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 03-018

Approximate Multiple Protein Structure Alignment Using the Sum-of-Pairs Distance

Jieping Ye and Ravi Janardan

March 31, 2003

Approximate multiple protein structure alignment using the Sum-of-Pairs distance

Jieping Ye* Ravi Janardan*

Abstract

An algorithm is presented to compute a multiple structure alignment for a set of proteins and to generate a consensus (pseudo) protein for the set. The algorithm is a heuristic in that it computes an approximation to the optimal multiple structure alignment that minimizes the sum of the pairwise distances between the protein structures. The algorithm chooses an input protein as the initial consensus and computes a correspondence between the protein structures (which are represented as sets of unit vectors) using an approach analogous to the center-star method for multiple sequence alignment. From this correspondence, a set of rotation matrices (optimal for the given correspondence) is derived to align the structures and derive the new consensus. The process is iterated until the sum of pairwise distances converges. The computation of the optimal rotations is itself an iterative process that both makes use of the current consensus and generates simultaneously a new one. This approach is based on an interesting result that allows the sum of all pairwise distances to be represented compactly as distances to the consensus. Experimental results on several protein families are presented, showing that the algorithm converges quite rapidly.

Keywords: structure alignment, center-star method, dynamic programming, consensus protein, correspondence, rotation matrix, Frobenius norm, singular value decomposition.

1 Introduction

Proteins are macromolecules that regulate all biological processes in a cellular organism [2]. The human body has about one hundred thousand different proteins that control functions as diverse as oxygen transport, blood clotting, tissue growth, immune system response, inter-cell signal transmission, and the catalysis of enzymatic reactions.

Proteins are synthesized within the cell and immediately after its creation each protein folds spontaneously into a three-dimensional (3D) configuration that is determined uniquely by its constituent amino acid sequence [20]. It is this 3D structure that ultimately determines the function of a protein be it the catalysis of a reaction or the growth of muscle tissue or arming the body's immune system. Indeed, it is the case that where proteins are concerned "function follows form" [19].

Proteins have evolved over time through the modification and re-use of certain substructures that have proven successful [11]. It is well known [13] that during this process, structure is better conserved than sequence, i.e., proteins that are related through evolution tend to have similar

*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.

structures even though their sequences may be quite different. Thus, the ability to identify common substructures in a set of proteins could yield valuable clues to their evolutionary history and function. This motivates the *multiple structure alignment problem* that we consider in this paper: Informally, given a collection of protein structures, we seek to align them in space, via rigid motions, such that large matching substructures are revealed. (A formal definition is given in Section 3.) A second goal is to extract from this alignment a consensus structure that can serve as a proxy for the whole set. This consensus may not be a real protein, but it should capture the essence of the set so that it can, for instance, serve as a template to perform fast searches through protein structure databases, such as the PDB [1] to identify similar proteins.

In this paper, we present an algorithm to compute a multiple structure alignment for a set of proteins, along with their consensus structure. Our algorithm represents the input proteins and the consensus as sets of unit vectors and it computes an approximation to the optimal multiple structure alignment, i.e., the one that minimizes the so-called sum-of-pairs (SP) distance. The *SP distance* of a multiple structure alignment is the sum of the distances between the vector representations of each pair of proteins in the alignment. (This is similar to the notion of *SP score* used for multiple sequence alignment [11], and is defined formally in Section 3.)

Our algorithm begins by computing a correspondence between the unit vectors in the different protein structures, by choosing one of the proteins as the initial consensus and applying an algorithm that is analogous to the center-star method for multiple sequence alignment [11]. It then derives a set of rotation matrices that are optimal for the computed correspondence and uses these to align the structures in space via rigid motions and obtain the new consensus. The process is repeated until the change in SP distance is less than a prescribed threshold. The computation of the optimal rotations is itself an iterative process that both uses the current consensus and generates simultaneously a new one. This approach is based on an interesting result that we establish, which allows the sum of all pairwise distances to be represented succinctly as distances to the consensus. This allows us to use the singular value decomposition method [10] to compute the rotation matrices. Our experiments on several protein families show that the algorithm converges quite rapidly.

Some prior work on multiple structure alignment includes [4, 8, 9, 14, 17, 18]. In [17, 18], algorithms are given for pairwise structure alignment using a 2-level dynamic programming approach and the multiple alignment is obtained by aligning the pairs according to their pairwise similarity scores. Leibowitz *et.al.* [14] use the technique of geometric hashing to compute a multiple alignment and core; unlike most other algorithms, theirs does not require an ordered sequence of atoms along the protein backbone. Gerstein and Levitt [8, 9] use an iterative dynamic programming method to compute a multiple structure alignment. In [7], Gerstein and Altman show how to compute a consensus from a given multiple sequence alignment. Recently, Chew and Kedem [4] have shown how to compute both a multiple structure alignment and a consensus structure. They represent proteins as vector sets (in 4D to facilitate the use of special gap vectors), compute an initial consensus, and use iterative dynamic programming to compute the alignment and refine it using different heuristics. In this respect our method is somewhat similar, but, as we will see, it starts with a completely different perspective in terms of the computation of the correspondence and the refinement. More importantly, our method formulates the multiple alignment problem in a compact matrix form, which facilitates efficient computation.

The multiple structure alignment problem is much harder than its sequence counterpart since it requires not only computing a correspondence between the structures (often done via sequence alignment) but also computing the rigid motions that bring corresponding elements into alignment. The sequence version has been shown to be NP-hand [23]. It can be solved exactly using an

expensive dynamic programming-based method [16, 11]. Algorithms to approximate it include the center-star method [11], the progressive method [11, 6], and a method based on Hidden Markov Models [5].

An important special case of multiple structure alignment is pairwise structure alignment, which involves aligning only two protein structures. Indeed, this problem arises in this paper when computing of correspondences between different proteins. We use here a variant of an algorithm that we have developed recently for pairwise structure alignment [24]. This algorithm uses a representation of the protein backbones that is independent of the relative orientations of the two proteins and applies dynamic programming to obtain an initial alignment, which is further refined iteratively. Some other algorithms for pairwise alignment include LOCK [22], DALI [12], CE [21], and a method in [3].

The rest of the paper is organized as follows. In Section 2 we introduce some terminology and discuss the center-star method and a variant of this that we will use. Section 3 defines the multiple structure alignment problem formally and establishes a key result on the SP distance function. We give an overview of our algorithm in Section 4 and describe it in detail in Section 5. We discuss experimental results in Section 6 and conclude in Section 7.

2 Preliminaries

2.1 Representation of protein structures

Let A be a protein of length n consisting of a chain of C_α atoms, numbered $1, 2, \cdots, n$, along the backbone in $\mathbb{R}^3$. (As is customary [12, 22], we consider only the backbone, not the amino acid residues themselves.) Following [3, 4], 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. Hence we can simplify the representation of the backbone and view it as a sequence of unit vectors, $\mathbf{u}_i$, for $i = 1, \cdots, n-1$, where $\mathbf{u}_i$ has the same direction as $\mathbf{a}_i$. As in [3, 4], we assume that $\mathbf{u}_i$ has been translated so that its tail is at the origin. Let B be another protein of length n, whose backbone is represented by the unit vectors $\mathbf{v}_1, \cdots, \mathbf{v}_n$. Suppose that there is a 1-1 correspondence between $\mathbf{u}_i$ and $\mathbf{v}_i$, $1 \leq i \leq n$. Then the *(squared) distance between A and B* is defined as

$$D(A, B) = \sum_{i=1}^{n} ||\mathbf{u}_i - \mathbf{v}_i||_2^2. \tag{1}$$

We will use this distance measure to define a scoring scheme for multiple structure alignment in Section 3.

2.2 Review of center-star method for multiple sequence alignment

The center-star method is an efficient approximation algorithm for multiple sequence alignment [11]. Given a set $\mathcal{S}$ of K proteins, a *center protein* is determined, which minimizes the sum of the distances to all the other proteins in the given data set. Specifically, let $d(i, j)$ be the minimum pairwise edit distance between the ith and jth proteins in the set $\mathcal{S}$. Then the center protein is one that minimizes the sum of the edit distances to all the other proteins. That is, if

$$k^\star = \arg \min_{1 \leq k \leq K} \sum_{i=1}^{K} d(k, i),$$

then the $k^\star$th protein is a center protein.

The center protein is then aligned iteratively with each of the other $K-1$ proteins in the data set, using an optimum pairwise sequence alignment algorithm. All of these $K-1$ pairwise alignments are then combined iteratively to get a multiple alignment.

The following simple example shows the main idea of the center-star method. $\mathcal{S}$ consists of the following sequences:

$$\begin{aligned} P_1 &= BBCA \\ P_2 &= CBBA \\ P_3 &= BCCA \end{aligned}$$

If we assume zero distance for an exact match and 1 for a mismatch (including alignment of a space with a non-space), then an optimal pairwise alignment of each pair is as follows

$$\begin{array}{llllll} P_1: & - & B & B & C & A \\ P_2: & C & B & B & - & A \end{array}, \quad \begin{array}{llllll} P_1: & B & B & C & A \\ P_3: & B & C & C & A \end{array}, \text{ and } \quad \begin{array}{llllll} P_2: & C & B & B & A \\ P_3: & B & C & C & A \end{array}. \tag{2}$$

Thus,

$$d(1,2) = 2, d(1,3) = 1, d(2,3) = 3.$$

Thus, P_1 is the center protein. A multiple sequence alignment is now computed as follows (details can be found in [11, pages 347–350]): First an optimal pairwise alignment of P_1 and P_2 is computed. This gives "expanded versions", $\bar{P}_1$ and $\bar{P}_2$, of P_1 and P_2, respectively, that include spaces. Then an optimal pairwise alignment of $\bar{P}_1$ and P_3 is computed. Any new spaces introduced in $\bar{P}_1$ are also added into $\bar{P}_2$. For our example, this gives the following multiple sequence alignment (here no new spaces are introduced in $\bar{P}_1 = -BBCA$ when it is aligned with $P_3 = BCCA$.)

$$\left\{ \begin{array}{llllll} - & B & B & C & A \\ C & B & B & - & A \\ - & B & C & C & A \end{array} \right. \tag{3}$$

Note that since the distance between opposing spaces is zero, the induced pairwise alignments between the center protein P_1 and the proteins P_2 and P_3 are consistent with the original pairwise alignments, and have the same cost as the corresponding optimal pairwise alignments.

The quality of a multiple sequence alignment is measured usually by summing the edit distances between each pair of proteins, scoring 0 for a match (including matching of two spaces) and 1 for a mismatch (including matching of a space to a non-space). This is called the *Sum-of-Pairs (SP) score* [11]. The SP score of the multiple alignment in Equation (3) is 6.

2.3 Center-star-like method

In our algorithm, we need to compute a correspondence between the protein structures prior to applying rotations. For this we choose an initial consensus protein, apply pairwise alignment [24] between this and each of the other proteins, and then combine these using a method similar to the center-star method. In what follows, we call this a *center-star-like* method. As we will see, a key difference between the two is that in the center-star-like method we will be aligning not alphabet characters representing amino acids but unit vectors derived from the protein backbones.

A second difference is the choice of the initial consensus protein. For multiple sequence alignment, the alignment produced is quite sensitive to the choice of the initial consensus and the resulting pairwise alignments. It turns out [11] that the center protein is a good choice, but this requires computing all pairwise alignments, which is expensive. In multiple structure alignment, the correspondence computed using the center-star-like method is but a first step; it is followed by an optimization step to compute the optimal rotation matrices. It turns out that the final multiple structure alignment is not that sensitive to the initial consensus protein chosen. This is also borne out by our experiments in Section 6, where we compare several methods, for making the initial choice. We will see that a simple, computationally inexpensive choice does just as well as more expensive methods.

3 Multiple structure alignment

Let $\{P_i\}_{i=1}^{K}$ be the K proteins in the given data set, each represented by a sequence of unit vectors $\{\mathbf{u}_j^i\}_{j=1}^{L_i}$, for $i = 1, \cdots, K$ as described in section 2.1. Here L_i is the number of unit vectors in the ith protein P_i.

A *correspondence* $\mathcal{C}$ of the K proteins can be represented as a matrix $H = (\mathbf{h}_{ij})_{1 \le i \le K, 1 \le j \le L}$ for some $L \ge \max_{1 \le i \le K}\{L_i\}$, where $\mathbf{h}_{ij}$ is either a unit vector belonging to the ith protein or a special vector called a *gap vector*, which represents a space.[1] Omitting the spaces, the ith row reproduces the sequence of unit vectors of the ith protein.

To distinguish between regular unit vectors obtained from a protein and the gap vector, we extend the unit vectors in the original 3D space to four dimensions by introducing a special *gap direction*, as in [4]. As a result, gap vectors are represented as $(0, 0, 0, 1)$ and regular unit vectors are extended by introducing zeros in the fourth dimension. For simplicity, we assume the terms $\mathbf{h}_{ij}$ in the matrix H have already been extended to $\mathbb{R}^4$. Distances are based on the squared distance between the vectors in $\mathbb{R}^4$. Hence the distance (gap penalty) between a regular vector (with 0 in its fourth dimension) and a gap is 2, since

$$|| \, (x, y, z, 0) - (0, 0, 0, 1) \, ||^2 = || \, (x, y, z) \, ||^2 + ||1||^2 = 1 + 1 = 2,$$

if (x, y, z) is a unit vector in 3D.

A *multiple structure alignment*, $\mathcal{M}$, of K proteins based on the correspondence $\mathcal{C}$ can be represented as another matrix $G = (\mathbf{g}_{ij})_{1 \le i \le K, 1 \le j \le L}$, where the set of unit vectors in the ith row is the rotation of the set of unit vectors in the ith row of the matrix H. More specifically, we combine all of the unit vectors $\{\mathbf{h}_{ij}\}_{j=1}^{L}$ from the ith protein, i.e. the ith row of the matrix H, into a column vector H_i as follows (G_i is defined similarly from the matrix G):

$$H_i = \begin{pmatrix} \mathbf{h}_{i1} \\ \vdots \\ \mathbf{h}_{iL} \end{pmatrix} \in \mathbb{R}^{L \times 4} \quad \text{and} \quad G_i = \begin{pmatrix} \mathbf{g}_{i1} \\ \vdots \\ \mathbf{g}_{iL} \end{pmatrix} \in \mathbb{R}^{L \times 4}, \quad \text{for} \quad i = 1, \cdots, K.$$

Then $G_i = H_i Q_i$, where

$$Q_i = \begin{pmatrix} R_i & 0 \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{4 \times 4}$$

[1] "Space vector" is perhaps a more appropriate term. However, we will retain the terminology introduced in [4]

and $R_i \in \mathbb{R}^{3\times 3}$ is some rotation matrix. (The two zeros in Q_i are zero vectors of appropriate dimension.) Recall the fourth dimension of every unit vector $\mathbf{h}_{ij} \in \mathbb{R}^4$ will not change after the rotation, i.e. vectors $\mathbf{h}_{ij}$ and $\mathbf{g}_{ij}$ have the same last component. This is the reason for going to 4D space, since the regular unit vectors (the first three components of the unit vectors in $\mathbb{R}^4$) and the gaps (the last component) are completely separate. After the rotation, gap vectors remain gap vectors and the regular unit vectors still have zero as their last component. Hence we can treat gaps and regular vectors in a uniform way in our matrix computations. In the rest of this paper, we use $\boldsymbol{\chi}$ to denote the set of rotation matrices in 4D. More formally, $\boldsymbol{\chi}$ is defined as

$$\boldsymbol{\chi} = \left\{ Q \in \mathbb{R}^{4\times 4} : Q = \begin{pmatrix} R & 0 \\ 0 & 1 \end{pmatrix} \text{ and } R \in \mathbb{R}^{3\times 3} \text{ is a rotation matrix in 3D} \right\}. \tag{4}$$

Under the multiple structure alignment $\mathcal{M}$, we can define the *distance between protein P_i and protein P_j* as $D^{\mathcal{M}}(P_i, P_j) = ||H_iQ_i - H_jQ_j||_F^2 = ||G_i - G_j||_F^2 = \sum_{\ell=1}^{L} ||\mathbf{g}_{i\ell} - \mathbf{g}_{j\ell}||_2^2$, where $||\cdot||_F$ denotes the Frobenius norm [10].

The *Sum-of-Pairs (SP) distance*of the K proteins in $\mathcal{M}$ is then defined as [2]

$$SP(\mathcal{M}) = \sum_{1\le i<j\le K} D^{\mathcal{M}}(P_i, P_j) = \sum_{1\le i<j\le K} ||H_iQ_i - H_jQ_j||_F^2. \tag{5}$$

We can now define our multiple structure alignment problem as follows:

> **Multiple Structure Alignment Problem:** *Given a set $\mathcal{S} = \{P_1, \cdots, P_K\}$, find a correspondence, $\mathcal{C}$, and rotation matrices, Q_i, for $i = 1, \cdots, K$, such that the resulting multiple structure alignment, $\mathcal{M}$, has minimum SP distance, $SP(\mathcal{M})$, as defined in Equation (5).*

In the next two sections, we will present a heuristic for this problem. Our algorithm approximates the global minimum of the SP distance, $SP(\mathcal{M})$, by iterative refinement of an initial multiple structure alignment and converges to a local minimum.

3.1 Average Structure

We now introduce the notion of an average protein which will enable us to define a consensus structure. We define the *average structure* P_a of the K proteins $\{P_i\}_{i=1}^K$ in $\mathcal{M}$ as a sequence of L unit vectors $\{\mathbf{u}_a^j\}_{j=1}^L$ in $\mathbb{R}^4$, where

$$\mathbf{u}_a^j = \frac{1}{K} \sum_{i=1}^{K} \mathbf{u}_i^j.$$

Note that the vector $\mathbf{u}_a^j$ is not necessarily a unit vector; furthermore, it could point in a direction that is a combination of the gap direction and the other three dimensions.

A major advantage of the average structure is that it leads to a simplified and more compact formulation for the SP distance in Equation (5). As the following lemma shows, rather than consider the distance between every pair P_i and P_j of proteins, it suffices to consider the distance of each protein to the average structure P_a.

[2] We use "SP distance" to distinguish our scoring function from the "SP score" used for multiple sequence alignment.

Lemma 3.1 *Let $\{P_i\}_{i=1}^K$, $\mathcal{M}$, and $SP(\mathcal{M})$ be defined as above, and let P_a be the average structure of $\{P_i\}_{i=1}^K$ in $\mathcal{M}$. Then*

$$SP(\mathcal{M}) = K \sum_{1 \le i \le n} D^{\mathcal{M}}(P_a, P_i).$$

Proof Follows directly from Lemma 3.2 below. ∎

Lemma 3.2 *Let $\{a_i\}_{i=1}^n$ be n real numbers and $a = \frac{1}{n}\sum_{i=1}^n a_i$. Then*

$$\sum_{1 \le i < j \le n} (a_i - a_j)^2 = n \sum_{i=1}^{n} (a_i - a)^2. \tag{6}$$

Proof The right hand side of (6) can be expanded as

$$\begin{aligned}
n\sum_{i=1}^{n}(a_i - a)^2 &= n\sum_{i=1}^{n}(a_i^2 - 2a_i a + a^2) \\
&= (n\sum_{i=1}^{n} a_i^2) - n^2 a^2 \\
&= (n\sum_{i=1}^{n} a_i^2) - (a_1 + a_2 + \cdots + a_n)^2 \\
&= (n-1)\sum_{i=1}^{n} a_i^2 - 2\sum_{1 \le i < j \le n} a_i a_j \\
&= \sum_{1 \le i < j \le n} (a_i - a_j)^2,
\end{aligned}$$

which is the left side of (6). ∎

It follows from Lemma 3.1 that finding an $\mathcal{M}$ with minimum SP distance, $SP(\mathcal{M})$, is equivalent to finding an $\mathcal{M}$ which minimizes $\sum_{1 \le i \le n} D^{\mathcal{M}}(P_a, P_i)$. The algorithm developed in the next section computes an approximation to the optimal $\mathcal{M}$ iteratively.

4 Overview of the algorithm

Our strategy is similar to steepest descent. Starting from an initial multiple alignment, we will update the alignment incrementally with decreasing SP distance. The algorithm finally stops at some local minimum. The expectation is that with a good starting alignment, the final alignment will be close to the optimal solution.

We choose an initial consensus protein J^0 from the given set of proteins. There are many ways to choose J^0, as we discuss in Section 5.1. J^0 is then aligned with the rest of the proteins using the center-star-like method from Section 2.3 to get a correspondence $\mathcal{C}^1$ between the K proteins. As mentioned earlier, to get an multiple alignment $\mathcal{M}^1$ from $\mathcal{C}^1$, we need to find the rotation matrices R_j^1 for every P_j (optimal for $\mathcal{C}^1$). We will show in the next section how to find these matrices such that the SP distance is minimum for the chosen correspondence. The multiple alignment $\mathcal{M}^1$ is then post-processed by removing all columns consisting of only gaps. Since the distance between any two gaps is zero, this will not change the SP distance of the multiple alignment.

After we compute the multiple alignment $\mathcal{M}^1$, we compute the average structure from $\mathcal{M}^1$ as described in Section 3 and use this as our next consensus protein J^1. Dynamic programming is

Algorithm 1: The overall algorithm for multiple protein structure alignment

1. Choose initial consensus protein J^0 from $\{P_i\}_{i=1}^K$. $i \leftarrow 0$. $SP^0 \leftarrow \infty$.
2. Do
3. if $i = 0$ then compute pairwise structure alignment between J^i and every P_j.
4. else use standard dynamic programming to align J^i with every P_j.
5. $i \leftarrow i + 1$.
6. Compute correspondence $\mathcal{C}^i$ from the above pairwise alignments using center-star-like method.
7. Compute optimal rotation matrices R_j^i, and transform P_j by R_j^i for every j to obtain multiple structure alignment $\mathcal{M}^i$. $SP^i \leftarrow SP(\mathcal{M}^i)$.
8. Post-process $\mathcal{M}^i$ by removing all columns consisting of only gaps.
9. Compute new consensus protein J^i from $\mathcal{M}^i$ by taking the average structure.
10. Until $\left|SP^i - SP^{i-1}\right| \leq \eta$.

applied between J^1 and every other protein P_j to further reduce the pairwise distances between J^1 and the P_j's. This yields a new correspondence $\mathcal{C}^2$. (Recall that all the proteins P_j and the consensus protein J^1 are represented as sequences of vectors in $\mathbb{R}^4$. The distance measure used in the dynamic programming is the squared distance between two vectors in $\mathbb{R}^4$.) Based on the new correspondence $\mathcal{C}^2$, we find new rotation matrices for every protein and get a new multiple alignment with lower SP distance. The process is repeated until the SP distance converges.

The main steps are summarized in *Algorithm* 1; details are discussed in the next section. In our implementation, we chose the convergence threshold $\eta = 0.001$. That the algorithm converges, i.e., the SP distance is non-increasing from iteration i to iteration $i + 1$, is based on the following observations: First, $SP(\mathcal{M}^i) = K \sum_j D^{\mathcal{M}^i}(J^i, P_j) \geq K \sum_j D^\star(J^i, P_j)$, where $D^\star(J^i, P_j)$ is the optimal cost of the alignment between J^i and P_j computed by dynamic programming. Second, by the property of the center-star-like method, $\sum_j D^\star(J^i, P_j)$ is equal to the sum of the costs of the pairwise alignments between J^i and the P_j's induced by $\mathcal{C}^{i+1}$. And, third, the multiple alignment using the rotation matrices computed from $\mathcal{C}^{i+1}$ (*Algorithm* 2, given later) does not increase the latter cost.

5 Details of the algorithm

5.1 Step 1: Obtain initial consensus protein

There are many ways to choose the initial consensus protein J^0. One possibility is to choose J^0 as the center protein, as in the center-star method, so that it minimizes the sum of the minimum pairwise distances to all the other proteins. That is J^0 is the $k^\star$th protein, where

$$k^\star = \arg \min_{1 \leq k \leq K} \sum_{i=1}^{K} D(P_k, P_i).$$

Another possibility is to choose J^0 as the $k^\star$th protein, where

$$k^\star = \arg \min_{1 \leq k \leq K} \left(\max_{i \neq k} D(P_k, P_i) \right).$$

Both choices make sense intuitively, since they yield consensus proteins that are "not too far away" from the others; however they are expensive computationally, as they involve $\frac{K(K-1)}{2}$ pairwise alignments. A less expensive choice that appears to work well is to pick J^0 such that it is the protein of median length. We report our experimental results for all three choices in Section 6.

5.2 Step 2: Compute pairwise structure alignment

After we determine the consensus protein J^0 in the first step, the $K-1$ pairwise structure alignments between J^0 and $P_i \neq J^0$, for every $i = 1, \cdots, K$, are computed using the pairwise alignment algorithm developed by us in [24], with one small change: In [24], we apply dynamic programming (in steps 4 and step 5 of that algorithm) to the coordinates of the alpha carbon atoms to align the proteins, whereas here we operate on the unit vector representation. (Other pairwise structure alignment algorithms, such as LOCK [22], DALI [12], CE [21] etc. could also be used instead.)

5.3 Step 3: Compute an initial correspondence

The $K-1$ pairwise structure alignments obtained from Step 2 are combined using the center-star-like method described in Section 2.3 to get an initial correspondence $\mathcal{C}^1$ of the K proteins. This initial correspondence depends strongly on the consensus protein chosen in Step 1 and also the $K-1$ pairwise structure alignments obtained in Step 2.

5.4 Step 4: Compute optimal rotation matrices and average structure

Given a correspondence $\mathcal{C}$ and a consensus protein J, we show how to find both the optimal rotation matrix R_j for each protein P_j as well as the new consensus protein $\bar{J}$.

Assume the correspondence $\mathcal{C}$ is represented as a matrix $H = (\mathbf{h}_{ij})$, as defined in Section 3. Protein P_j in $\mathcal{C}$ can be represented as

$$H_j = \begin{pmatrix} \mathbf{h}_{j1} \\ \vdots \\ \mathbf{h}_{jL} \end{pmatrix} \in \mathbb{R}^{L \times 4}.$$

The objective is to find the rotation matrices $Q_j \in \boldsymbol{\chi}$, for $j = 1, \cdots, K$, such that the SP distance of the multiple alignment $\mathcal{M}$ associated with $\mathcal{C}$ is minimum.

From Equation (5), the SP distance of $\mathcal{M}$ can be represented as

$$SP(\mathcal{M}) = \sum_{1 \leq i < j \leq K} ||H_i Q_i - H_j Q_j||_F^2,$$

$$Q_i = \begin{pmatrix} R_i & 0 \\ 0 & 1 \end{pmatrix} \in \boldsymbol{\chi}$$

and $R_i \in \mathbb{R}^{3 \times 3}$ is some rotation matrix, and $||\cdot||_F$ denotes the Frobenius norm. After we compute the optimal rotation matrices, each of the K proteins is transformed by its corresponding rotation matrix to get a new orientation. Hence the new consensus protein $\bar{J}$ is obtained by taking the average structure of the K proteins (after rotation).

Next, we show how we can obtain the rotation matrices and the consensus protein simultaneously in the following theorem:

Theorem 5.1 *Consider the problem of finding matrices $Q_i \in \boldsymbol{\chi}$, $1 \leq i \leq K$, and $J \in \mathbb{R}^{L \times 4}$ that minimize*

$$\sum_{i=1}^{K} ||H_i Q_i - J||_F^2. \tag{7}$$

Let $Q_i^\star$, $1 \leq i \leq K$ and $J^\star$ be the optimal solution to this problem. Then $J^\star = \frac{1}{K}\sum_{i=1}^{K} H_i Q_i^\star$. Furthermore, the $Q_i^\star$, $1 \leq i \leq K$, minimize

$$\sum_{1 \leq i < j \leq K} ||H_i Q_i - H_j Q_j||_F^2. \tag{8}$$

Proof First, we show that if $Q_i^\star \in \boldsymbol{\chi}$, for $i = 1, \cdots, K$, and $J^\star$ minimize Equation (7), then

$$J^\star = \frac{1}{K}\sum_{i=1}^{K} H_i Q_i^\star.$$

Denote $H_i Q_i$ by A_i. Let $a_i(j,k)$ be the (j,k)th term of A_i and $r(j,k)$ be the (j,k)th term of J. Then Equation (7) can be rewritten as

$$\sum_{i=1}^{K} ||H_i Q_i - J||_F^2 = \sum_{i=1}^{K}\sum_{j=1}^{L}\sum_{k=1}^{4} \left(a_i(j,k) - r(j,k)\right)^2 = \sum_{j=1}^{L}\sum_{k=1}^{4}\left(\sum_{i=1}^{K}\left(a_i(j,k) - r(j,k)\right)^2\right). \tag{9}$$

The $4L$ terms in the innermost summation in Equation (9) are independent of each other, hence the minimization of Equation (7) is equivalent to the minimization of each of the terms

$$\sum_{i=1}^{K}\left(a_i(j,k) - r(j,k)\right)^2, \text{ for } 1 \leq j \leq L \text{ and } 1 \leq k \leq 4.$$

By Lemma 5.1 below, $\sum_{i=1}^{K}\left(a_i(j,k) - r(j,k)\right)^2$ is minimized when

$$r(j,k) = \frac{1}{K}\sum_{i=1}^{K} a_i(j,k).$$

In particular, this is true if $a_i(j,k)$ is (j,k)th term of $H_i Q_i^\star$. In this case, $r(j,k)$ is the (j,k)th term of $J^\star$ and it follows that

$$J^\star = \frac{1}{K}\sum_{i=1}^{K} H_i Q_i^\star.$$

We now show the second part of the theorem. By Lemma 3.1,

$$\sum_{1 \leq i < j \leq K} ||H_i Q_i - H_j Q_j||_F^2 = K\sum_{i=1}^{K} ||H_i Q_i - J||_F^2. \tag{10}$$

Since the $Q_i^\star$ and $J^\star$ minimize the right hand side of Equation (10), it follows that the $Q_i^\star$ minimize

$$\sum_{1 \leq i < j \leq K} ||H_i Q_i - H_j Q_j||_F^2, \tag{11}$$

which completes the proof of the lemma. ∎

Lemma 5.1 *Let a_i, $i = 1, \cdots, n$, be n real numbers. The minimum of $\sum_{i=1}^{n}(a_i - r)^2$ is attained when $r = \frac{1}{n}\sum_{i=1}^{n} a_i$.*

Proof Follows by differentiating of $f(r) = \sum_{i=1}^{n}(a_i - r)^2$ with respect to r and setting the result to zero. ∎

Algorithm 2: Solving optimization problem (7)
1. $W_0 \leftarrow J$; $s \leftarrow 0$; $Z^0 \leftarrow \infty$.
2. Do
3. Find $\{Q_i\}_{i=1}^K$ minimizing $\sum_{i=1}^K \|\|H_i Q_i - W_s\|\|_F^2$.
4. $Q_i^s \leftarrow$ minimum Q_i computed in line 3.
5. $s \leftarrow s+1$
6. $W_s \leftarrow \frac{1}{K}\sum_{i=1}^K H_i Q_i^{s-1}$.
7. $Z^s \leftarrow \sum_{i=1}^K \|\|H_i Q_i^{s-1} - W_s\|\|_F^2$.
8. Until $\left\|Z^s - Z^{s-1}\right\| \leq \gamma$.

5.4.1 Solving the optimization problem in Equation (7)

In this section, we show how to solve the optimization problem in Equation (7). We are not aware of an exact analytical solution for this problem, so we solve it approximately. The basic steps for solving (7) are summarized in *Algorithm 2*.

Step 1 in *Algorithm 2* is straightforward. W_0 is our initial estimate of the consensus protein $\bar{J}$ for the current optimization step. Since before every optimization step, we are given a correspondence $\mathcal{C}$ and a consensus protein J (which comes from the previous iteration), we take $W_0 = J$ and successively refine it to arrive at $\bar{J}$.

The main step in *Algorithm 2* is step 3, which is itself an optimization problem. However it is much easier than Equation (7), since the matrix W_s is now fixed. Clearly the K summands of the objective function

$$\sum_{i=1}^{K} ||H_i Q_i - W_s||_F^2$$

are independent of each other. Hence we can find Q_i for every $i = 1, \cdots, K$, by minimizing

$$||H_i Q_i - W_s||_F^2.$$

Since $Q_i = \begin{pmatrix} R_i & 0 \\ 0 & 1 \end{pmatrix}$ for some rotation matrix R_i in $\mathbb{R}^3$, we can write

$$||H_i Q_i - W_s||_F^2$$

as

$$||H_i(:, 1:3)R_i - W_s(:, 1:3)||_F^2 + ||H_i(:, 4) - W_s(:, 4)||_F^2,$$

where for any matrix $A \in \mathbb{R}^{L\times 4}$, $A(:, 1:3)$ denotes the sub-matrix of A including the first three columns and $A(:, 4)$ is the sub-matrix of A containing its last column. Hence we can find the optimal $R_i \in \mathbb{R}^{3\times 3}$ by solving

$$\min_{R_i \in \mathbb{R}^{3\times 3}} ||H_i(:, 1:3)R_i - W_s(:, 1:3)||_F^2. \tag{12}$$

By the definition of the Frobenius norm,

$$\begin{aligned} & || \ H_i(:, 1:3)R_i - W_s(:, 1:3)||_F^2 \\ = \ & \text{trace}\left((H_i(:, 1:3)R_i - W_s(:, 1:3))^T (H_i(:, 1:3)R_i - W_s(:, 1:3))\right) \\ = \ & \text{trace}\left(R_i^T H_i(:, 1:3)^T H_i(:, 1:3)R_i - R_i^T H_i(:, 1:3)^T W_s(:, 1:3)\right. \end{aligned}$$

$$
\begin{aligned}
&- \quad W_s(:,1:3)^T H_i(:,1:3)R_i + W_s(:,1:3)^T W_s(:,1:3)\Big) \\
&= \quad \text{trace}\left(R_i^T H_i(:,1:3)^T H_i(:,1:3)R_i\right) - \text{trace}\left(R_i^T H_i(:,1:3)^T W_s(:,1:3)\right) \\
&- \quad \text{trace}\left(W_s(:,1:3)^T H_i(:,1:3)R_i\right) + \text{trace}\left(W_s(:,1:3)^T W_s(:,1:3)\right). \qquad (13)
\end{aligned}
$$

Using the properties of the trace, which say $\text{trace}(A) = \text{trace}(A^T)$ and $\text{trace}(AB) = \text{trace}(BA)$, for any matrices A and B, we have

$$
\text{trace}\left(R_i^T H_i(:,1:3)^T W_s(:,1:3)\right) = \text{trace}\left(W_s(:,1:3)^T H_i(:,1:3)R_i\right),
$$

and

$$
\begin{aligned}
\text{trace}\left(R_i^T H_i(:,1:3)^T H_i(:,1:3)R_i\right) &= \text{trace}\left(H_i(:,1:3)^T H_i(:,1:3)R_i R_i^T\right) \\
&= \text{trace}\left(H_i(:,1:3)^T H_i(:,1:3)\right),
\end{aligned}
$$

since R_i is orthogonal.

Hence Equation (13) can be rewritten as

$$
\begin{aligned}
||H_i(:,1:3)R_i - W_s(:,1:3)||_F^2 &= \text{trace}\left(H_i(:,1:3)^T H_i(:,1:3)\right) \\
&- 2\text{trace}\left(W_s(:,1:3)^T H_i(:,1:3)R_i\right) \\
&+ \text{trace}\left(W_s(:,1:3)^T W_s(:,1:3)\right), \qquad (14)
\end{aligned}
$$

where the first and third terms are fixed. Hence the optimization problem in (12) is equivalent to

$$
\max_{R_i \in \mathbb{R}^{3\times 3}} \text{trace}\left(W_s(:,1:3)^T H_i(:,1:3)R_i\right), \qquad (15)
$$

which can be solved exactly by computing the singular value decomposition [10] of the matrix $W_s(:,1:3)^T H_i(:,1:3)$. (Note that we have a maximization problem instead of a minimization problem, because of the negative sign in Equation (14).) More specifically, if

$$
W_s(:,1:3)^T H_i(:,1:3) = U\Sigma V^T
$$

is the singular value decomposition, for orthogonal matrices U, V and diagonal matrix Σ, then the optimal $R_i = VU^T$. The details can be found in [10, 15].

Note that by line 7,

$$
Z^s = \sum_{i=1}^{K} ||H_i Q_i^{s-1} - W_s||_F^2 \geq \sum_{i=1}^{K} ||H_i Q_i^s - W_s||_F^2 \geq \sum_{i=1}^{K} ||H_i Q_i^s - W_{s+1}||_F^2 = Z^{s+1}.
$$

The first inequality is because the Q_i^s's minimize $\sum_{i=1}^{K} ||H_i Q_i - W_s||_F^2$ and the second inequality is by Lemma 5.1. Thus $Z^{s+1} \leq Z^s$, so the algorithm converges.

Data Set	Proteins (PDB ID)
Set 1 (*Globin*)	`1mbc, 1mba, 1dm1, 1hlm, 2lhb, 2fal, 1hbg, 1flp, 1eca, 1ash`
Set 2 (*Thioredoxin*)	`3trx, 1aiu, 1erv, 1f9mA, 1ep7A, 1tof, 2tir, 1thx, 1quw, 1fo5A`
Set 3 (*all-alpha*)	`1le2, 2fha, 1nfn, 1grj`
Set 4 (*Alpha-beta*)	`1mek, 1a8l, 1f37B, 1ghhA`
Set 5 (*Globin*)	`1hlb, 1hlm, 1babA, 1babB, 1ithA, 1mba, 2hbg, 2lhb`
	`3sdhA, 1ash, 1flp, 1myt, 1eca, 1lh2, 2vhbA, 5mbn`
Set 6 (*all-beta*)	`1cd8, 1ci5A, 1qa9A, 1cdb, 1neu, 1qfoA`
Set 7 (*mixed*)	`1cnpB, 1jhgA, 1hnf, 1aa9, 1eca`

Table 1: The seven data sets used for the experiments

5.5 Complexity analysis

Let n be the maximum length of the K proteins. Line 1 of Algorithm 1 takes $O(K^2n^2)$ time if we choose the initial consensus protein to minimize the sum or the maximum of the pairwise distances, and $O(K)$ time if we choose it to be the protein of median length. Lines 3, 4 and 6 take $O(Kn^2)$, if we use our pairwise alignment algorithm from [24]. Let L be the length of the correspondence resulting from line 6. L is at most $O(Kn)$ (this happens if no two proteins overlap in the correspondence).

What is the time for line 7, which uses Algorithm 2? In each iteration of Algorithm 2, we spend $O(KL) = O(K^2n)$ time to compute $\{Q_i\}_{i=1}^K$ using the SVD. Therefore, line 4 takes $O(K^2nI)$ time, where I is the number of iterations in Algorithm 2.

Therefore each iteration of Algorithm 1 takes $O(Kn^2 + K^2nI)$ time. Let I' be the number of iterations of the loop in lines 2–10 of Algorithm 1. Then the total time taken by the algorithm is either $O(K^2n^2 + (Kn^2 + K^2nI)I')$ or $O(K + (Kn^2 + K^2nI)I')$, depending on the choice of the initial consensus. In practice, I and I' tend to be small constants, so the running time is either $O(K^2n^2)$ or $O(Kn^2 + K^2n)$.

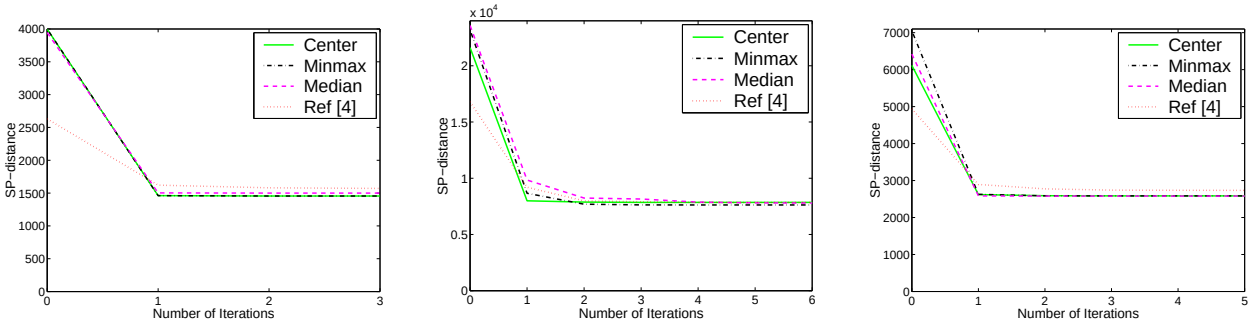
6 Experimental results

We implemented the algorithm in MATLAB and ran it on the seven data sets listed in Table 1. The ten proteins in Set 1 are from the *Globin* family, while the ten proteins in Set 2 are from *Thioredoxin* family. Set 3 contains four all-alpha proteins, which are structural neighbors of `1mbc` from the DALI database. The four alpha-beta proteins in Set 4 are all structural neighbors of `3trx` from the DALI Database. Sets 5–7 are from [4]: Set 5 contains sixteen proteins from the *Globin* family, Set 6 contains six all-beta proteins from the *immunoglobulin* family, and Set 7 contains five proteins that are unrelated.

The results of our experiments on the data from Table 1 are summarized in Table 2. We ran our algorithm with three different choices for the initial consensus: "center" refers to the choice that minimizes the sum of pairwise distances, "minmax" refers to the choice that minimizes the maximum pairwise distance, and "median" refers to the choice of the protein of median length. In each case, the SP distance and the number of iterations to convergence are shown. The convergence for three of the data sets in Table 1 is illustrated in Figure 1. As seen from the figure, the SP-distance in each case is reduced by a large amount within a few iterations. The experiments also

Data Set	Median		Center		Minmax		Ref.[4]	
	SP-Dist.	#Iters.	SP-Dist.	#Iters	SP-Dist.	#Iters	SP-Dist.	#Iters
Set 1	2325	3	2325	3	2321	3	2327	5
Set 2	1783	3	1776	3	1786	4	1803	3
Set 3	1068	3	1068	3	1064	3	1275	3
Set 4	1460	3	1460	3	1460	3	1571	3
Set 5	7850	3	7632	3	7674	6	7854	4
Set 6	1075	4	1048	3	1048	3	1066	3
Set 7	2634	3	2579	4	2579	3	2735	5

Table 2: Experimental results for the seven data sets in Table 1. The SP distance and number of iterations to convergence are shown for three different choices for the initial consensus. Also shown are the corresponding results for the algorithm in [4].



(a) (b) (c)

Figure 1: Illustrating convergence of the SP-distance for data sets 4 (graph (a)), 5 (graph (b)), and 7 (graph (c)) from Table 1. Each graph shows results for three different choices of the initial consensus (center, minmax, and median) and also results for the method from [4]. The x-axis is the number of iterations and the y-axis is the SP-distance.

---

show that the "median" method produces results that are comparable to the other two while being computationally less expensive. Table 2 also shows the results for the algorithm in [4]. (We obtained the code from the authors and added code to computes the SP distance and to track the number of iterations to convergence, using the same threshold $\eta = 0.001$ as for our algorithms.) As can be seen, the "median" method generally produces lower SP distances than [4] with fewer iterations.

We follow the approach in [4] to illustrate the consensus protein in Figures 2 and 3. The consensus is shown in the upper left corner and some or all of the proteins from the corresponding set are displayed below it. The graph in the upper right corner plots the length of each consensus vector (y-axis) as a function of its position along the backbone (x-axis). Lengths close to unity indicates that the protein vectors at that position agree.

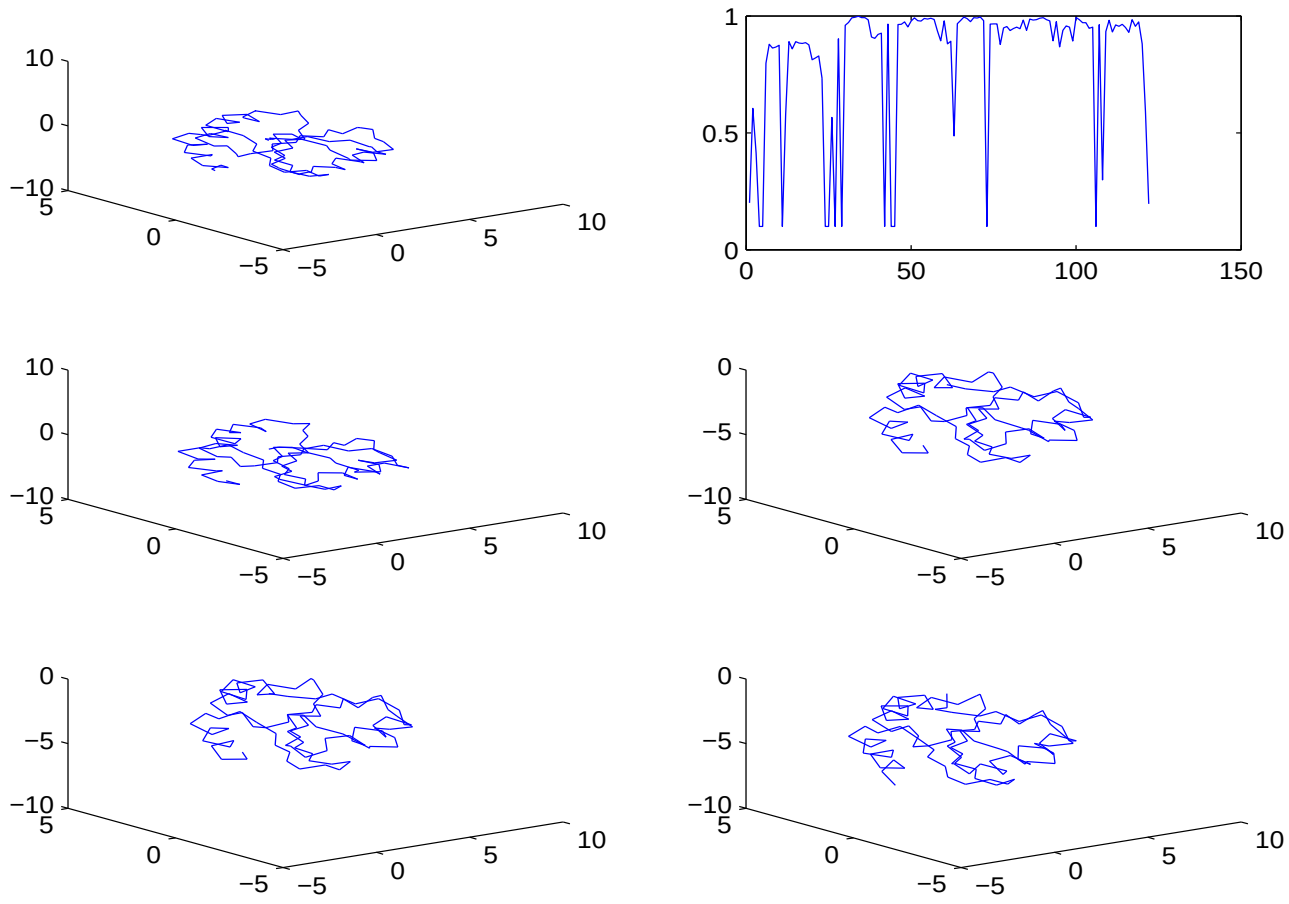


Figure 2: Illustrating the consensus structure for Set 2. The upper left corner is the consensus protein. Four of the proteins (`3trx, 1aiu, 1erv, 1ep7A`) from Set 2 are also shown from left to right in the second and third rows. The graph shows the length of each vectors of the consensus protein as a function of its position on the backbone.

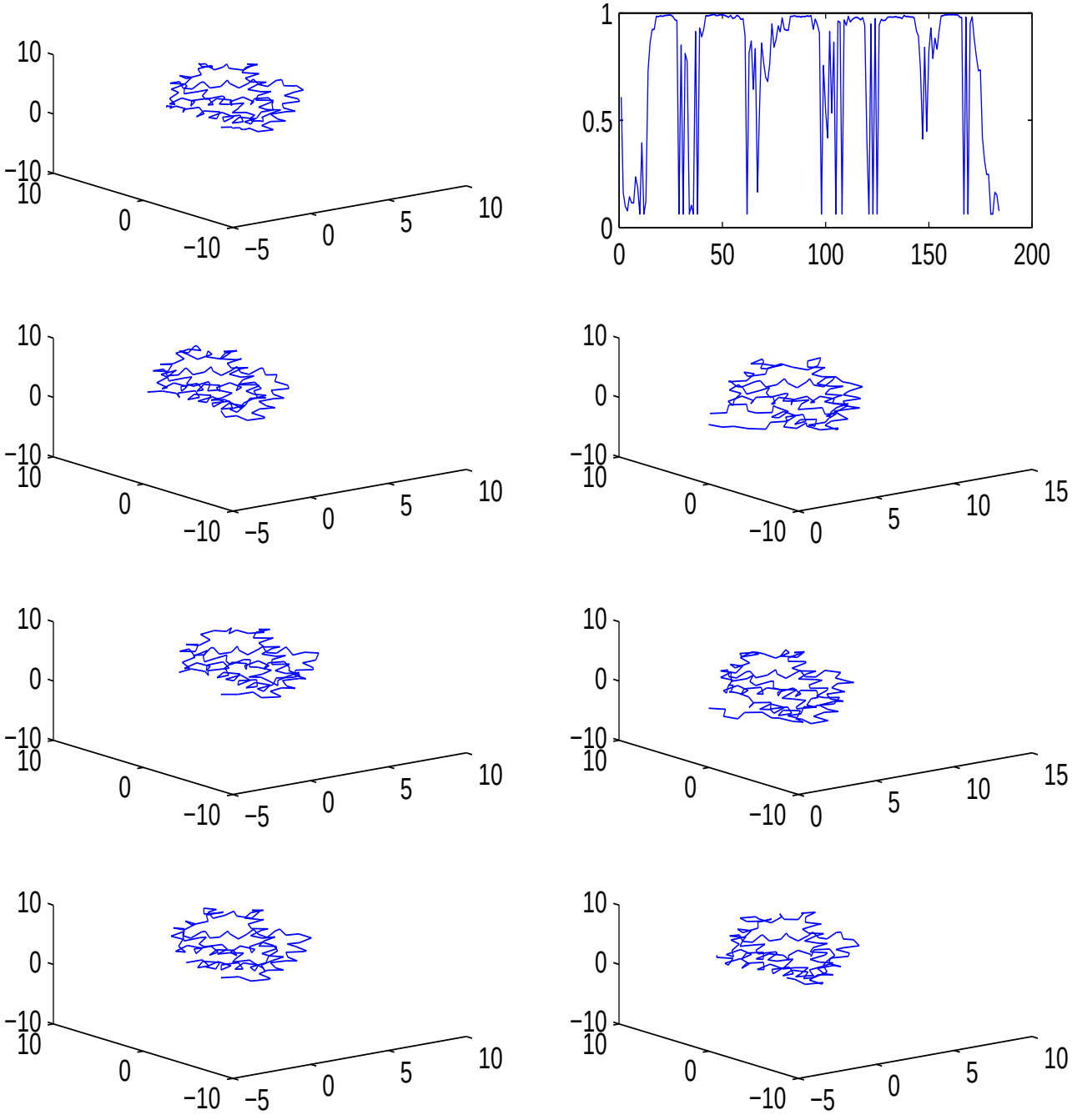


Figure 3: Illustrating the consensus structure for Set 5. The upper left corner is the consensus protein. Six of the proteins (`1hlb, 1hlm, 1babA, 1babB, 1ithA, 1mba`) from Set 5 are also shown. The graph shows the length of each vector of the consensus protein as a function of its position on the backbone.

7 Conclusions

We have presented an algorithm to compute a multiple structure alignment for a set of proteins, together with their consensus structure. Our algorithm uses a vector-based representation of the input proteins and the consensus and computes an approximation to the optimal multiple structure alignment. The algorithm iteratively uses a center-star-like method to compute a correspondence between the protein structures and then determines a set of optimal rotation matrices to align the structures and derive the new consensus. The computation of the optimal rotations is based on a result we establish that allows a compact representation of the objective function. Experimental results are also provided for several protein families.

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] S.R. Eddy. Multiple alignment using Hidden Markov Models. *Proceeding of the Third International Conference on Intelligent Systems for Molecular Biology*, 1995. pp. 114-120.

[6] D. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Biology*, 60, 1987, pp. 351–360.

[7] M. Gerstein and R. Altman. Average core structures and variability measures for protein families: application to the immunoglobulins. *Journal of Molecular Biology*, 251, 1995, pp. 161–175.

[8] M. Gerstein and M. Levitt. Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of protein structures. In *Proceedings of ISMB'96 Intelligent Systems for Molecular Biology*, 1996, pp. 59–66. AAAI Press.

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

[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: Computer Science and Computational Biology, Cambridge University Press, 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] N. Leibowitz, Z. Fligelman, R. Nussinov, and H. Wolfson: Multiple Structural Alignment and Core Detection by Geometric Hashing. *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, 1999, pp. 169–177.

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

[16] M. Murata, J.S. Richardson, J.L. Sussman. Simultaneous comparison of three protein sequences. *Proceedings of the National Academy of Sciences*, 82, 1985, pp. 3073–3077.

[17] C.A. Orengo. CORA-topological fingerprints for protein structure families. *Protein Science*, 8, 1999. pp. 699-715.

[18] C. Orengo and W. Taylor. SSAP: Sequential structure alignment program for protein structure comparison. *Methods in Enzymology*, 266, 1996, pp. 617–635.

[19] G. Rose. No assembly required. *The Sciences*, 36, 1996, pp. 26–31.

[20] M. Sela, F. H. White Jr, and C. B. Anfinsen. Reductive cleavage of disulfide bridges in Ribonuclease. *Science*, 125, 1957, pp. 691-692.

[21] 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.

[22] 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.

[23] L. Wang and T, Jiang. On the complexity of multiple sequence alignment. *Journal of Computational biology*, 1, pp. 337–348, 1994.

[24] J. Ye, R. Janardan, and S. Liu. Pairwise protein structure alignment based on an orientation-independent representation of the backbone geometry. Technical Report 03–001, University of Minnesota, Department of Computer Science and Engineering, 2003. (`http://www.cs.umn.edu/~jieping/Research`). Submitted.