levitt M., and Richardson D. Helix to helix packing in proteins. *Journal of Molecular Biology*, 145:215–250, 1981.
13. Shyu C. R. Chi P. H., Scott G. J. A fast protein structure retrieval system using image-based distance matrices and multidimensional index. In *4th IEEE International Symposium on BioInformatics and BioEngineering (BIBE 2004)*, pages 522–532, Taichung, Taiwan, March 2004.
14. Tan K.L. Yao Z. Chionh C.H., Huang Z. Augmenting sses with structural properties for rapid protein structure comparison. In *Proceedings of the 3rd IEEE Symposium on Bioinformatics and Bioengineering (BIBE'2003)*, pages 341–350, Washington DC, March 2003.
15. Walther D. Webmol - a java based pdb viewer. *Trends Biochem Sci*, 22:274–275, 1997.
16. Altschul S. F., Gish W., Miller W., Myers E. W., and Lipman D. J. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.
17. Lu G. Top: a new method for protein structure comparisons and similarity searches. *Journal of Applied Crystallography*, 33:176–183, 2000.
18. Murzin A. G., Brenner S. E. amd Hubbard T., and Chothia C. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.
19. Pearl F. M. G., Lee D., Bray J. E., Sillitoe I., Todd A. E.and Harrison A.P., Thornton J.M., and Orengo A. D. Assigning genomic sequences to cath. *Nucleic Acids Research*, 28(1):277–282, 2000.
20. Zhou X. Huang Z. and Song D. High dimensional indexing for protein structure matching using bowties. In *3rd Asia Pacific Bioinformatics Conference (APBC 2005)*, pages 21–30, Singapore, January 2005. Imperial College Press.
21. Eidhammer I. and Jonassen I. Protein structure comparison and structure patterns. *Journal of Computational Biology*, 7(5):685–716, 2000.
22. Gibrat J-F., Madej T., and Bryant H. Surprising similarities in structure comparison. *Current Opinion in Structural Biology*, 6:377–385, 1996.
23. Holm L. and Sander C. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 233:123–138, 1993.
24. Holm L. and Sander C. 3-d lookup: fast protein structure database searches at 90% re-

liability. In *3rd International Conference on Intelligent Systems for Molecular Biology*, pages 179–187, AAAI Press, 1995.
25. Holm L. and Sander C. Mapping the protein universe. *Science*, 273:595–602, 1996.
26. Shindyalov I. N. and Bourne P.E. Protein structure alignment by incremental combinatorial extension (ce) of the optimal path. *Protein Engineering*, 11(9):739–747, 1998.
27. NCBI. Ncbi-blast readme. In *ftp://ncbi.nlm.nih.gov/blast/db/README*, 2002.
28. Singh A. P. Protein structure alignment: A comparison of methods. In *http://cmgm.stanford.edu/ brutlag/Papers/singh00.pdf*, 2000.
29. PDB. PDB database. In *http://www.rcsb.org/pdb/*, 2002.
30. Taylor W. R. and Orengo C. A. Protein structure alignment. *Journal of Molecular Biology*, 208(1):209–229, 1989.
31. SCOP. Structural classification of proteins. In *http://scop.mrc-lmb.cam.ac.uk/scop/*, 2002.
32. Ohkawa T., Hirayama S., and Nakamura H. A method of comparing protein structures based on matrix representation of secondary structure pairwise topology. In *4th IEEE Symposium on Intelligence in Neural and Biological Systems*, pages 10–15, 2001.
33. Aung Z. and Tan K. L. Rapid 3d protein structure database searching using information retrieval techniques. *Bioinformatics*, 20:1045–1052, May 2004.