JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 11, Number 6, 2004

Pp. 1061–1072

PROuST: A Comparison Method of Three-Dimensional Structures of Proteins Using Indexing Techniques

MATTEO COMIN,[1] CONCETTINA GUERRA,[1], GIUSEPPE ZANOTTI[2]

ABSTRACT

We present a new method for protein structure comparison that combines indexing and dynamic programming (DP). The method is based on simple geometric features of triplets of secondary structures of proteins. These features provide indexes to a hash table that allows fast retrieval of similarity information for a query protein. After the query protein is matched with all proteins in the hash table producing a list of putative similarities, the dynamic programming algorithm is used to align the query protein with each protein of this list. Since the pairwise comparison with DP is applied only to a small subset of proteins and, furthermore, DP reuses information that is already computed and stored in the hash table, the approach is very fast even when searching the entire PDB. We have done extensive experimentation showing that our approach achieves results of quality comparable to that of other existing approaches but is generally faster.

Key words: protein structure comparison, secondary structure, indexing, dynamic programming.

1. INTRODUCTION

THE ENORMOUS GROWTH of experimentally determined three-dimensional structures of proteins, either by X-ray diffraction or NMR techniques (PDB) poses the problem of the structural comparison of distantly related proteins. It is well known that similar three-dimensional folding can be present in proteins that bear small or even undetectable amino acid sequence homology. Three-dimensional (3D) similarity can be useful, for instance, in structural proteomics projects, where the discovery of a new fold is often coupled to the search for a function: structural similarity can help to assign a function to a newly determined protein structure. Several servers that perform an automatic search for structural homology are already available. Among them are DALI, CE, SSAP, and VAST. They are based on different algorithms and often produce slightly different results. In a recent paper by Novotny *et al.* (2004), a comparative evaluation of all the 11 servers publicly available was used to assess their performance, in particular in terms of ability in recognizing established similarities. The conclusion was that, despite the fact that some servers behave better than others, none of them can be considered fully reliable; i.e., none of them gives a 100% success

[1]Department of Information Engineering, University of Padova, 35131 Padova, Italy.
[2]Deptartment of Chemistry and VIMM, University of Padova, 35131 Padova, Italy.

rate. As a consequence, the authors conclude that more than one server should be used before the novelty of a fold is established.

In this paper, we present an approach to protein structure comparison that combines indexing and dynamic programming. Recently, we proposed "indexing techniques" for comparing three-dimensional structures of proteins (Ferrari *et al.*, 2003; and Guerra *et al.*, 2002). The method is based on the use of simple geometric features of secondary structures of proteins. Briefly, the entire protein is reduced to linear segments, each of them representing a secondary structure element (SSE). All possible triplets of such elements are considered and their geometric properties (angles and distances) are computed and then used to build a hash table. For a given set of proteins, each entry of the table contains all triplets characterized by similar dihedral angles. Once the table is built, it can be easily used to perform different kinds of internal comparisons or statistical analysis (Platt *et al.*, 2003) or for matching a query protein against the database. Given a query protein, one can retrieve from the index-based table a list of similar proteins ranked according to a given similarity criterion. This step is very fast since it does not examine each stored protein separately but only considers through indexing the entries of the database corresponding to similar substructures. However, since false positive results may be found at this stage, a more detailed and refined analysis is needed to improve the quality of the results.

In this paper, we describe a new dynamic programming (DP) algorithm for pairwise structure alignment and show how to integrate indexing and dynamic programming to achieve robust and reliable comparison in an efficient way. In the integrated approach, after the query protein is matched with all proteins in the index-based table producing a list of putative similarities, DP is used to determine an alignment according to protein sequence order of the SSE of the query protein with each protein of this list. DP is based on a score function derived from the same geometric properties computed and used by indexing. After this step, all proteins of the list are reranked according to the more stringent similarity criterion used by DP thus obtaining a new and final list of proteins similar to the query protein. As an additional result, from the aligned SSE, a superposition at the atomic level of the query protein with any selected protein from the list can be obtained. Since the pairwise comparison with DP is applied only to a subset of proteins whose size is much smaller than the original set and, furthermore, DP reuses information that is already computed and stored, the computational complexity of the overall method can be kept within reasonable bounds, even when searching the entire PDB.

Based on these principles, we have built a server for protein structure comparison called PROuST (PROtein STructure comparison) (a preliminary version is available at *www.angela.dei.unipd.it/PROuST*). It contains the geometric database of the entire Protein Data Bank, and its characteristics are described here. More precisely, the overall method is presented in Section 3, where Subsection 3.1 reviews the table construction and search, and Subsection 3.2 introduces the dynamic programming approach, based on the geometric features computed and stored by indexing. Previous work on protein structure comparison is briefly reviewed in Section 2, concentrating on existing indexing and DP approaches. Experimental results are presented in Section 4, which also contains a discussion on the performance of the method and its comparison with other existing methods.

2. PREVIOUS WORK

A large number of approaches to structural alignment and comparison have been proposed and developed to date (e.g., Abagyan and Maiorov, 1989; Brown *et al.*, 1996; Lancia *et al.*, 2001; Grindley *et al.*, 1993; Gibrat *et al.*, 1996; Shindyalov and Bourne, 1998; Overington *et al.*, 1993). Some of the methods are based on comparing the distance matrices of each structure with the objective of minimizing the difference in intraatomic distances of the aligned structural elements (Holm and Sander, 1996, 1996b). Other methods try to minimize the interatomic distances of two structures (Sali and Blundell, 1990; Taylor and Orengo, 1989). In this section, we focus on methods based on dynamic programming and on indexing techniques, since our own approach is a combination of these two. Surveys on the general structure comparison problem are found in Bourne and Weisseig (2003), Ferrari and Guerra (2003), and Lemmen and Lengauer (2000).

Indexing techniques, initially proposed to solve the model-based object recognition problem in the area of computer vision (Lamdan *et al.*, 1990), have been applied to compare protein structures at the atomic level (Fischer *et al.*, 1995) or at the level of secondary structures (Alesker *et al.*, 1996; Holm and Sander, 1995).

Indexing methods mainly consist of choosing a suitable 3D representation of the structures in terms of features that can be conveniently used to access and search a table or database of 3D structures. Fischer *et al.* (1995) use as features the affine coordinates in a reference frame formed by quadruples of points (the C_α atoms). These features generate numbers from which indexes to specific locations of the table are derived. Once constructed, the table is used as look-up table to retrieve hypotheses of similarity for a query protein. This approach allows fast retrieval even for large databases since the search for a query protein is not performed on a pairwise basis with all the proteins in the database.

Dynamic programming has been used for protein structural alignment, applied either at the level of secondary structures or at the atomic level. Often the DP algorithm has been applied iteratively: the results of one iteration of the DP are used to set new values of the scoring function until convergence (Taylor, 1999; Akutsu, 1996). In STRUCTAL (Gerstein and Levitt, 1998), the alignment between two protein backbones is obtained by iteratively applying DP to minimize the RMSD between the aligned atoms. Initially, the DP matrix is filled with the pairwise interatomic distances between all atoms from the two structures. Following each iteration of DP, the obtained correspondences are used to derive a rigid transformation that maps one protein into the other: new pairwise distances are computed after this transformation.

In LOCK (Singh and Brutlag, 1997), a combination of secondary structures alignment and atomic superposition is used to minimize the RMSD between atoms. The DP is iteratively applied to derive an alignment of secondary structures. Initially, the scores between vectors are orientation independent; following each DP iteration, the new results are used to derive orientation-dependent scores for pairs of secondary structures. Every subsequent successive DP iteration involves orientation-independent as well as the newly computed orientation-dependent scores.

3. METHODS

Our approach to protein structure comparison uses a combination of techniques that allow fast retrieval of similarity information from a database containing all the protein structures present in the PDB. The steps of the methods are shown in the workflow of Fig. 1. For a query protein, the first step is to search the hash table using the same features that are used for the table construction. The search returns hypotheses of similarity with the query protein as a list of candidate proteins. Then, in the next step, dynamic programming is used

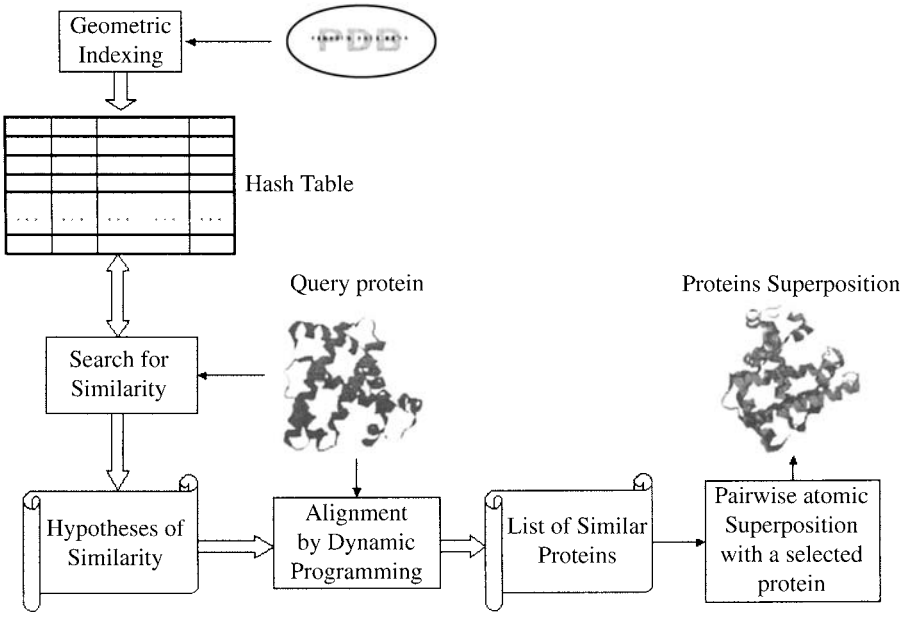

FIG. 1. PROuST workflow: the tasks for structural comparison of proteins.

for pairwise structure alignment between the query protein and the proteins of this list. DP also returns a similarity score according to which the proteins of the list are rearranged and sorted to produce the final list of similar proteins. From the aligned SSE, it is also possible to obtain a superposition at the atomic level of the query protein with any protein selected from the list. In the following, we describe in detail each step of this workflow.

First, we design an index structure based on geometric features of triplets of secondary structures; the same features are the basis for the fast search in the database as well as for the pairwise comparison that we will be described in Section 3.2. Secondary structures are represented by the best-fit line segments. A singular-value decomposition (SVD) routine is used to associate a segment to each α-helix and strand of a β-sheet (Gerstein, 1992). The angles and distances of triplets of segments are used to store the protein structures and retrieve similarity information.

Let P be a protein and $(p_1, \ldots, p_n)$ the best-fit line segments associated to its n SSE, listed according to their order along the backbone. Consider a triplet (p_u, p_v, p_z) of segments with $u < v < z$; let $(\alpha_{uv}, \alpha_{vz}, \alpha_{uz})$ be the dihedral angles formed by the segments of the triplet. Furthermore, $(d_{uv}, d_{vz}, di_{uz})$ are the distances between the three pairs of segments of the triplet (the distance between two segments is defined as the distance between their midpoints).

The angle values of the triplets of segments, after proper quantization, define the index structure for the database that stores the proteins. The search procedure accesses the database looking for triplets of SSE of stored proteins that are equivalent to triplets of the query protein. Two triplets are considered equivalent if they have similar angle and distance values. More precisely, consider a triplet (q_r, q_s, q_t), $r < s < t$, of secondary structures segments of another protein Q with the angles $\phi_{rs}, \phi_{st}, \phi_{rt}$ and distances h_{rs}, h_{st}, h_{rt}.

Definition. *The two triplets* (p_u, p_v, p_z) *and* (q_r, q_s, q_t) *are* equivalent *if*

$$|\alpha_{uv} - \phi_{rs}| < TA,\ |\alpha_{vz} - \phi_{st}| < TA \text{ and } |\alpha_{uz} - \phi_{rt}| < TA$$

and

$$|d_{uv} - h_{rs}| < TD,\ |d_{vz} - h_{st}| < TD \text{ and } |d_{uz} - h_{rt}| < TD$$

with TA *and* TD *given thresholds.*

The threshold values were experimentally determined and set as follows: $TA = 18°$ and $TD = 8$ Å. We tried several threshold values for *TA* in the range 5–20° and for *TD* in the range 5–15 Å and the chosen ones seem to be a good compromise between selectivity and sensitivity.

In the following, we describe the index-based method that allows us to determine equivalent triplets efficiently. The method first builds a hash table containing all proteins indexed by the angles of triplets of segments; then, given a query protein, it performs a search in the table for equivalent triplets and uses a voting process to select a list of similar proteins.

3.1. Building the database by indexing

We build a four-dimensional table with the following index structure: for a given triplet of segments, to access the appropriate entry of the table, three indexes are given by the quantized values of the angles of the triplet; the fourth index depends on the composition of the triplet in terms of the number and position of helices and strands. This index, called *triplet type*, is introduced to resolve the ambiguity of comparing a segment representing a helix with one representing a strand. The size of the cells of the table used to quantize the angle values is related to the threshold *TA* and, in our approach, it is set equal to 18°. Each cell of the table contains a list of records, one for every triplet that hashed into it. The following procedure inserts protein P into the database and is a variant of the one described by Guerra *et al.* (2002).

PROCEDURE: *Protein insertion*
All triplets of secondary structures of P are examined, and for each triplet (p_u, p_v, p_z) with $u < v < z$ the following steps are executed:

i. Compute the angles $(\alpha_{uv}, \alpha_{vz}, \alpha_{uz})$ and determine *triplet type*

ii. Access the cell of the table at the location indexed by *triplet type* and by the quantized values of $(\alpha_{uv}, \alpha_{vz}, \alpha_{uz})$

iii. Append to the list of records at that cell a new record that contains:
- the name of Protein P
- the identifier of each secondary structure element of the triplet
- the distances (d_{uv}, d_{vz}, d_{uz}) between every pair of segments in the triplet.

The above procedure is repeated for all proteins in the dataset. The construction of the table is the most computationally intensive part of the overall procedure. With the current size of the PDB (more than 23,000 structures), the storage requirement for the hash table is more than 5 Gbytes. For the insertion of a protein with n secondary structures, $O(n^3)$ triplets have to be stored into the table each involving access to secondary memory. We have described elsewhere (Ferrari *et al.*, 2003) a distributed implementation of the table construction that partitions the table across the nodes of a computational GRID (Foster and Kesselman, 1998) and achieves a good load balancing while reducing accesses to secondary storage.

3.1.1. Retrieving similarity from the table. Once the table has been built, one can retrieve similarity information for a query protein Q very efficiently. A detailed description of this phase is given by (Ferrari *et al.* (2003). Briefly, the procedure computes, for every triplet of secondary structures segments of Q, the three angles and distances between segments, then accesses the cell indexed by three angles and casts a vote to the records in the cell with similar distance values. For simplicity, in the voting process, each triplet of Q may only cast one vote to a given target protein T, and conversely, a triplet of T only contributes a vote to Q, even if it is equivalent to several triplets of Q. Because of the quantization of the angular values, the equivalent stored triplets may not lie in exactly the same hash cell as the query triplet, but in one of the neighboring cells. That is why our method examines also the adjacent cells in the search process.

Hypotheses of matches of the query protein with the stored proteins are ranked according to the number of votes they accumulate. Since false positive matches may arise, a verification phase is needed. However, in practice, simply counting the number of votes provides a reasonable matching score. The approach we have presented provides a good basis for further analysis. The hypotheses of similarity can be verified by a pairwise comparison between the proteins, as described in the next section.

3.2. Pairwise alignment using secondary structures

In this section, we describe a new method to align secondary structure elements, using the information computed and stored by the indexing method, thus integrating our two main approaches.

The alignment problem seeks associations between pairs of secondary structures of two given proteins P and Q. It can be formally described as follows. Let $(p_1, \ldots, p_n)$ be the n segments associated to secondary structures of P and $(q_1, \ldots, q_m)$ the m segments of Q. The structural alignment procedure determines an ordered set of pairs (i, j) associating secondary structure p_i of P to q_j of Q. The alignment satisfies the continuity constraints, i.e., if (i, j) and (h, k) are two aligned pairs and $i < h$, then $j < k$; furthermore, it allows gaps that correspond to nonsequential i values in consecutive pairs. The entry $M(i, j)$ of a two-dimensional score matrix stores the score of the pair of segments p_i of P and q_j of Q and is defined as follows.

Two equivalent triplets (p_{i1}, p_{i2}, p_{i3}) of P and (q_{j1}, q_{j2}, q_{j3}) of Q determine three candidate pairs of corresponding segments: (p_{i1}, q_{j1}), (p_{i2}, q_{j2}), and (p_{i3}, q_{j3}). The score $M(i, j)$ of the pair (p_i, q_j) is given by the number of times the pair (p_i, q_j) occurs in any two equivalent triplets of segments of secondary structures of P and Q. It is important to stress the fact that the similarity score is derived from the data stored in the hash table, since this is crucial to the performance of the method. In the following, we describe how to compute the matrix M from the information stored in the hash table by using a procedure similar to the search described in the previous section. We assume that protein Q has been already inserted into the hash table.

PROCEDURE. *Compute the score matrix* $M(i, j)$

Step 1.

Initialize $M(i, j)$ to 0 for $i = 1, \ldots n$, $j = 1, \ldots m$.

Step 2.
All triplets of secondary structures of P are examined and for each such triplet (p_u, p_v, p_z) with $u < v < z$ the following steps are executed:

i. Compute the three angles $(\alpha_{uv}, \alpha_{vz}, \alpha_{uz})$ and the three distances (d_{uv}, d_{vz}, d_{uz}). Access the hash table at the cell indexed by the three quantized angles and by the determined triplet type.

ii. Examine all triplets of Q, if any, in that cell. Let (q_r, q_s, q_t) be one such triplet, $r < s < t$.
if the distances of the two triplets are within the threshold *TD*, i.e.,
$|d_{uv} - h_{rs}| < TD$, $|d_{vz} - h_{st}| < TD$ and $|d_{uz} - h_{rt}| < TD$
then
$\{ M(u, r) = M(u, r) + 1;\ M(v, s) = M(v, s) + 1;\ M(z, t) = M(z, t) + 1; \}$

iii. Examine all triplets of Q, if any, in the neighboring cells of the cell considered at step ii. Let (q_r, q_s, q_t) be one such triplet, $r < s < t$.
if the distances and the angles of the two triplets are within the thresholds , i.e.,
$|d_{uv} - h_{rs}| < TD$, $|d_{vz} - h_{st}| < TD$ and $|d_{uz} - h_{rt}| < TD$ and
$|\alpha_{uv} - \phi_{rs}| < TA$, $|\alpha_{vz} - \phi_{st}| < TA$ and $|\alpha_{uz} - \phi_{rt}| < TA$
then
$\{ M(u, r) = M(u, r) + 1;\ M(v, s) = M(v, s) + 1;\ M(z, t) = M(z, t) + 1; \}$

End.

The neighboring cells are the ones with indexes differing by at most $+1$ or -1.

3.2.1. The dynamic programming algorithm. Given a pairwise scoring matrix M with $M(i, j)$ indicating the similarity score of the pair (p_i, q_j), the alignment procedure optimizes the following function: the sum over all secondary structures associations of the above score. The general recurrence used by DP is

$$D(i, j) = max\{D(i-1, j-1) + M(i, j), D[i][j-1] - gp, D[i-1][j] - gp\}$$

where $D(i, j)$ is initially set to 0. The parameter gp indicates the gap penalty and is set to 0 in our procedure since we want to allow missing helices and strands. As is well known, even for very high structural similarity, often a helix appears split into two helices in one of the proteins. The dynamic programming algorithm determines an optimal nondecreasing path in the matrix D. The total weight of the optimal path is stored in $D(n, m)$. The entire D matrix is kept in memory along with back pointers to be able to reconstruct at the end the optimal path and its length, i.e., the number of aligned secondary structures.

The DP is repeatedly applied to all proteins from the list produced by the index-based search for a given query protein. The proteins of such list are now reranked by a new similarity measure derived from the values $D(n, m)$ computed by DP. To account for differences in protein size and penalize the occurrences of a small substructure in large proteins, we introduce the following similarity score:

$$Score = D(n, m)(SS_{al}/SS_{tot})^2$$

where SS_{al} is the number of aligned SSE and SS_{tot} is the total number of SSE of protein Q.

In summary, DP produces a list of similar proteins by taking into consideration both the continuity constraint and the triplet equivalences, thus overcoming the drawbacks of indexing. We recall that in the indexing method it can happen that two pairs of equivalent triplets contribute to a result even if together they do not satisfy the continuity constraint.

Before concluding this section, we want to point out that we have investigated graph-based algorithms to align secondary structures using the features contained in the hash tables. These algorithms tend to be computationally more demanding than DP; on the other hand, the added complexity does not seem to be justified since the results obtained by DP are already quite satisfactory, as we will show later.

3.3. Protein atomic superposition

The final stage of the proposed scenario for protein structure comparison is the superposition of the two molecules, that is, the determination of the rigid transformation that "best" overlaps the two proteins.

In the superposition step, we consider a protein representation that includes the coordinates of the C_α atoms. This allows us to formulate the goodness of the overlap in a more common way and to assess the performance of our method in comparison with other approaches. Most of the existing approaches measure the quality of their results considering the number of aligned atoms and the root mean square deviation (RMSD) between them.

Given a set of pairs of points in 3D space, finding the roto-translation that minimizes the RMSD is a well known optimization problem. Among the many solutions that can be applied, we have chosen Horn's algorithm, which provides a closed form solution (Horn, 1987). Horn's algorithm computes, in a time proportional to the number of pairs, the 4×4 transformation matrix that minimizes the RMSD between the input points.

As seen in the previous section, the result of DP is a set of associations between secondary structures. We use these associations as the starting point in developing a superposition algorithm. In other words, we find pairs of atoms that are constrained by the secondary structures associations produced by DP.

In the following, we describe how to select pairs of corresponding points of the aligned structures as input to Horn's procedure. The steps of the superposition procedure are the following:

1. Compute an initial rigid transformation based on the starting and ending residues of all aligned secondary structures.
2. After applying the obtained transformation to one of the proteins, for each pair of aligned secondary structures, find pairs of nearest atoms.
3. Use the pairs of nearest atoms computed in step 2 to derive a new transformation that minimizes the RMSD.
4. Repeat steps 2 and 3 until RMSD converges.

Step 1 produces typically a good starting point for the rigid transformation. Often, however, the number of atoms involved in this step is very small, and the transformation needs to be refined. To this end, from the obtained transformation, we extract pairs of atoms that are potentially homologous, by determining for each atom of a secondary structure of the first protein the closest atom in the associated secondary structure of second protein. From these pairs of points, we compute a new transformation by minimizing the RMSD. The algorithm repeats these two steps until RMSD converges and hopefully the number of atoms increases.

Finally, we try to extend the obtained transformation and alignment to the atoms outside the aligned secondary structures, while maintaining the continuity constraint. More precisely, we examine all atoms belonging to loop regions in between corresponding secondary structures to check whether they can be transformed into each other within a given distance threshold value. By doing so, typically a considerable fraction of atoms is added to the output.

4. RESULTS

As already mentioned, the index-based comparison procedure produces a rough list of hypotheses of similarities of the target model with the complete database, and the fine alignment by DP is next performed on each pair of the polypeptide chains from the list. The result of a query consists of a list of matching proteins in decreasing order of score. The initial part of the output for a test case is reported in Table 1 sorted according to the *Score* generated by the DP algorithm. Like in other fold comparison servers, an absolute cut-off value is not assigned in order to decide what can be considered similar and what is not. At present, the web-server allows one to display a maximum of 9,999 output chains, which represents a consistent proportion of the total entries of the Protein Data Bank and, in any case, quite a large number to be examined manually.

To assess the performance of our approach, we conducted a first set of experiments using the SCOP classification (Murzin *et al.*, 1995) as standard-of-truth. The goal was to determine whether, for a given model or query protein, our server would retrieve proteins from the same fold, according to SCOP, as the query protein. Three representative proteins present in the PDB, belonging to well-known protein families, were used for the test. The selected proteins were i) triose phosphate isomerase, chain A (1tim), an α/β

TABLE 1. AN EXAMPLE OF OUTPUT FOR PROTEIN 101m[a]

Rank	*Protein*	SS_{al}	*Score*	*Name*	N_A	*RMSD(Å)*
1	1abs .	8	172	OXYGEN STORAGE	154	0.3
2	2mgf .	8	170	OXYGEN STORAGE	154	0.4
3	2mga .	8	170	OXYGEN STORAGE	154	0.3
4	1mtk .	8	170	OXYGEN STORAGE	154	0.1
5	1mti .	8	170	OXYGEN STORAGE	154	0.2
6	1mls .	8	170	OXYGEN STORAGE	154	0.3
7	1mlr .	8	170	OXYGEN STORAGE	154	0.2
⋮	⋮	⋮	⋮	⋮	⋮	⋮
604	1f5p E	7	8	OXYGEN STORAGE/TRANSPORT	128	2.7
605	1lfl D	5	8	OXYGEN STORAGE/TRANSPORT	130	3.1
606	1gcw A	5	8	OXYGEN STORAGE/TRANSPORT	133	2.6
607	1ny9 A	5	8	TRANSCRIPTION	31	2.9
608	1aoi H	4	8	NUCLEOSOME CORE/DNA	49	2.9
609	1hbr A	6	7	OXYGEN STORAGE/TRANSPORT	130	2.9
610	1a0y C	6	7	OXYGEN TRANSPORT	126	3

[a]The results are displayed as a list of proteins sorted according to the *Score* value with SS_{al} the number of aligned SSEs, *Score*, the similarity measure computed by DP, N_A the number of atoms aligned, and the RMSD computed between corresponding atoms.

TABLE 2. ACCURACY OF THE RESULTS FOR PROTEINS 1tim CHAIN A, 110m, AND 1rbp[a]

Triose phosphate isomerase (1tim A)		*Myoglobin (110m)*		*Retinol binding protein (1rbp)*	
Number of items	*Accuracy*	*Number of items*	*Accuracy*	*Number of items*	*Accuracy*
100	100.00%	100	100.00%	30	100.00%
200	100.00%	200	100.00%	50	93.88%
300	100.00%	300	100.00%	100	91.92%
400	100.00%	400	100.00%	150	81.76%
500	99.58%	500	100.00%	200	71.79%
600	99.29%	600	100.00%	250	63.27%
700	97.87%	700	94.89%	300	55.09%
800	96.01%	800	87.32%		
900	92.19%	900	79.15%		
1,000	88.88%	1,000	74.21%		
1,100	85.00%	1,200	64.21%		
1,200	80.42%				
1,500	70.38%				
2,000	59.24%				

[a]A number in the first column of each table refers to the top considered items in the output list. The accuracy is computed as the number of correctly classified proteins over the number of considered items.

protein belonging to the TIM barrel fold; ii) myoglobin (110m), an all-α protein fold; and iii) retinol binding protein (1rbp), a representative of an all-β proteins. The results using each protein as a query are summarized in Table 2. This table contains the percentage of correct results in the top n items of the output, for various values of n. A result is considered correct or true positive if the protein reported as similar is classified by SCOP in the same fold as the query protein.

Since the search is performed against all the PDB and not a selected subset, the number of significant outputs is different in the three cases, reflecting the different number of homologues that populates the

TABLE 3. COMPARISON OF STRUCTURE ALIGNMENT FOR SOME "DIFFICULT" CASES (SEE SHINDYALOV AND BOURNE, 1998) OBTAINED BY PROUST, CE, DALI AND VAST

		PROuST		*CE*		*DALI*		
Prot 1 (size)	*Prot 2 (size)*	n_a/*rmsd*	*Time (sec)*	n_a/*rmsd*	*Time (sec)*	n_a/*rmsd*	*Time (sec)*	*VAST* n_a/*rmsd*
1clc (639)	1hoe (74)	61/3.1	0.6	66/3.1	10.5	66/3.4	24.4	45/2.5
3hla B (99)	2rhe (299)	71/3.1	0.4	85/3.5	1.3	75/3.0	15.8	58/2.3
2aza A (276)	1paz (120)	74/2.8	0.5	85/2.9	1.6	81/2.5	13.9	70/2.1
2sim (799)	1nsb A (466)	263/3.3	1.3	276/2.9	50.7	292/3.3	388.2	296/3.9
1tie (272)	4fgf (217)	101/3.1	0.4	115/2.8	2.3	114/3.1	29.7	76/1.6

database. In the case of 1tim, more than 400 chains are correctly recognized before a protein with a different fold (according to the SCOP classification) is found. Besides, the top 120 polypeptide chains of the list belong to triose phosphate isomerases from different sources or mutants, whilst position 121 is occupied by a different protein sharing the same folding pattern (indole-3-glycerole phosphate synthase, 1j5t). From that position, enzymes that share a TIM barrel fold alternate with few other triose phosphate isomerases. A similar situation holds for 110m, with all globins at the top 618 positions. In the case of 1rbp, since only a relatively limited number of chains with similar fold are present in PDB, the first false positive that, according to SCOP, does not belong to the same fold is encountered at position 31. This corresponds to the crystal structure of avidin, a β-barrel fold that presents an overall similarity with 1rbp. All the other proteins that are considered incorrect in Table 2 represent other β-barrel folds (streptavidin-like, or GFP, or immunoglobulin-like): they are considered to belong to other protein families, since their topology is different from that of 1rbp. Nevertheless, a significant portion of their β-barrel superimposes quite nicely with that of 1rbp. This demonstrates the flexibility of our method, which considers only spatial relationships among secondary structural elements, neglecting the connections among them, and is able to detect a partial similarity between two proteins of totally different size. We have done several other tests on proteins of different classes and sizes and observed the same behavior; our results are typically consistent with SCOP with few false positive intermixed with true positive at relatively high positions in the output list.

Besides producing a list of similarities, the server can compute the best transformation matrix that superimposes each pair of chains. It gives the RMSD among corresponding C_α atoms, and the coordinates of the two superimposed proteins can be downloaded. A second set of experiments was done to compare the results of our pairwise protein structure superposition with those obtained by the servers CE, DALI, and VAST on pairs of proteins that are classified as "difficult cases" by Shindyalov *et al.* (1998). For the selected pairs of proteins, Table 3 shows the number of aligned atoms and the corresponding RMSD with each of the four methods, PROuST, CE, DALI, and VAST. The execution times of all four programs are also shown in the last column. More details on how the execution times are obtained are given later.

In these tests (Table 3) PROuST behaves reasonably well and typically is much faster than other approaches. The RMSD between the pairs of models is, in general, comparable even if not identical.

4.1. Time performance

One of the main advantages of our method is that the comparison and alignment of the query protein structure with the entire database can be performed on-line, since the time consuming step of the procedure, i.e., the building of the hash table, is done off-line in a preprocessing stage. With the current size of the PDB of more than 27,000 structures, the hash table construction would take a few days on a standard computer. Thus, we have resorted to a distributed implementation on a grid computational infrastructure (Ferrari *et al.*, 2003); using four computational nodes of a grid, the overall computing time for the table construction is approximately 28 hours.

Once the table has been built, searching the table for structures similar to a query protein is really fast. The execution times of the search are generally in the range of seconds or a few minutes depending on the query protein size. For instance, the total time required to comparing protein 110m, sperm whale

myoglobin, consisting of eight helices, against the entire database is 18 seconds. (In fact, of the 18 seconds, only 2.1 are used for searching, while the remaining are used for the grid initialization and authentication phases.) For the larger protein 1tim chain A, containing 12 helices and 8 strands, the total time goes up to four minutes.

The search results of the indexing method from a database of 23,000 proteins of the PDB are available at the website *www.angela.dei.unipd.it/PROuST*. Note that a small percentage of all proteins from the PDB could not be included in our database, because of inaccurate data due to discrepancies between PDB secondary structure entries, the atomic coordinates, and the offset numbers of secondary structures.

The PROuST server provides also an alignment of the query protein with each of the proteins in the output list. This step using DP and rigid body transformation is also performed on-line very efficiently, since the execution time of the pairwise comparison is generally seconds or fractions of seconds (see Table 3). The results of the alignment can be visualized with Rasmol or the superimposed coordinates downloaded. At the present, the web page does not show the list of output proteins reranked according to the results of the alignment. The development of the new website with this additional feature is underway.

A systematic comparison of the time performance of different servers for similarity retrieval is somewhat difficult for several reasons. First of all, some servers use only a subset of representative proteins and not the entire PDB. Furthermore, the results are often precomputed or provided by an e-mail in reply to a submitted query. Even if the execution times were available, they would be have been achieved on different computational platforms (supercomputers, clusters of PC, etc.). In the recent paper by Novotny *et al.* (2004) the time performance of 11 servers is compared by using as measure the time elapsed between submission of a query and receipt of the results. While this is certainly an important criterion, especially from a user's point of view, here we prefer to consider the actual execution times. Thus, we limit the analysis of the time requirements to the pairwise comparison only, for which the computer programs can be downloaded and run on the same computer.

The execution times of PROuST, CE, and DALI for the alignment and superposition of few pairs of proteins are given in Table 3. They have been obtained by running PROuST, CE, and DALI on the same standard PC. For PROuST, the execution time reported in the table is that of the DP algorithm followed by the protein superposition procedure. Unlike the actual implementation of PROuST, the DP algorithm in this experiment computes the score matrix from scratch without using the information already stored in the hash table. In other words, it does not assume any preprocessing of the two proteins but creates a temporary hash table where it inserts only the two proteins to be compared. The CE and DALI pairwise comparison programs have been downloaded from their respective websites. Since the time execution reported by DALI refers to the algorithm applied to all chains of the input proteins while the other two programs consider only one chain per protein, the execution time of DALI is divided by the number of chain comparisons under the assumption that the time is approximately the same on all pairs of chains.

From all our experiments, the time performance of pairwise comparison in PROuST is typically superior to that of other programs, often by a large factor. Since our overall approach to similarity retrieval is especially efficient in the first step of the indexing search, which allows to drastically reduce the number of pairwise comparisons needed, it is reasonable to conclude our method performs extremely well in terms of time complexity.

5. CONCLUSIONS

We have presented a fast and accurate method for protein structure comparison based on indexing and dynamic programming. In our approach, a macromolecular complex is split into its component polypeptides, and the search over the entire database is performed chain by chain. In such a way, similarities can be found among quaternary structures that present a similar fold only in one subunit and not in the others. Owing to the algorithm that takes into account triplets of secondary structure elements, partial similarities can be found without imposing any a priori restriction in the size of the comparison, i.e., similarities among domains or portions of the search model with domains or portions of models in the data base. Moreover, only secondary structure elements, i.e., α-helices and β-strands, are considered, and loops or other undefined strands are somewhat neglected and considered only in the final superposition step. Since the latter are the parts of proteins that quite often present different conformations even among similar

molecules, this improves the performances of the search. It must be noted that, since our server compares a single molecule against the entire database, a very large number of obvious similarities can be found, in particular for proteins that are largely represented in the PDB: for example, all the mutants and complexes of the search protein, along with all the proteins of the same family. This disadvantage is compensated by the fact that the choice of a structure representative of a given fold sometimes does not allow an exhaustive comparison. For this reason, our server is particularly useful when the protein we are using as a search model is a potential new fold or does not belong to a well known family. In this case, the output will contain a limited number of significant entries and eventually allow the detection of relatively distant similarities. Finally, the method presented here can be used to determine in a very efficient way the superposition between two polypeptide chains in a stand-alone version. In fact, it does not require any knowledge about the amino acid sequence similarity, since only secondary structure elements are considered and aligned.

ACKNOWLEDGMENTS

Support for Guerra and Comin was provided in part by the Italian Ministry of University and Research under the FIRB Project "Enabling Platforms for High Performance Computational Grids in Scalable Virtual Organizations." Additional support for Guerra and support for Zanotti was provided by the Italian Ministry of University and Research under the FIRB Project "Bioinformatics for Genomics and Proteomics."

WEBSITES

- *www.rcsb.org/pdb*
- *www.ebi.ac.uk/dali*
- *www.cl.sdc.ede/ce.html*
- *www.biochem.ucl.ac.uk/orengo/ssap.html*
- *www.ncbi.nlm.nih.gov/Structure/VAST/vast.html*
- *www.scop.mrc-lmb.cam.ac.uk/scop*

REFERENCES

Abagyan, R.A., and Maiorov, V.N. 1989. An automatic search for similar spatial arrangements of α helices and β-strands in globular proteins. *J. Mol. Struct. Dyn.* 6(5), 1045–1060.

Akutsu, T. 1996. Protein structure alignment using dynamic programming and iterative improvement. *IEICE Trans. Inf. Syst.* E78-D, 0, 1–8.

Alesker, A., Nussinov, R., and Wolfson, H.J. 1996. Detection of non-topological motifs in protein structures. *Protein Eng.* 9(5), 1103–1119.

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., and Bourne, P.E. 2000. The Protein Data Bank. *Nucl. Acids Res.* 28, 235–242.

Bourne, P.E., and Weisseig, H., eds. 2003. *Structural Bioinformatics*, Wiley-Liss, New York.

Brown, N.P., Orengo, C.A., and Taylor, W.R. 1996. A protein structure comparison methodology. *Comp. Chem.* 27, 359–380.

Ferrari. C., and Guerra, C. 2003. Geometric methods for protein structure comparison. Protein structure analysis and design, *in* C. Guerra and S. Istrail, eds., *Lecture Notes in Bioinformatics*, Springer-Verlag, New York.

Ferrari, C., Guerra, C., and Zanotti, G. 2003. A Grid-aware approach to protein structure comparison. *J. Parallel and Distributed Computing*, Special Issue on "High Performance Computational Biology."

Fischer, D., Tsai, C.J., Nussinov, R., and Wolfson, H. 1995. A 3D sequence-independent representation of the protein data bank. *Protein Eng.* 8, 981–997.

Foster, I., and Kesselman, C., eds. 1999. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco.

Gerstein, M. 1992. A resolution-sensitive procedure for comparing protein surfaces and its application to the comparison of antigen-combining sites. *Acta Cryst.* A8, 271–276.

Gerstein, M., and Levitt, M. 1998. Comprehensive assessment of automatic structural alignment against a manual standard, the SCOP classification of proteins. *Protein Sci.* 7, 445–456.

Gibrat, J.F., Madej, T., and Bryant, S.H. 1996. Surprising similarities in structure comparison. *Curr. Opin. Struct. Biol.* 6(3), 377–385.

Grindley, H.M., Artymiuk, P.J., Rice, D.W., and Willett, P. 1993. Identification of tertiary structure resemblance in proteins using a maxima common subgraph isomorphism algorithm. *J. Mol. Biol.* 229, 707–721.

Guerra, C., Lonardi, S., and Zanotti, G. 2002. Analysis of proteins secondary structures using indexing techniques. *IEEE Proc. 1st Int. Symposium on 3D Data Processing Visualization and Transmission*, 812–821.

Holm, L., and Sander, C. 1995. 3D-Lookup: Fast protein structure database searches at 90% reliability. *Proc. 3rd Int. Conf. on Intell. Sys. for Mol. Biol.* 179–187.

Holm, L., and Sander, C. 1996. Mapping the protein universe. *Science* 273, 595–602.

Holm, L., and Sander, C. 1996b. The FSSP database: Fold classification based on structure–structure alignment of proteins. *Nucl. Acids Res.* 24(1), 206–209.

Horn, B.K.P. 1987. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am.* 4(4), 629–642.

Lamdan, Y., Schwartz, J.T., and Wolfson, H.J. 1990. Affine invariant model-based object recognition. *IEEE Trans. on Robotics and Automation*, 578–589.

Lancia, G., Carr, R., Walenz, B., and Istrail, S. 2001. Optimal PDB structure alignments: A branch-and-cut algorithm for the maximum contact map overlap problem. *Proc. 5th ACM Research in Computational Biology*, 193–202.

Lemmen, C., Lengauer, T. 2000. Computational methods for the structural alignment of molecules. *J. Computer-Aided Molecular Design* 14, 215–232.

Murzin, A.G., Brenner, S.E., Hubbard, T., and Chothia, C. 1995. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247, 536–540.

Novotny, M., Madsen, D., and Kleywegt, G.J. 2004. Evaluation of protein fold comparison servers. *Proteins* 54, 260–270.

Overington, J.P., Zhu, Z.Y., Sali, A., Johnson, M.S., Sowdhamini, R., Louie, G.V., and Blundell, T.L. 1993. Molecular recognition in protein families: A database of aligned three-dimensional structures of related proteins. *Biochem. Soc. Trans.* 21(3), 597–604.

Platt, D.E., Guerra, C., Rigoutsos, I., and Zanotti, G. 2003. Global secondary structure packing angle bias in proteins. *Proteins: Struct. Funct. Genet.* 53, 252–261.

Sali, A., Blundell, P.T.L. 1990. Definition of general topological equivalence in protein structures. A procedure involving comparison of properties and relations through simulated annealing and dynamic programming. *J. Mol. Biol.* 212(2), 403–428.

Shindyalov, I.N., and Bourne, P.E. 1998. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Eng.* 11(9), 739–747.

Singh, A.P., and Brutlag, D.L. 1997. Hierarchical protein structure superposition using both secondary structures and atomic representations. *Proc. 5th Int. Conf. Intell. Sys. Mol. Biol.* 284–293.

Taylor, W.R. 1999. Protein structure comparison using iterated double dynamic programming. *Protein Sci.* 8, 654–665.

Taylor, W.R., and Orengo, C.A. 1989. Protein structure alignment. *J. Mol. Biol.* 208(1), 1–22.

Address correspondence to:
Concettina Guerra
Department of Information Engineering
University of Padova
via Gradenigo 6A
35131 Padova, Italy

E-mail: guerra@dei.unipd.it