Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 03-001

Pairwise Protein Structure Alignment Based on an Orientation-Independent Representation of the Backbone Geometry

Jieping Ye, Ravi Janardan, and Songtao Liu

January 14, 2003

Pairwise protein structure alignment based on an orientation-independent representation of the backbone geometry

Jieping Ye* Ravi Janardan* Songtao Liu†

June 3, 2003

Abstract

Determining structural similarities between proteins is an important problem since it can help identify functional and evolutionary relationships. In this paper, an algorithm is proposed to align two protein structures. Given the protein backbones, the algorithm finds a rigid motion of one backbone onto the other such that large substructures are matched. The algorithm uses a representation of the backbones that is independent of their relative orientations in space and applies dynamic programming to this representation to compute an initial alignment, which is then refined iteratively. Experiments indicate that the algorithm is competitive with two well-known algorithms, namely DALI [12] and LOCK [19].

1 Introduction

Three-dimensional (3D) structure plays a central role in research directed towards understanding evolutionary and functional relationships among proteins. It is well-known that structural information is better conserved than sequence information in the evolution of proteins [13], hence can be used in the construction of phylogenetic trees [15]. Protein-protein interactions are governed in large part by the shape, location, and composition of the so-called active sites [2]. The assignment of proteins to fold families is accomplished via structural analysis [14, 17]. The need for effective structural analysis techniques has increased with the rapid growth in the number of 3D structures available now in the Protein Data Bank (PDB) [1].

A key problem in protein structure analysis is pairwise protein structure alignment: Given the C_α backbones of two proteins, the goal is to find a rigid motion of one backbone onto the other such that large, contiguous regions of the backbones are matched. (A formal definition is given later in Section 2.). Besides the applications mentioned above, pairwise structural alignment is also a key component of algorithms that seek to align multiple protein structures in order to find a core structure that captures essential structural information for the whole set [4].

*Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN 55455, U.S.A. {jieping,janardan}@cs.umn.edu . This effort is sponsored, in part, by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

†Department of Mathematics, University of Alberta, Edmonton, Alberta T6G 2G1, Canada. songtao@math.ualberta.ca .

Pairwise alignment of protein structures has been a subject of much research. Holm and Sander [12, 13] propose an algorithm, called DALI, which works with the distance matrices obtained from the interatomic (C_α-C_α) distances on each backbone. The algorithm decomposes each matrix into submatrices that represent so-called elementary contact patterns, aligns a pair of patterns from the two matrices, and iteratively builds a connected chain of such pairwise aligned patterns using a Monte Carlo method to optimize the similarity score. Singh and Brutlag [19] give an algorithm, called LOCK, which represents the secondary structure elements (α-helices and β-strands) as vectors and computes a local alignment of these via dynamic programming. A suite of seven different scoring functions is used to score the alignment. This yields an initial superposition, which is then improved iteratively by operating at the atomic level, on the 3D coordinates of the C_α atoms, until the root-mean squared deviation (RMSD) [6] of the aligned atoms converges. Chew *et. al.* [3] propose an approach which represents the backbone by a chain of unit-vectors. These vectors are then translated to the origin, yielding a representation of the protein as a set of points on the unit-sphere. To compute a structural alignment, they first compute a small set of shifts in sequence space that are likely to bring 3D structures into correspondence. For each such shift, they compute substructures that are contiguous on the backbone and for which there is a 3D alignment, and then combine such substructures into large (possibly non-contiguous) substructures called domains. They use a variant of the RMSD measure, called unit-vector RMSD, which is robust to outliers.

A non-exhaustive list of other related work includes the combinatorial extension (CE) method of Shindyalov and Bourne [18], a method based on geometric hashing by Fischer, Nussinov, and Wolfson [8], the method of Falicov and Cohen [7] which attempts to minimize the so-called soap-bubble surface area between the backbones, and the double-dynamic programming technique of Orengo and Taylor [20, 21].

In this paper, we propose a new approach to the pairwise structural alignment problem. Our algorithm computes a representation of each backbone in terms of certain angles defined by consecutive C_α-C_α bonds. (This representation is related to the pseudodihedral angle used in [5]; as noted in [5], this angle is highly correlated to the identity of the central pair of amino acids.) The resulting backbone representation which consists of a sequence of triples of angles, is independent of the relative orientation of the two proteins in space. We apply dynamic programming on this representation (not on the protein sequence) and compute an initial alignment of the two proteins. We then refine this alignment iteratively. Our algorithm takes time that is quadratic in the sum of the lengths of the two proteins. We have implemented this alignment and tested it against LOCK and DALI. We have found that our algorithm is quite competitive with these algorithms. For proteins that are closely related, all three algorithms find large structural matches. In most cases, the number of matches found by our algorithm is close to that of DALI and fairly larger than that of LOCK; correspondingly our RMSD value is higher than LOCK's and lower than DALI's. For unrelated or distantly-related proteins, none of the methods consistently outperforms the others.

The rest of the paper is organized as follows. Section 2 gives an overview of our method. Section 3 describes in detail the five steps that comprise our algorithm. We report on experimental results in Section 4 and conclude in Section 5.

2 Overview of the approach

Let A and B be the two proteins under consideration, each represented by a chain of C_α atoms (the backbone) in R^3. (As is customary [19, 12], we consider only the backbone, not the amino acid

residues themselves.) Assuming B is fixed in space, we would like to find a rigid motion (translation and rotation) of A that aligns large substructures of A and B. Specifically, we would like to find substructures A' of A and B' of B, a bijection between atoms of A' and B', which preserves their order on the backbones, and a rigid motion of A onto B such that $|A'| = |B'|$ is as large as possible and the Euclidean distance between atom pairs in the bijection is at most a user-specified threshold ε. We present a heuristic for this problem which is very competitive with other known methods.

The key to our approach is a geometric representation of the backbone structure that is independent of the relative orientations of the proteins A and B. Let the C_α atoms of A and B be numbered 1 through n and 1 through m, respectively, in order along the backbone. In effect, the geometric representation of A (resp. B) may be viewed as a collection $\{A_i\}_{i=2}^{n-2}$ (resp. $\{B_i\}_{i=2}^{m-2}$) of points in R^3, where each point A_i (resp. B_i) captures the local geometry of the (virtual) bond joining the ith and $(i+1)$th C_α atoms of A (resp. B) in relation to the preceding and succeeding bond on the backbone. A major advantage of this representation is that it is independent of the relative positions and orientations of the backbones. Indeed, as we will see in Section 3, the question of whether there exists a rigid motion of A that aligns it to B can be framed as the question of determining the similarity of the (static) point-sets representing A and B. In particular, A can be aligned exactly with B if and only if the sets $\{A_i\}_{i=2}^{n-2}$ and $\{B_i\}_{i=2}^{m-2}$ are identical.

In practice, of course, an exact alignment is unlikely to exist; instead the goal is to align approximately one or more substructures of A to substructures of B. Towards this end, we use dynamic programming to compute an optimal global alignment of the sequences $A_2, \cdots, A_{n-2}$ and $B_2, \cdots, B_{m-2}$, using the Euclidean distance between the A_i's and B_j's as the scoring function. The resulting alignment consists of runs of matches between elements of the two sequences, with gaps in between. (See Figure 1.)

However, it may not be possible to simultaneously align, in 3D, all of the substructures corresponding to the different runs of matches found in the sequence alignment. This is because the transformation matrix that aligns the structures corresponding to one run may not be consistent with the matrix for another run. For instance, in Figure 1, the matrix that aligns atoms 1–5 in A with atoms 1–5 in B may be different from the one that aligns atoms 9–13 in A with atoms 5–9 in B.

To overcome this problem, we compute a subset of the runs such that the transformation matrices for any two runs in the subset are similar (as measured by the Frobenius norm [10]) and the total number of atoms in the runs is large. We show how this can be reduced to the problem of finding a large weighted clique in an undirected graph and solve this using a greedy heuristic. For the subset of consistent runs thus found, we compute a transformation matrix that aligns the corresponding substructures. This alignment is further improved via dynamic programming on the coordinates of the C_α atoms, during which pairs that are not within the threshold of ε are discarded. This yields an initial alignment of the proteins. The closeness of the overall alignment is measured by the RMSD [6].

The algorithm then enters an iterative refinement phase. From the matches in the initial alignment, a new transformation matrix is derived. This is used to re-align the proteins via dynamic programming, as above. The process is repeated until the RMSD converges.

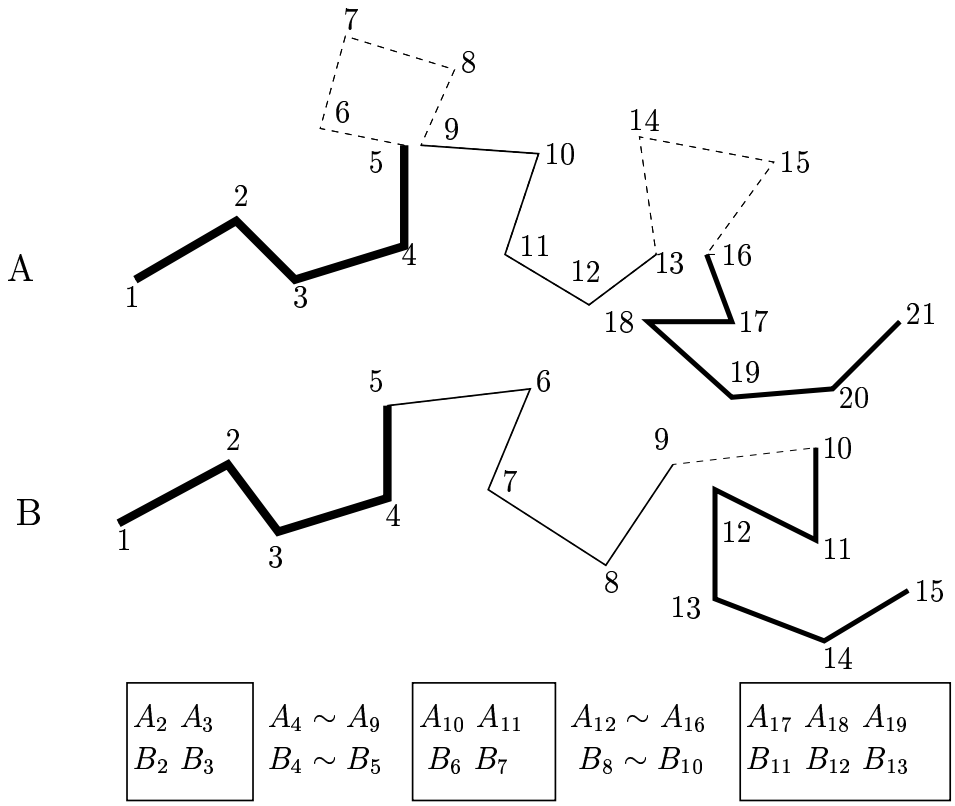

Figure 1: A local alignment of the sequences $A_2, \cdots, A_{19}$ and $B_2, \cdots, B_{13}$ and the corresponding matched substructures of proteins A and B, shown in solid lines of varying thickness. Dashed lines denote unmatched substructures.

---

3 Details of the algorithm

3.1 Step 1: Computing a local geometric representation

Recall that protein A consists of n C_α atoms, number $1, \cdots, n$ along the backbone. We define a sequence of vectors $\mathbf{a}_i$, $1 \leq i \leq n-1$ on the backbone, where $\mathbf{a}_i$ is the vector from the ith C_α atom to the $(i+1)$th C_α atom. Each $\mathbf{a}_i$ has the same length as the corresponding (virtual) bond; this is about 3.8 Angstroms. We represent the geometry of the backbone in the vicinity of $\mathbf{a}_i$, $2 \leq i \leq n-2$ by a triple of angles $A_i = (\alpha_i^A, \beta_i^A, \gamma_i^A)$. Here $\alpha_i^A \in [0, \pi]$ is the angle between $-\mathbf{a}_{i-1}$ and $\mathbf{a}_i$ and $\beta_i^A \in [0, \pi]$ is the angle between $-\mathbf{a}_i$ and $\mathbf{a}_{i+1}$. Both of these angles can be computed via dot products of the corresponding vectors. We define γ_i^A as follows: Consider the vectors $\mathbf{n}_i = -\mathbf{a}_{i-1} \times \mathbf{a}_i$ and $\mathbf{n}_{i+1} = -\mathbf{a}_i \times \mathbf{a}_{i+1}$. Note that $\mathbf{a}_i$ is perpendicular to both $\mathbf{n}_i$ and $\mathbf{n}_{i+1}$. Let $\theta \in [0, 2\pi]$ be the angle between $\mathbf{n}_i$ and $\mathbf{n}_{i+1}$. Then $\gamma_i^A = \theta$ if $\mathbf{n}_i \times \mathbf{n}_{i+1}$ has the same direction as $\mathbf{a}_i$; otherwise $\gamma_i^A = 2\pi - \theta$. In other words, γ_i^A is the dihedral angle between the plane containing $-\mathbf{a}_{i-1}$ and $\mathbf{a}_i$ and the plane containing $-\mathbf{a}_i$ and $\mathbf{a}_{i+1}$ (Figure 2 illustrates these angles.) Thus, $A_i = (\alpha_i^A, \beta_i^A, \gamma_i^A)$ captures the orientation of $\mathbf{a}_i$ relative to its predecessor $\mathbf{a}_{i-1}$ and its successor $\mathbf{a}_{i+1}$. The set $\{A_i\}_{i=2}^{n-2}$ is the local geometric representation of the backbone A. We can define similarly a sequence of vectors $\mathbf{b}_i$, $2 \leq i \leq m-2$, for the backbone of protein B, and a

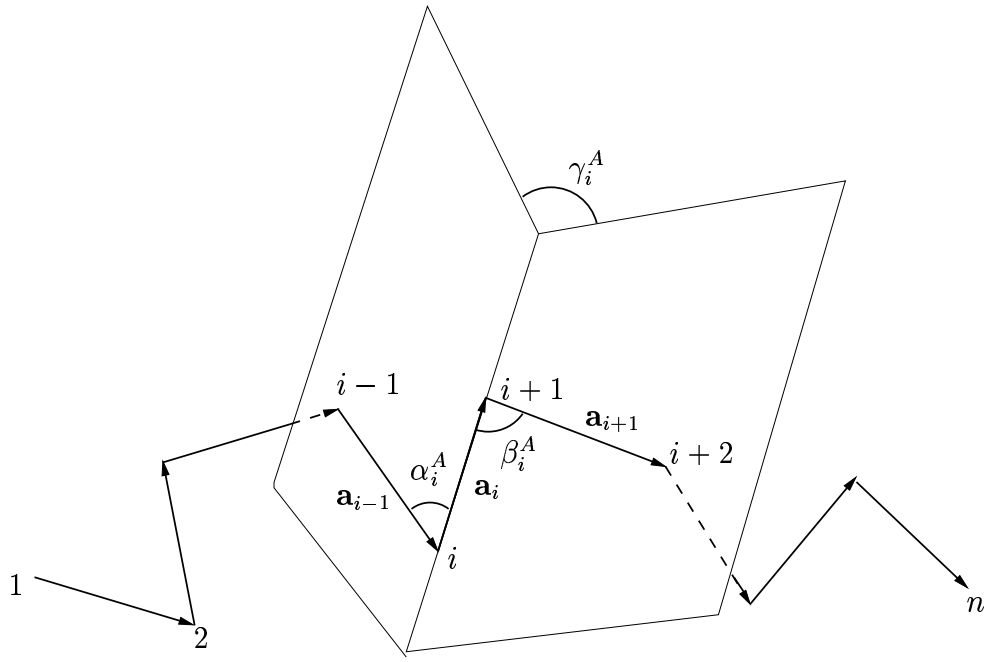

Figure 2: Vector representation of backbone of protein A and associated angles. The triple $A_i = (\alpha_i^A, \beta_i^A, \gamma_i^A)$ is associated with the vector $\mathbf{a}_i$, $2 \leq i \leq n-2$.

---

triple $B_i = (\alpha_i^B, \beta_i^B, \gamma_i^B)$ for $\mathbf{b}_i$. The local geometric representation of B is then the set $\{B_i\}_{i=2}^{m-2}$.

The following lemma establishes the orientation independence of this representation.

Lemma 3.1 *Let A' and B' be substructures of A and B, respectively, where A' contains the pth through $(p+\ell)$th C_α atoms of A and B' contains the qth through $(q+\ell)$th C_α atoms of B, $\ell \geq 3$, $1 \leq p \leq n-\ell$, and $1 \leq q \leq m-\ell$. Then there is a rigid motion (translation and rotation) of A to B which aligns exactly the $(p+j)$th C_α atom of A with the $(q+j)$th C_α atom of B if and only if $||A_{p+i} - B_{q+i}||_2 = 0$, where $0 \leq j \leq \ell$ and $1 \leq i \leq \ell - 2$. (Here $||A_{p+i} - B_{q+i}||_2$ is the Euclidean distance between A_{p+i} and B_{q+i} when they are viewed as points in R^3.)*

Proof ($\Rightarrow$) Obvious

($\Leftarrow$) Consider first the case $\ell = 3$. W.l.o.g., we may assume that the qth, $(q+1)$th, and $(q+2)$th C_α atoms of B lie in the xy-plane, with the $(q+1)$th atom at the origin, and the qth atom on the positive x-axis. Clearly, we can transform A so that the pth, $(p+1)$th, and $(p+2)$th C_α atoms of A also lie on the xy-plane, with the $(p+1)$th atom at the origin and the pth atom on the positive x-axis. Since $\mathbf{a}_p$ and $\mathbf{b}_q$ have the same length, $\mathbf{a}_{p+1}$ and $\mathbf{b}_{q+1}$ have the same length, and $\alpha_{p+1}^A = \alpha_{q+1}^B$, it follows that the pth and qth C_α atoms coincide and $(p+2)$th and $(q+2)$th C_α atoms coincide.

Now, $\gamma_{p+1}^A = \gamma_{q+1}^B$ implies that the $(p+3)$th and $(q+3)$th C_α atoms of A and B, respectively, lie in a common half-plane. (This half-plane makes an angle of γ_{p+1}^A with the xy-plane and its intersection with the xy-plane is the line containing the (coincident) vectors $\mathbf{a}_{p+1}$ and $\mathbf{b}_{q+1}$.) Then since $\beta_{p+1}^A = \beta_{q+1}^B$ and $\mathbf{a}_{p+2}$ and $\mathbf{b}_{q+2}$ have the same length, it follows that the $(p+3)$th and $(q+3)$th C_α atoms coincide. This completes the proof for the case $\ell = 3$.

Next, let $\ell > 3$. The argument above shows that once the first three pairs of atoms of A' and B' coincide, then the fourth pair also coincides. But now since the second through fourth pairs of atoms of A' and B' coincide, by a similar argument, the fifth pair must also coincide. This argument can be repeated until the $(p+\ell)$th and $(q+\ell)$th atoms are shown to coincide. ∎

3.2 Step 2: Aligning the local representation

As mentioned in Section 2, a perfect alignment between A and B (Lemma 3.1) is unlikely to exist. Instead we try to align substructures of A and B. Towards this end, we treat the sets $\{A_i\}_{i=2}^{n-2}$ and $\{B_i\}_{i=2}^{m-2}$ as sequences $A_2, \cdots, A_{n-2}$ and $B_2, \cdots, B_{m-2}$ and compute an optimal global alignment for them using dynamic programming [11]. This alignment is produced using a global alignment method with zero end-gap penalties. Internal gaps are penalized using an affine gap penalty of the form: $h(k) = a + bk$, where k is the length of the gap, and a and b are gap opening and extension penalties, respectively. We score the alignment using a scoring function S_0, defined as :

$$S_0(A_i, B_j) = K - d(A_i, B_j). \tag{1}$$

Here $d(A_i, B_j)$ is the Euclidean distance between the points $A_i = (\alpha_i^A, \beta_i^A, \gamma_i^A)$ and $B_j = (\alpha_j^B, \beta_j^B, \gamma_j^B)$. (A similar function is used in Dali [12].) Our goal is to compute global alignment whose total score is maximum. It is tempting to write

$$d(A_i, B_j) = \left((\alpha_i^A - \alpha_j^B)^2 + (\beta_i^A - \beta_j^B)^2 + (\gamma_i^A - \gamma_j^B)^2\right)^{1/2} \tag{2}$$

However, if one of the angles γ_i^A and γ_j^B is close to zero and the other one is close to 2π, then A_i and B_j are actually close to each other in the γ-dimension, even though $|\gamma_i^A - \gamma_j^B|$ is large (i.e. close to 2π). Thus we modify $d(A_i, B_j)$ as follows:

$$d(A_i, B_j) = \left((\alpha_i^A - \alpha_j^B)^2 + (\beta_i^A - \beta_j^B)^2 + g(|\gamma_i^A - \gamma_j^B|)^2\right)^{1/2} \tag{3}$$

where $g(x) = \min(2\pi - x, x)$.

What about the term K in the definition of S_0 and the gap opening and extension penalties a and b? We have done several experiments and found that $K = 1.4$, $a = 0.2$, and $b = 0.2$ work very well, and this is what we use throughout.

3.3 Step 3: Computing consistent runs of alignments

The alignment computed in step 2 yields runs of matched elements of $A_2, \cdots, A_{n-2}$ and $B_2, \cdots, B_{m-2}$, interspersed with gaps (see Figure 1). Let $R_1, \cdots, R_k$ be the runs and let $N_1, \cdots, N_k$ be their lengths. For each R_i, we can compute a transformation matrix $T_i = (T_i^t, T_i^r)$ that aligns the substructures of A and B corresponding to R_i such that the RMSD is minimized. Here T_i^t is the translation matrix and T_i^r is the rotation matrix. T_i can be computed using the Singular Value Decomposition (SVD) [10, 16]. However, the matrices $T_1, T_2, \cdots, T_k$ will not necessarily be mutually consistent, in the sense that the transformation specified by one matrix T_i may "interfere" with that specified by another matrix T_j, so that it may not be possible to align simultaneously the structures corresponding to all the runs $R_1, \cdots, R_k$.

Therefore, in this step, we compute a subset of the runs such that the total number of atoms in the runs is as large as possible and the transformation matrices for all the runs are "similar".

Formally, we wish to compute a subset I of $\{1, \cdots, k\}$ such that $\sum_{i \in I} |R_i|$ is as large as possible and for all $i, j \in I$, $||T_i^t - T_j^t||_F < \tau^t$ and $||T_i^r - T_j^r||_F < \tau^r$. Here τ^t and τ^r are similarity thresholds and $||C||_F = (\sum_{i=1}^{3} \sum_{j=1}^{3} c_{ij}^2)^{1/2}$ is the Frobenius norm of matrix $C = (c_{ij})$ [10].

To compute I, we first build an undirected graph $G = (V, E)$, where $V = \{1, 2, \cdots, k\}$ and $E = \{(i, j) \in V \times V : ||T_i^t - T_j^t||_F < \tau^t \quad \text{and} \quad ||T_i^r - T_j^r||_F < \tau^r\}$. With each vertex i of G, we associate a weight $w_i = |R_i|$. Clearly, our problem now is equivalent to finding a clique (complete subgraph) of G whose total vertex weight is maximum. This problem is NP-hard [9], so we solve it approximately via a greedy heuristic, as follows. Among all the vertices of G, we find a vertex, v, such that the total weight of v and all its neighbors in G is maximum. We add v to an initially empty list L, which accumulates the growing clique. We then repeat the above step on the subgraph, G', of G induced by the neighbors of v. That is, among the vertices of G', we find a vertex, v', such that the total weight of v' and its neighbors in G' is maximum, and add v' to L. And so on, until at some point the current induced subgraph becomes empty. At this point, the vertices in L form a clique of G of large (but not necessarily maximum) total weight.

What should be the values of the similarity thresholds τ^t and τ^r? If τ^t and τ^r are very small, the matrix T_i and T_j are required to be very similar; this yields a subset of runs of small total size. If τ^t and τ^r are large then the matrices can be quite different, so the quality of the structural alignment is poor. We found via experiments that choosing τ^t between 10 and 40, and τ^r between 1 and 1.5 works very well. Our experiments also show that τ^r is much more sensitive than τ^t. The degree of similarity of two rotation matrices is determined by the threshold τ^r. As shown in Lemma 3.2 below, the expected value of the Frobenius norm of the difference of two rotation matrices whose elements are chosen randomly is about 2.4. Thus our choice of $\tau^r \in [1, 1.5]$ ensures that the associated matrices are all quite similar and not random.

Lemma 3.2 *The expected value of the Frobenius norm of the matrix that is the difference of two rotation matrices whose elements are chosen randomly is about* 2.4.

Proof Let C and D be two rotation matrices. Assume $C = M_z M_y M_x$, where M_x rotates by an angle c_x about the x-axis, M_y rotates by an angle c_y about the y-axis, and M_z rotates by an angle c_z about the z-axis. Similarly, let $D = N_z N_y N_x$, where N_x, N_y, and N_z rotate by angles d_x, d_y, and d_z about the x-, y-, and z-axes, respectively.

It follows that

$$
\begin{aligned}
C - D \;=\; & \begin{pmatrix} cos(c_z) & -sin(c_z) & 0 \\ sin(c_z) & cos(c_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(c_y) & 0 & sin(c_y) \\ 0 & 1 & 0 \\ -sin(c_y) & 0 & cos(c_y) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(c_x) & -sin(c_x) \\ 0 & sin(c_x) & cos(c_x) \end{pmatrix} \\
- \; & \begin{pmatrix} cos(d_z) & -sin(d_z) & 0 \\ sin(d_z) & cos(d_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(d_y) & 0 & sin(d_y) \\ 0 & 1 & 0 \\ -sin(d_y) & 0 & cos(d_y) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(d_x) & -sin(d_x) \\ 0 & sin(d_x) & cos(d_x) \end{pmatrix}.
\end{aligned}
$$

Since each of the six rotation angles is chosen at random from $[0, 2\pi]$, the expected value of $||C-D||_F$ then is

$$
\iiiiiint_{0 \le c_x, c_y, c_z, d_x, d_y, d_z \le 2\pi} \|C - D\|_F \quad \frac{1}{(2\pi)^6} \quad d(c_x) d(c_y) d(c_z) d(d_x) d(d_y) d(d_z).
$$

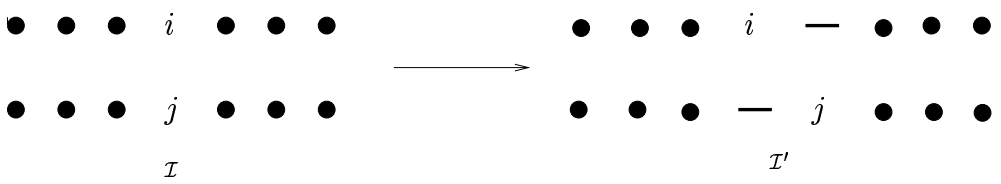

Figure 3: Local insertion of gaps in $\mathcal{I}$

Performing this calculation, via numerical integration, in MATLAB yields an expected value of about 2.3958. ∎

We also computed the expected value experimentally using MATLAB, by picking 40,000 pairs of rotation matrices whose elements are chosen at random and computing the Frobenius norm of the difference of each pair. This yielded an expected value of 2.3982.

3.4 Step 4: Computing an initial structural alignment

The previous step yields a subset of runs with pairwise similar transformation matrices. This means that there is now a single transformation matrix T, which can simultaneously align well the substructures corresponding to all runs in the subset. We compute T by SVD and use this to transform protein A and obtain an initial alignment with protein B.

Recall that we also require that each aligned pair of atoms from A and B to be within distance ε of each other. One possibility is to take the alignment provided by T and simply discard any pairs that do not meet the ε threshold. However, a large number of pairs that are close to the threshold could get discarded. A better strategy is to first re-align A and B by using dynamic programming on the coordinates just computed by the application of T. In the process, we also enforce the ε threshold automatically by choosing the gap penalty suitably.

Specifically, we compute a minimum-score global alignment of A and B [11]. We score the alignment of atom $i \in A$ and atom $j \in B$ using the Euclidean distance $d(i,j)$ between them. The score for matching a gap with an atom (i.e., the gap penalty) is a constant equal to $\varepsilon/2$. (In the experiments reported in Section 4, we used $\varepsilon = 8$ Angstroms.) At the end of this step, we get an initial structural alignment, $\mathcal{I}$, of A and B such that its score $s(\mathcal{I})$ is minimum. We then compute its RMSD.

We argue now that $\mathcal{I}$ satisfies the ε threshold. Suppose, for a contradiction, that, in $\mathcal{I}$, atoms $i \in A$ and $j \in B$ are aligned but that $d(i,j) > \varepsilon$. We modify $\mathcal{I}$ locally by inserting two gaps as shown in Figure 3 to get a new alignment $\mathcal{I}'$. Since the gap penalty is $\varepsilon/2$, $s(\mathcal{I}') = s(\mathcal{I}) - d(i,j) + \varepsilon/2 + \varepsilon/2 < s(\mathcal{I})$, which contradicts the optimality of $\mathcal{I}$.

3.5 Step 5: Refining the alignment iteratively

Our experiments have shown that the size of the alignment (i.e., the number of matched pairs) can be increased as follows. For the initial alignment computed in step 4, we recompute a new transformation matrix T' corresponding to all of the matched pairs in the initial alignment, again using SVD. We then use dynamic programming as in step 4 to obtain a new alignment, with a new RMSD value.

We iterate on the above transform-and-realign process until one of the two conditions is met: Either the number of iterations exceeds a specified limit or the absolute value of the difference in RMSD values of two successive alignments drops below a preset threshold η. (Recall that the RMSD of an alignment $\mathcal{I}$ containing matched pairs (i, j) of atoms, where $i \in A$ and $j \in B$ have coordinates (x_i, y_i, z_i) and (x_j, y_j, z_j), respectively, is $\left(\frac{1}{|\mathcal{I}|}\sum_{(i,j)\in\mathcal{I}}((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)\right)^{1/2}$.) In our experiments, we limited the number of iterations to ten and chose $\eta = 0.1$ Angstroms.

Complexity analysis: Step 1 clearly takes $O(m + n)$ time. Step 2 takes $O(mn)$ time, and produces an alignment of length at most $m + n$. In step 3, the computation of T_i, using the SVD, takes time $O(N_i)$, so the total time for all the T_i's is $O(\sum_{i=1}^{k} N_i) = O(m + n)$. Computing the clique takes the time proportional to the size of the graph G, which is $O((m + n)^2)$, since G has at most $m + n$ vertices. In step 4, the computation of the transformation matrix T, via the SVD, takes $O(m + n)$ time, and the dynamic programming takes an additional $O((m + n)^2)$ time. In step 5, each iteration takes $O((m + n)^2)$ time, similar to step 4. Since the number of iterations is bounded by a constant (ten in our experiments), the total time for step 5 is $O((m + n)^2)$. Thus, the overall running time of the algorithm is $O((m + n)^2)$.

4 Experimental results

We implemented our algorithm (using MATLAB) and tested it against two well-known structural alignment algorithms, namely LOCK [19] and DALI [12].

Our approcah was similar to that in [19] in that we aligned a query protein with each member of a set of proteins identified from the FSSP [13] and SCOP [17] databases as structural neighbors of the query.

The first query protein was the protein *Sperm whale myoglobin* (PDB ID: 1mbc), from the Globin family. The results of the alignment (i.e., the number of matched atoms and the RMSD value) are shown in Table 1. Note that the upper half of the Table 1 contains proteins that are in the same family as the query, hence closely-related, whereas the lower half contains proteins that are in different families, but still related. (Nearly the same set was used in [19] also.) The results of Table 1 are shown graphically in Figure 4. Note that here the proteins are listed in the x-axis in non-increasing order of the matches found by our method. The alignment of 1mbc with each of the proteins 5mbn, 2fal, and 1lia-A is shown in Figure 6 (left half) [1].

The second query protein that we used was *Thioredoxin-Reduced Form* (PDB ID: 3trx), from the Thioltransferase family. The results for this are shown in Table 2 and Figure 5, and Figure 6 (right half).

These results show that when the proteins in question are closely related, then all three methods (ours, LOCK, and DALI) are able to detect large structural matches. In most cases, the number of matches found by our method is close to that found by DALI and fairly larger than that found by LOCK. Correspondingly, the RMSD value of our method is quite a bit smaller than that of DALI and larger than that of LOCK. When the proteins in question are distantly-related or unrelated, then the number of matches is much smaller (as is to be expected) and none of these methods perform consistently better than the others. Overall, our method appears to be competitive with LOCK and DALI.

[1] Color version of all the figures may be viewed at http://www.cs.umn.edu/~jieping/Research.html

5 Conclusions

We have presented an iterative algorithm for pairwise protein structure alignment. The algorithm uses an angle-based representation of the protein backbones, which is independent of the relative orientation of the proteins in space. An initial alignment is found using dynamic programming (on the backbone representation) and a graph-based method and then refined iteratively, such that the number of matched C_α atoms is large and the distance between matched atoms is within a prescribed threshold. The heuristic has been implemented and found to be competitive with two other algorithms (LOCK and DALI), especially for closely-related proteins.

References

[1] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne: The Protein Data Bank. *Nucleic Acids Research*, 28, 2000, pp. 235-242.

[2] C. Branden, and J. Tooze. Introduction to Protein Structure, Garland Press. ISBN 0-8153-2305-0, 1999.

[3] L.P. Chew, K. Kedem, D.P. Huttenlocher, and J. Kleinberg. Fast detection of geometric substructure in proteins. *Journal of Computational Biology*, 6:(3-4), 1999, pp. 313–325.

[4] L.P. Chew, and K. Kedem. Finding the consensus shape of a protein family. *Proc. 18th Annual ACM Symposium on Computational Geometry*, Barcelona, Spain, June 2002, pp. 64–73.

[5] R.S. Dewitte, and E.I. Shakhnovich. Pseudodihedrals: Simplified protein backbone representation with knowledge-based energy. *Protein Science*, 3, 1994, pp. 1570–1581.

[6] I. Eidhammer, I. Jonassen, and W.R. Taylor. Structure Comparison and Structure Patterns. *Journal of Computational Biology*, 7(5), 2000, pp. 685–716.

[7] A. Falicov, and F.E. Cohen. A surface of minimum area metric for the structural comparison of proteins. *Journal of Molecular Biology*, 258, 1996, pp. 871–892.

[8] D. Fischer, R. Nussinov, and H. Wolfson. 3D Substructure Matching in Protein Molecules, *Proc. 3rd Intl. Symp. Combinatorial Pattern Matching*, Lecture Notes in Computer Science 644, Springer-Verlag, New York, 1992, pp. 136–150.

[9] M.R.Garey, and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H Freeman, San Francisco, CA, 1979.

[10] G.H. Golub, and C.F. Van Loan. Matrix Computations, John Hopkins University Press, 3rd edition, 1996.

[11] D. Gusfield. Algorithms on strings, trees, and sequences. Cambridge University Press, New York, NY, 1997.

[12] L. Holm, and C. Sander. Protein Structure Comparison by Alignment of Distance Matrices. *Journal of Molecular Biology*, 233, 1993, pp. 123–138.

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

[14] L. Holm, and C. Sander. The FSSP database: Fold classification based on structure-structure alignment of proteins. *Nucleic Acids Research*, 24(1), 1996, pp. 206–209.

[15] M.S. Johnson, A. Sali, and T.L. Blundell. Phylogenetic relationships from three-dimensional structures of proteins. *Methods in Enzymology*, 183, 1990, pp. 670–690.

[16] A.M. Lesk. A toolkit for computational molecular biology. II. On the optimal superposition of two sets of coordinates. *Acta Cryst*, A42, 1986, pp. 110–113.

[17] A. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247, 1995, pp. 536–540.

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

[19] A.P. Singh, and D.L. Brutlag. Hierarchical protein structure superposition using both secondary structure and atomic representation. *Proc. Intelligent Systems for Molecular Biology*, pp. 284–293, 1997.

[20] W.R. Taylor. Protein structure comparison using iterated double dynamic programming. *Protein Science*, 8, 1999, pp. 654–665.

[21] W.R. Taylor, and C. Orengo. Protein structure alignment. *Journal of Molecular Biology*, 208, 1989, pp. 1–22.

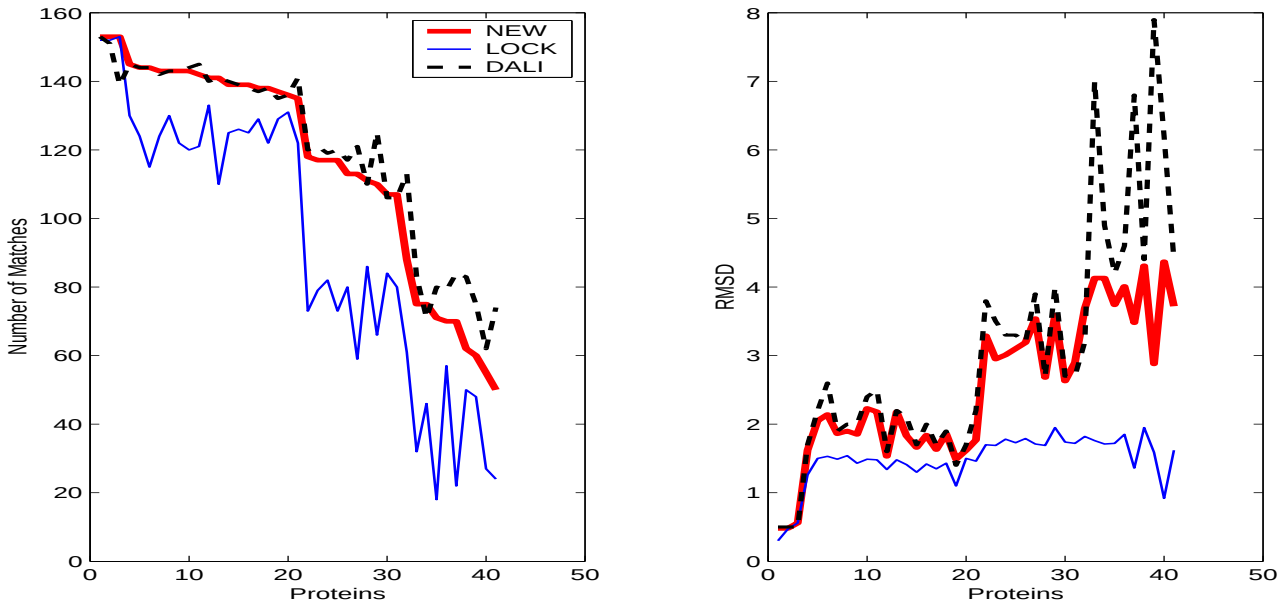

Figure 4: Comparison between the proposed method (NEW), LOCK, and DALI using query protein 1mbc (Data used is from Table 1.). The proteins on the x-axis are ordered by non-increasing number of matches in the NEW method.

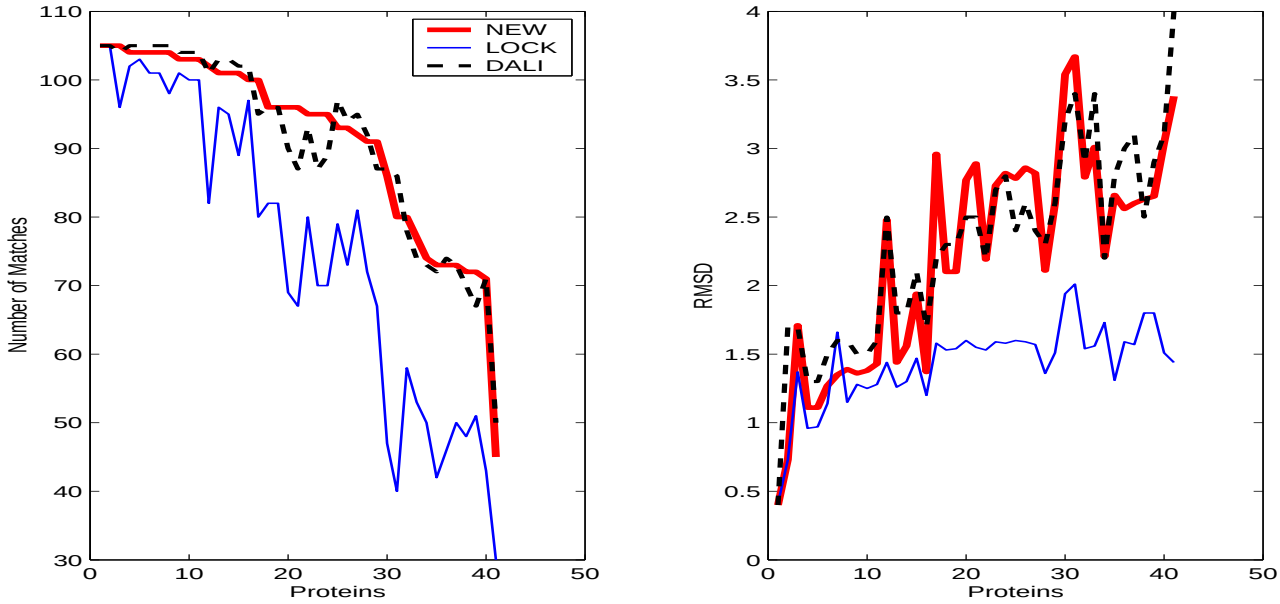

Figure 5: Comparison between the proposed method (NEW), LOCK, and DALI using query protein 3trx (Data used is from Table 2.). The proteins on the x-axis are ordered by non-increasing number of matches in the NEW method.

Protein PDB ID	PDB Header (partial)	NEW #	NEW RMSD	LOCK #	LOCK RMSD	DALI #	DALI RMSD
5mbn	MYOGLOBIN (DEOXY)	153	0.48	152	0.30	153	0.50
1mbn	MYOGLOBIN (FERRIC IRON)	153	0.48	152	0.47	151	0.50
1myh-A	MYOGLOBIN (AQUOMET, PH 7.1)	153	0.56	153	0.56	139	0.50
1hds-B	HEMOGLOBIN (SICKLE CELL)	145	1.62	130	1.26	145	1.70
2dhb-A	HEMOGLOBIN (HORSE,DEOXY)	141	1.53	133	1.34	140	1.60
1mba	MYOGLOBIN (MET)	143	1.86	124	1.49	142	1.90
1dm1	MYOGLOBIN	143	1.90	130	1.54	143	2.00
1hlm	HEMOGLOBIN (CYANO-MET)	144	2.05	124	1.50	144	2.20
2lhb	HEMOGLOBIN V (CYANO,MET)	137	1.49	129	1.10	135	1.40
2fal	MYOGLOBIN (FERRIC)	143	1.85	122	1.43	143	2.00
1hbg	HEMOGLOBIN (CARBON MONOXY)	139	1.83	125	1.41	140	2.10
1ith-A	HEMOGLOBIN (CYANOMET)	139	1.66	126	1.30	139	1.70
1flp	HEMOGLOBIN I (MONOMERIC)	138	1.63	129	1.35	137	1.70
1eca	HEMOGLOBIN (ERYTHROCRUORIN)	136	1.62	131	1.50	136	1.70
2hbg	HEMOGLOBIN (DEOXY)	139	1.84	125	1.42	138	2.00
1ash	HEMOGLOBIN (DOMAIN ONE)	138	1.86	122	1.43	138	1.90
1hbi-B	HEMOGLOBIN I (OXYGENATED)	135	1.77	122	1.46	141	2.20
1gdi	LEGHEMOGLOBIN	144	2.14	115	1.53	144	2.60
1hlb	HEMOGLOBIN (SEA CUCUMBER)	142	2.18	121	1.48	145	2.50
1lh2	LEGHEMOGLOBIN (AQUO,MET)	143	2.23	120	1.49	144	2.40
1h97-A	HEMOGLOBIN	141	2.18	110	1.48	141	2.20
1dly-A	HEMOGLOBIN	111	2.68	86	1.69	110	2.70
1idr-A	HEMOGLOBIN HBN	107	2.63	84	1.74	106	2.70
1dlw-A	HEMOGLOBIN	107	2.90	80	1.72	106	2.70
1all-A	ALLOPHYCOCYANIN	117	2.95	79	1.69	121	3.50
1phn-A	PHYCOCYANIN	117	3.01	82	1.78	119	3.30
1cpc-A	C-PHYCOCYANIN	113	3.19	80	1.79	117	3.20
1lia-A	R-PHYCOERYTHRIN	117	3.10	73	1.73	120	3.30
1cpc-B	C-PHYCOCYANIN	118	3.29	73	1.70	120	3.80
1qgw-C	CRYPTOPHYTAN PHYCOERYTHRIN	110	3.59	66	1.95	125	4.00
1lia-B	R-PHYCOERYTHRIN	113	3.54	59	1.71	121	3.90
1col-A	COLICIN *A (C-TERMINAL DOMAIN)	88	3.69	61	1.82	113	3.20
2cp4	CYTOCHROME P450CAM	62	4.31	50	1.95	83	4.40
1eum-A	FERRITIN 1	75	4.13	32	1.76	83	7.00
1fpo-A	CHAPERONE PROTEIN HSCB (HSC20)	71	3.74	18	1.72	80	4.20
1oxa	CYTOCHROME P450 ERYF	70	4.01	57	1.85	79	4.60
1le2	APOLIPOPROTEIN-*E2	75	4.13	46	1.71	71	4.90
2fha	FERRITIN	70	3.48	22	1.36	84	6.80
1nfn	APOLIPOPROTEIN E3 FRAGMENT	60	2.88	48	1.59	75	7.90
1grj	GREA TRANSCRIPT CLEAVAGE FACTOR	55	4.37	27	0.92	62	6.20

Table 1: Query Protein: 1mbc (*Sperm whale myoglobin*)

Protein PDB ID	PDB Header (partial)	NEW #	NEW RMSD	LOCK #	LOCK RMSD	DALI #	DALI RMSD
4trx	THIOREDOXIN (REDUCED FORM)	105	0.40	105	0.41	105	0.40
1mdi-A	THIOREDOXIN MUTANT	105	0.73	105	0.73	105	1.70
1aiu	HUMAN THIOREDOXIN (MUTANT)	104	1.11	102	0.96	105	1.30
1erv	THIOREDOXIN	104	1.11	103	0.97	105	1.30
1f9m-B	THIOREDOXIN F	103	1.36	101	1.28	104	1.50
1f9m-A	THIOREDOXIN F	103	1.38	100	1.25	104	1.50
1gh2-A	THIOREDOXIN-LIKE PROTEIN	104	1.27	101	1.14	105	1.50
1ep7-A	THIOREDOXIN CH1, H-TYPE	104	1.35	101	1.66	105	1.60
1ep7-B	THIOREDOXIN CH1, H-TYPE	104	1.39	98	1.15	105	1.60
1faa	THIOREDOXIN F	103	1.43	100	1.28	104	1.60
1tof	THIOREDOXIN H	105	1.71	96	1.37	104	1.70
2tir	THIOREDOXIN MUTANT	101	1.44	96	1.26	103	1.80
1thx	THIOREDOXIN-2	101	1.56	95	1.30	103	1.80
1fb6-B	THIOREDOXIN M	100	1.37	97	1.20	102	1.70
1quw	THIOREDOXIN	101	1.94	89	1.47	102	2.10
1kte	THIOLTRANSFERASE	86	3.54	47	1.94	87	3.20
1jhb	GLUTAREDOXIN	80	3.67	40	2.01	86	3.40
3grx	GLUTAREDOXIN 3	74	2.21	50	1.73	73	2.20
1h75-A	GLUTAREDOXIN-LIKE PROTEIN NRDH	72	2.63	48	1.80	70	2.50
1ego	GLUTAREDOXIN (OXIDIZED)	73	2.66	42	1.31	72	2.80
1ilo	HYPOTHETICAL PROTEIN MTH895	71	3.04	43	1.51	71	3.10
1aba	GLUTAREDOXIN MUTANT	72	2.65	51	1.80	67	2.90
1fo5-A	THIOREDOXIN	77	3.01	53	1.56	74	3.40
1mek	PROTEIN DISULFIDE ISOMERASE	102	2.48	82	1.44	101	2.50
1a8y	CALSEQUESTRIN	100	2.96	80	1.58	95	2.20
1e2y-A	TRYPAREDOXIN PEROXIDASE	96	2.10	82	1.53	96	2.30
1e2y-C	TRYPAREDOXIN PEROXIDASE	96	2.10	82	1.54	96	2.30
re 1qmv-A	HUMAN THIOREDOXIN PEROXIDASE-B	93	2.78	79	1.60	97	2.40
1bjx	PROTEIN DISULFIDE ISOMERASE	95	2.19	80	1.53	93	2.20
1gp1-B	GLUTATHIONE PEROXIDASE	92	2.82	81	1.57	95	2.40
1qq2-A	THIOREDOXIN PEROXIDASE 2	93	2.86	73	1.59	94	2.60
1ezk-A	TRYPAREDOXIN I	96	2.77	69	1.60	90	2.50
1ewx-A	TRYPAREDOXIN I	95	2.73	70	1.59	87	2.70
1qk8-A	TRYPAREDOXIN-I	95	2.82	70	1.58	89	2.80
1fg4-A	TRYPAREDOXIN II	96	2.89	67	1.55	87	2.50
1a8l	PROTEIN DISULFIDE OXIDOREDUCTASE	91	2.11	72	1.36	92	2.30
1fg4-B	TRYPAREDOXIN II	91	2.60	67	1.51	87	2.60
1fvk-A	DISULFIDE BOND FORMATION PROTEIN	80	2.79	58	1.54	78	2.90
1f37A	FERREDOXIN [2FE-2S]	73	2.56	46	1.59	74	3.00
1f37B	FERREDOXIN [2FE-2S]	73	2.60	50	1.57	73	3.10
1ghh-A	DNA-DAMAGE-INDUCIBLE PROTEIN I	45	3.38	30	1.44	50	4.00

Table 2: Query Protein: 3trx (*Thioredoxin-Reduced Form*)

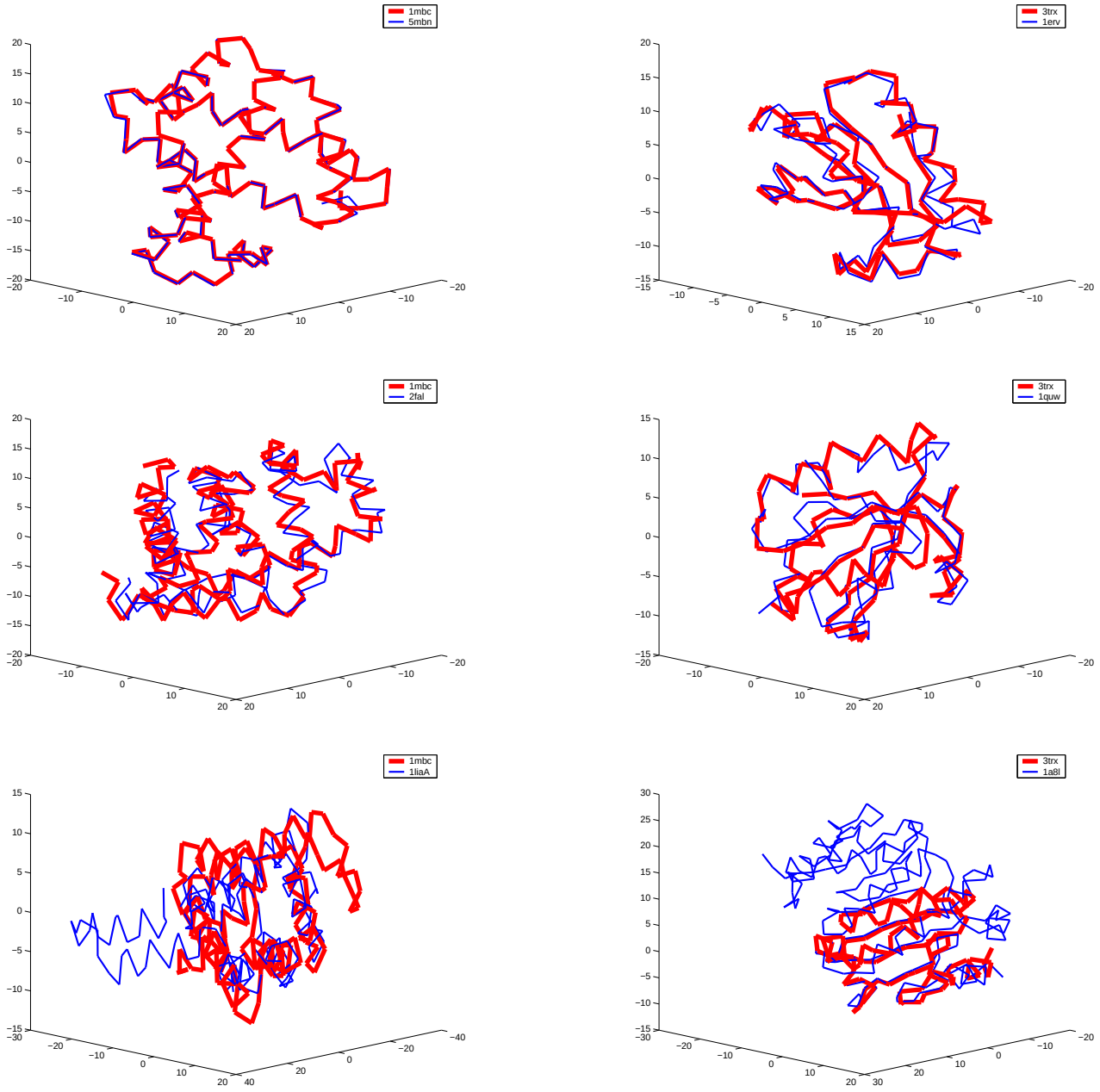

Figure 6: Left side: Query protein 1mbc aligned with 5mbn, 2fal, and 1lia-A. Right side: Query protein 3trx aligned with 1erv, 1quw, and 1a8l.