FlexSnap: Flexible Non-sequential Protein Structure Alignment*

Saeed Salem[1], Mohammed J. Zaki[1], and Chris Bystroff[1,2]

[1] Department of Computer Science
salems@cs.rpi.edu, zaki@cs.rpi.edu
[2] Department of Biology,
Rensselaer Polytechnic Institute, Troy NY 12180, USA
bystrc@rpi.edu

Abstract. Proteins have evolved subject to energetic selection pressure for stability and flexibility. Structural similarity between proteins which have gone through conformational changes can be captured effectively if flexibility is considered. Topologically unrelated proteins that preserve secondary structure packing interactions can be detected if both flexibility and sequence permutations are considered. We propose the FlexSnap algorithm for flexible non-topological protein structural alignment. The effectiveness of FlexSnap is demonstrated by measuring the agreement of its alignments with manually curated non-sequential structural alignments. FlexSnap showed competitive results against state-of-the-art algorithms, like DALI, SARF2, MultiProt, FlexProt, and FATCAT.

1 Background

The wide spectrum of functions performed by proteins are enabled by their intrinsic flexibility [1]. It is known that proteins go through conformational changes to perform their functions. Homologous proteins have evolved to adopt conformational changes in their structure. Therefore, similarity between two proteins which have similar structures with one of them having undergone a conformational change will not be captured unless flexibility is considered.

The problem of flexible protein structural alignment has not received much attention. Even though there are a plethora of methods for protein structure comparison [2, 3, 4, 5, 6, 7, 8], the majority of the existing methods report only sequential alignments and thus cannot capture non-sequential alignments. Non-sequential similarity can occur naturally due to circular permutations [9] or convergent evolution [10]. The case is even harder for flexible alignment since only two methods, FlexProt [11], and FATCAT [12] report flexible alignments. Nevertheless, both methods are inherently limited to *sequential* flexible structural alignment because both methods employ sequential chaining techniques. The complexity of protein structural alignment depends on how the similarity is

* This work was supported in part by NSF Grants EMT-0829835, and CNS-0103708, and NIH Grant 1R01EB0080161-01A1.

S.L. Salzberg and T. Warnow (Eds.): WABI 2009, LNBI 5724, pp. 273–285, 2009.

assessed. [13] showed that the problem is NP-hard if the similarity score is distance matrix based. Therefore, over the years, a number of heuristic approaches have been proposed, which can mainly be classified into two main categories, dynamic programming and clustering.

Dynamic Programming (DP) is a general paradigm to solve problems that exhibit the optimal substructure property [14]. DP-based methods, STRUCTAL [15] and SSAP [16], construct a scoring matrix S, where each entry, S_{ij}, corresponds to the score of matching the i-th residue in protein A and the j-th residue in protein B. Given a scoring scheme between residues in the two proteins, dynamic programming finds the global alignment that maximizes the score. DP-based methods suffer from two main limitations: first, the alignment is sequential and thus non-topological similarity cannot be detected, and second, it is difficult to design a scoring function that is globally optimal [13]. In fact, structure alignment does not have the optimal substructure property, therefore DP-based methods can find only a suboptimal solution [17].

The other category of alignment methods, the Clustering-based methods, DALI [2], SARF2 [4], CE [5], SCALI [7], and FATCAT [12], seek to assemble the alignment out of smaller compatible (similar) element pairs such that the score of the alignment is as high as possible [18]. Two compatible element pairs are consistent (can be assembled together) if the substructures obtained by elements of the pairs are similar. The clustering problem is NP-hard [19], thus several heuristics have been proposed. The approaches differ in how the set of compatible element pairs is constructed and how the consistency is measured. Both SARF2 and SCALI produce non-sequential alignments.

The two main flexible alignment methods, FlexProt [11] and FATCAT [12], work by clustering (chaining) aligned fragment pairs (AFPs) and allowing flexibility while chaining, by introducing hinges (twists). FlexProt searches for the longest set of AFPs that allow different number of hinges. It then reports different alignments with different number of hinges. The FATCAT method works by chaining AFPs using dynamic programming. The score of an alignment ending with a given AFP is computed as the maximum score of connecting the AFP with any of alignments that end before the AFP. A penalty is applied to the score to compensate for gaps, root mean squared deviation (*rmsd*), and hinges. A third method, which can handle flexible alignments, is the HingeProt [20] method. HingeProt first partitions one of the two proteins into rigid parts using a Gaussian-Network-Model-based (GNM) approach and then aligns each rigid region with the other protein using the MultiProt [6] method. HingeProt uses the MultiProt algorithm in the sequential mode and thus does not report flexible non-sequential alignments. Therefore, the accuracy of the HingeProt approach depends on the accuracy of identifying the rigid domains which is a hard problem as the best known method ,HingeMaster [21], has a sensitivity of only 50%.

In this paper, we propose FlexSnap[1], a greedy algorithm for flexible non-sequential protein alignment. The algorithm assembles the alignment from the

[1] A non-sequential permutation of the bold letters in **Flex**ible **n**on-**S**equential **p**rotein **A**lignment

set of AFPs and allows non-sequential alignments and hinges. We demonstrate the effectiveness of FlexSnap by evaluating its alignments' agreement with manually curated non-sequential alignments.

2 Methods

The main idea of the FlexSnap approach is to assemble the alignment from short well-aligned fragment pairs, which are called AFPs. As we assemble the alignment by adding AFPs, we introduce hinges when necessary. Figure 1 shows how the alignment is constructed from smaller aligned fragment pairs. When chaining a fragment pair to the alignment, we choose the fragment that has the highest score when joined with the last rigid region in the alignment. The score rewards longer alignments with small *rmsd* and penalizes large *rmsd*, gaps, and the introduction of hinges. In the next subsections, we provide a detailed discussion of the FlexSnap algorithm.

2.1 AFPs Extraction

Let $A = \{A_1, A_2, \ldots, A_n\}$ and $B = \{B_1, B_2, \ldots, B_n\}$ be two proteins with n and m residues respectively, and $A_i \in \Re^{3\times 1}$ (similarly B_i) represents the 3D coordinates of the C_α atom of the i-th residue in protein A. The first step in FlexSnap is to generate a list of aligned fragment pairs (AFPs):

$$AFPs = \{(i, j, l) |\ rmsd(i, j, l) \leq \epsilon\}$$

Each AFP, (i, j, l), is a fragment that starts at the i-th residue in A and j-th residue in B and it has a length of l residues. An AFP is formally represented as a set of l equivalenced pairs between the two proteins, and given as:

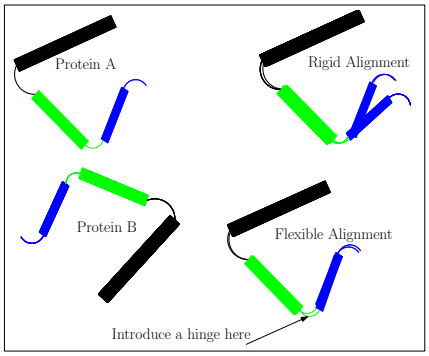

Fig. 1. Flexible Structural Alignment by chaining. The figure shows protein A and protein B which have 3 similar structure fragments. A rigid alignment (top right) is not able to align the blue fragment, but a flexible alignment (bottom right) can do this easily by introducing a hinge between the rigid block (the black and green fragments) and the blue fragment. As we assemble the alignment from well-aligned pairs, we introduce hinges to get a longer alignment and smaller *rmsd*.

$$(i, j, l) = \{(A_i, B_j), (A_{i+1}, B_{j+1}), \cdots, (A_{i+l-1}, B_{j+l-1})\}$$

where (A_i, B_j) indicates that the i^{th} residue of protein A is paired with the j^{th} residue of protein B, and l is AFP's length. Each AFP must satisfy a user-defined similarity constraint. In FlexSnap, approach, we employ the root mean square deviation as the similarity measure, i.e., $rmsd(i, j, l) \leq \epsilon$. Moreover, we require that the length of the AFP be at least L, i.e., $3 \leq L \leq l$. Furthermore, we define b_k^P and e_k^P to be the beginning and end of the AFP_k along the backbone of protein B. e.g, for a triplet $\text{AFP}_k = (i, j, l)$ and protein A, $b_k^A = i$ and $e_k^A = i + l - 1$.

The number of possible AFPs can be as large as $O(n^3)$. The set of all AFPs can be obtained by iterating over all the triplets (i, j, l),

$$\text{where} \begin{bmatrix} i & = & 1 \cdots n - L \\ j & = & 1 \cdots m - L \\ L \leq l \leq min(n - i + 1, m - j + 1) \end{bmatrix}$$

and for each triplet checking if the $rmsd(i, j, l) \leq \epsilon$. The *rmsd* of a fragment can be obtained in $O(l)$ [22]. A naive implementation that iterates over all the triplets (i, j, l) to obtain the set of all the AFPs would have an $O(n^4)$ time complexity. However, by observing that the *rmsd* of the AFP $(i, j, l + 1)$ can be computed incrementally from the *rmsd* of AFP (i, j, l) in constant time, the set of aligned fragment pairs (AFPs) can be obtained in $O(n^3)$ time complexity [11].

The main idea to incrementally compute the *rmsd* is to simplify the *rmsd* formula. Given two sets, A and B, of N points each, the root mean square deviation (*rmsd*) is calculated as [23]:

$$rmsd^2 = \frac{1}{N} \times \left(\sum_{i=1}^{N} {A'}_i^2 + \sum_{i=1}^{N} {B'}_i^2 - 2 \sum_{i=1}^{3} d_i \right) \tag{1}$$

where A' (similarly B') is the points after recentering, i.e., $A'_i = A_i - \frac{\sum_{i=1}^{N} A_i}{N}$, and the d_i's are the singular values of $C = A'B'^T$, which is a 3×3 covariance matrix given as:

$$C = \sum_{i=1}^{n} A_i B_i^T - \frac{\sum_{i=1}^{N} A_i \sum_{i=1}^{n} B_i^T}{N} \tag{2}$$

In rare cases when the determinant of C is negative, then $d_3 = -1 * d_3$. Equation 1 can be simplified as:

$$rmsd^2 = \frac{1}{N} \times \left(\sum_{i=1}^{N} A_i^2 - \frac{1}{N} \left(\sum_{i=1}^{N} A_i \right)^2 + \sum_{i=1}^{N} B_i^2 - \frac{1}{N} \left(\sum_{i=1}^{N} B_i \right)^2 - 2 \sum_{i=1}^{3} d_i \right)$$

It is clear that all the terms used in the *rmsd* computation can be updated in constant time and thus computing the *rmsd* for $N + 1$ points requires constant

time if we have all the terms evaluated for the first N points. Therefore computing the *rmsd* for AFP(i, j, l) for all values of l's requires only $O(n)$ time. Thus, the total time complexity for the seeds extraction step is $O(n^3)$.

2.2 Flexible Chaining

The second step in FlexSnap is to construct the alignment by selecting a subset of the AFPs. Given a set of AFPs, P, obtained in the AFPs extraction step, we are interested in finding a subset of AFPs, $R \subseteq P$, such that all the AFPs in R are mutually non-overlapping and the score of the selected AFPs in R is as large as possible. At one hand, we want to get as large an alignment as possible, while on the other hand, we want to minimize the number of hinges and gaps. Therefore, our goal is to optimize a score that rewards long alignments with small *rmsd* and penalizes the introduction of hinges and gaps.

The set of AFPs can be thought of as runs in an $n \times m$ matrix S, where n and m are the sizes of proteins A and B, respectively (see Figure 2). We define a precedence relation, $\prec$, between two AFPs such that $P_i \prec P_j$ if P_i appears either in the upper or lower left quadrant of P_j, i.e. $b_j^A > e_i^A$ and $b_j^B > e_i^B$, or $e_j^A < b_i^A$ and $b_j^B > e_i^B$ (recall that b_i^A (e_i^A) denote beginning (end) of AFP P_i in protein A). Generally speaking, we say that two AFPs, P_i and P_j, can be chained if they do not overlap, i.e., $P_i \prec P_j$ or $P_j \prec P_i$. In Figure 2, P_7 and P_8 can be chained to P_1. For sequential chaining, we define a sequential precedence relation, $\prec_s$, such that P_i precedes P_j (written as $P_i \prec_S P_j$) if P_i appears strictly in the upper left quadrant with respect to P_j, i.e. $b_j^A > e_i^A$ and $b_j^B > e_i^B$. Two AFPs P_i and P_j

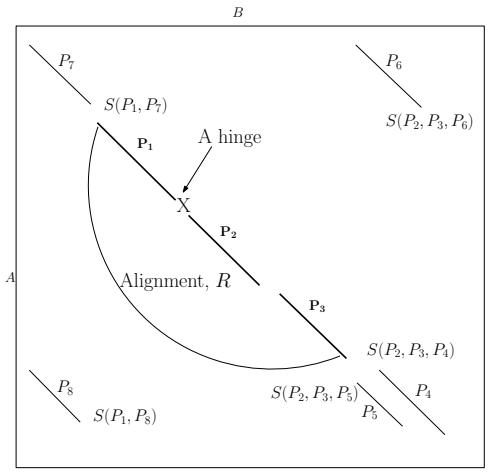

Fig. 2. Flexible Structural Alignment by chaining AFPs. When extending the alignment $R = \{P_1, P_2, P_3\}$, the score of adding each AFP is computed computed and we extend the alignment by the AFP that gives the best score. The score $S(P_4, P_2, P_3))$ indicates the score of adding P_4 to the region composed of P_2 and P_3.

can be sequentially chained together if $P_i \prec_S P_j$ or $P_j \prec_S P_i$. In Figure 2, P_7 and P_2 can be sequentially chained to P_1.

An AFP, P_i, can be chained to an alignment R, denoted as $(R \rightarrow P_i)$, if it does not overlap with any AFP in R. In Figure 2, P_7, P_4, and P_5 can be sequentially chained to R which consists of AFPs $\{P_1, P_2, P_3\}$; and both P_6 and P_8 can be non-sequentially chained to R. Next, we shall introduce our solution for the general flexible chaining problem.

2.3 The FlexSnap Approach

The goal of chaining is to find the highest scoring subset of AFPs, i.e., $R \subseteq P$, such that all the AFPs in R are mutually consistent and non-overlapping. The problem of finding the highest scoring subset of AFPs is essentially the same as finding the maximum weighted clique in a graph $G = (V, E, w)$ where the set of vertices V represent the set of AFPs, each vertex v_i has a weight equal to the score of the AFP, $w(v_i) = S(P_i)$, where the score of an AFP P_i, $S(P_i)$, could be its length or some other combination of length and *rmsd*. There is an edge $(v_i, v_j) \in E$ if the AFPs P_i and P_j do not overlap and are consistent (can be joined with small *rmsd* or have similar rotation matrices). The problem of finding the maximum weighted clique in a graph is computationally expensive; it is NP-hard [19].

We propose a greedy algorithm to find an approximate solution for the chaining problem. The main idea is to start building the alignment from an initial AFP and add AFPs to the alignment. We start the alignment by selecting the longest AFP, then we iteratively add new AFPs to the alignment as long as the newly added AFP improves the score of the alignment. Given an alignment, R, we add to it the AFP that contributes most. We keep growing the alignment until no more AFPs can be added. The contribution of an AFP to the alignment is scored by how consistent the AFP is with the alignment and how good the AFP is. When adding an AFP to an alignment, we reward longer AFPs with smaller *rmsd*, and we penalize for gaps, inconsistency, and hinges. The penalty takes into consideration: 1) the number of gaps introduced; 2) the increase in *rmsd* when combining two or more AFPs; 3) the introduction of new hinges.

As depicted in Figure 2, the scores of extending the alignment, R, with P_4, P_5, P_6, P_7, or P_8 are computed and the AFP with the best score is added to the alignment. When measuring the score of adding an AFP to the alignment, we actually measure the score of adding the AFP to the last rigid region in the alignment. In Figure 2, the score of adding P_4 to R is the score of adding P_4 to the region composed of P_2 and P_3. Since P_2 and P_3 together form a rigid sub-alignment (as we can see there is no hinge between them). When adding P_7 to R, the score of adding P_7 to the region composed only of P_1 is computed.

Figure 3 shows the pseudo-code for the greedy chaining algorithm used in FlexSnap. Since the chaining is a greedy algorithm, we run it K times starting from the longest non-overlapping K AFPs and we report the longest alignment. Next, we will discuss how we extend the alignment with the best AFP. More specifically, given an alignment R, the next AFP to chain to the alignment is the one that maximizes the following scoring function:

GreedyChaining($A, B, L, H, \epsilon, Dc, M_r, M_g$)
A, B: the two proteins to be aligned
L: the minimum length of an AFP, $L \geq 3$
ϵ: the maximum *rmsd* for an AFP
D_c: the *rmsd* for introducing a hinge
M_r: the penalty for a hinge
M_g: the penalty for a gap
H: the maximum number of hinges allowed
1.P= *seedExtraction*(A, B, L, ϵ)
2.P'= longest AFP in P
3.$R = P'$
4.**While**(R can be extended)
5.$P' \leftarrow \max_{P_i}(S(R, P_i))$
6.$R \leftarrow R \cup \{P'\}$
7.**End**

Fig. 3. A greedy algorithm for AFP chaining. The algorithm iteratively chooses an AFP to add to R (lines 5-6) until no more AFPs can be added, or the best score of adding an AFP to R is negative.

$$P' = \max_{\forall P_i, s.t. R \rightarrow P_i}(S(R, P_i)) \tag{3}$$

where $R \rightarrow P_i$ indicates that P_i does not overlap with R, and $S(R, P_i)$ is the score of chaining P_i to R. The score, $S(R, P_i)$, is a combination of the weight of the AFP, $W(P_i)$, and the penalty of extending R with P_i, $C(R \rightarrow P_i)$. The score is defined as follows:

$$S(R, P_i) = W(P_i) + C(R \rightarrow P_i) \tag{4}$$

where $C(R \rightarrow P_i)$ is the penalty incurred when connecting P_i to R, and $W(P_i)$ is the score of the AFP itself. The scoring function rewards longer AFPs with small *rmsd* and penalize gaps and hinges. If the addition of an AFP P_i to the alignment results in a large *rmsd*, then we introduce a hinge only if $W(P_i)$ is large enough to compensate for the penalty incurred. A similar approach for penalizing gaps and hinges was used in the FATCAT method [12]. Though their score and cost functions are different, and they do not consider rigid regions as we do in FlexSnap when connecting an AFP to the alignment. The score of connecting P_i to R is defined as follows:

$$C(R \rightarrow P_i) = M_r * Z(D_{RP_i}) + M_g * gap \tag{5}$$

$$\text{where } Z(D_{RP_i}) = \left\{ \begin{array}{cc} 1 & if D_{RP_i} > D_c \\ \left(\frac{D_{RP_i} - \epsilon}{D_c - \epsilon}\right)^2 & if \epsilon < D_{RP_i} < D_c \\ 0 & otherwise \end{array} \right\}$$

where M_g is the penalty for a gap, M_r is the maximum penalty for a hinge, and D_{RP_i} is the *rmsd* of connecting P_i to the last rigid region in R. If D_{RP_i} increases

above a user-defined threshold, D_c, we introduce a hinge and the penalty is maximum; if not, the penalty is proportional to how far the *rmsd* value is from ϵ (maximum *rmsd* for an AFP). Moreover, we allow only a maximum number of H hinges. The score for an AFP is a function of its length and *rmsd*. The score is the length of the AFP, $L(P_i)$, plus a contribution of the *rmsd* of the AFP, $rmsd(P_i)$, to the score, and is given as:

$$W(P_i) = L(P_i) + \alpha \ * L(P_i) * \left(\frac{\epsilon - rmsd(P_i)}{\epsilon}\right)^2 \tag{6}$$

The complexity of the chaining algorithm depends on the number of AFPs,M, that two structures have. In the worst case, M could be close to n^3, but in practice it is much less, i.e., $M \leq n^2$. The complexity of the algorithm is $Mlog(M) + k * M * n$, where k is the number of AFPs in the final solution and n is the size of the larger protein.

2.4 Sequential Flexible Chaining

The above general chaining algorithm reports both sequential and non-sequential alignments. In the results section, we show the quality of its alignments when compared to state-of-the-art non-sequential alignment methods. However, for sequential flexible alignment, there are more efficient chaining algorithms, namely FlexProt and FATCAT. The FATCAT algorithm follows a dynamic programing approach for chaining the AFPs. In FATCAT, the score of an alignment ending with AFP P_i is defined in terms of the score of P_j's and the connection cost of P_i with these P_j's such that $P_j \prec_s P_i$. More specifically, FATCAT defines the score of the alignment that ends with P_i as follows:

$$S(P_i) = W(P_i) + \max_{\forall P_j, s.t. P_j \prec_s P_i} \{max(S(P_j) + C(P_j \rightarrow P_i), 0)\}$$

where $C(P_j \rightarrow P_i)$ is the penalty incurred when connecting P_i to the alignment that ends with P_j and it is similar to the penalty function used in the general chaining and $W(P_i)$ is the score of the AFP itself. In FATCAT, $C(P_j \rightarrow P_i)$ is the connection cost of P_i and P_j. If P_j belongs to a rigid region and the connection cost of P_i with P_j is small, we will add P_i to the same rigid region as P_j even though P_i might not be consistent with other AFPs in the same region. In Figure 2, if we were connecting P_4 to P_3, FATCAT would compute the connection cost $C(P_3, P_4)$ while it makes more sense to compute $C((P_2, P_3) \rightarrow P_4)$. Moreover, $W(P_i)$ is a function of the length of P_i and its *rmsd* and thus $S(P_i)$ cannot be optimal because we do not know of a scoring function that involves the *rmsd* value that is additive and optimal (*rmsd* score is not a metric since it does not satisfy the triangle inequality property). Therefore, the optimality of FATCAT alignments is not guaranteed since the sub-optimality property of the dynamic programming does not hold if the score incorporates an *rmsd* term.

Our approach for sequential chaining is essentially the same as the FATCAT algorithm with the exception that we consider connecting an AFP to the last

rigid region in the alignment, not to the last AFP in the alignment as is the case in FATCAT. Moreover, we use a simpler function for scoring an AFP, e.g., $W(P_i) = length(P_i)$. In the results section, we investigate how these two modifications to the FATCAT algorithm would affect the performance of sequential chaining. Since we do not have the exact FATCAT scoring function because some terms are not adequately defined in the paper, we implemented our own and compared the results when including all the AFPs in the last rigid region or not including them.

3 Results and Discussion

To assess the quality of FlexSnap alignment compared to other structural alignment methods, we evaluated the agreement of the methods' alignments with reference manually-curated alignments. We compared our FlexSnap against sequential methods (DALI [2] and CE [5]), non-sequential methods (SARF2 [4], MultiProt [6], and SCALI [7]), and flexible sequential alignment methods (FlexProt [11] and FATCAT [12]). All the experiments were run on a 1.66 GHz Intel Core Duo machine with 1 GB of main memory running Ubuntu Linux. The chaining algorithm is efficient and its running time varies from 1 second to a minute depending on the size of the proteins. We used the corresponding web server for most of the other alignment methods. The optimal values for the different parameters were found empirically such that they give the best agreement with manually curated alignments; we used $L = 8$, $\epsilon = 2\mathring{A}$, $D_c = 3\mathring{A}$, $\alpha = 0.3$, $M_r = -10$, $M_g = -1$, and $H = 3$ (see Figure 3).

3.1 Non-sequential Alignments

We used the reference alignments for the structure pairs which have circular permutation in the RIPC dataset [24]. The RIPC set contains 40 structurally related protein pairs which are challenging to align because they have indels, repetitions, circular permutations, and show conformational flexibility [24]. There are 10 pairs in the RIPC dataset that have circular permutation. Since the structure pairs have non-sequential alignments, to be fair, we only compare with algorithms that can handle non-sequentiality. However, we report the average agreement for some sequential methods as well. The agreement of a given alignment, S, with the reference alignment, R, is defined as the percentage of the residue pairs in the alignment which are identically aligned as in the reference alignment (I_S) relative to the reference alignment's length (L_R), i.e., $A(S, R) = (I_S/L_R) \times 100$. Table 1 shows the agreements of four different methods with the reference alignments in the RIPC dataset. The results show that FlexSnap is competitive to state-of-the-art methods in non-sequential alignment. In fact, it has the highest average agreement (79%) among the methods shown. The average agreement of most of the sequential alignment methods, we compared with, were drastically lower: DALI [2] (40%), CE [4](36%), FATCAT [12](28%), and LGA [25](38%).

FlexSnap alignments have 100 percent agreement on four structure pairs. One such pair is the alignment of NK-lysin (1nkl, 78 residues) with prophytepsin (1qdm,

Table 1. Comparison of SARF, MultiProt, and FlexSnap on the RIPC dataset. Three values are reported for each alignment: its length, its *rmsd*, and A which is its agreement with the reference alignment in the RIPC dataset.

SCOPID		SARF			MultiProt			SCALI			FlexSnap		
Pro1	Pro2	size	*rmsd*	A	size	*rmsd*	A	size	*rmsd*	A	size	*rmsd*	A
d1nkl_	d1qdma1	67	2.21	92	67	1.82	68	62	1.94	69	73	2.39	100
d1nls_	d2bqpa_	212	1.50	83	213	1.03	100	195	1.62	83	210	2.81	83
d1qasa2	d1rsy_	109	2.27	65	107	1.24	93	98	1.92	82	111	1.73	100
d1b5ta_	d1k87a2	171	2.63	63	144	2.04	0	159	3.38	0	177	2.99	50
d1jwyb_	d1puja_	115	2.43	83	108	1.81	92	110	4.60	83	116	2.61	92
d1jwyb_	d1u0la2	97	2.02	100	103	1.86	91	91	4.52	90	96	2.82	100
d1nw5a_	d2adma_	129	2.52	85	130	2.11	92	132	3.73	84	128	2.91	100
d1gsa_1	d2hgsa1	73	2.59	20	74	1.56	40	69	3.23	40	73	2.81	20
d1qq5a_	d3chy_	88	2.39	67	82	1.97	67	52	2.08	66	93	2.94	67
d1kiaa_	d1nw5a_	146	2.48	83	153	1.85	75	138	3.99	75	141	2.69	75
Avg.	Agreement			74			72			67			79

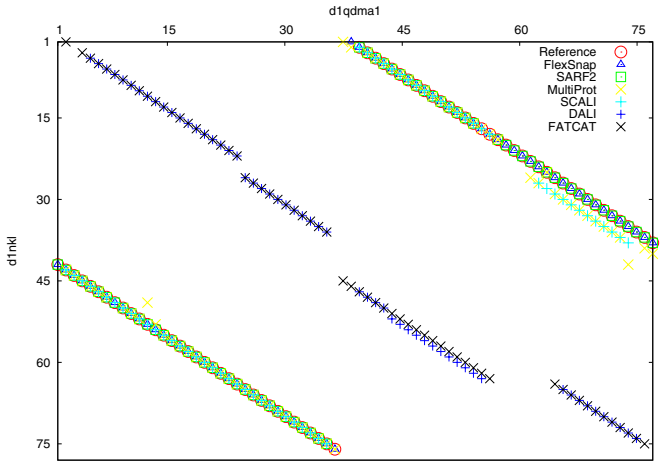

Fig. 4. Comparison of the agreement between the reference alignment and 6 other alignment methods on the structure pair of prophytepsin(d1qdma1) and nk-lysin(d1nkl_). Residue positions of d1qdma1 and d1nkl_ are plotted on the x-axis and y-axis, respectively. Note: The reference alignment pairs are shown in circles. The SARF, MultiProt, SCALI, and FlexSnap plots overlap with the reference alignment. FlexSnap has 100 percent coverage of the reference alignment; there is a triangle in every circle.

chain A, 77 residues). On this pair, all the sequential alignment methods(CE, DALI, FATCAT, and LGA) returned zero agreements. For the non-sequential ones: SARF returned 92%, MultiProt got 68%, and SCALI returned 69%. The reference

Table 2. Comparison of FlexProt, FATCAT, FlexSnap^F, and FlexSnap^{F_2}. Each alignment is reported in the following format: length, *rmsd*, and T which is the number of hinges introduced.

		FlexProt			FATCAT			FlexSnap^F			FlexSnap^{F_2}		
Pro1	Pro2	size	*rmsd*	T	size	*rmsd*	T	size	*rmsd*	T	size	*rmsd*	T
1wdnA(223)	1gggA(220)	218	0.94	2	220	1.01	2	220	0.96	2	220	0.96	2
1hpbP(238)	1gggA(220)	220	2.34	2	213	1.59	2	211	1.67	2	210	3.88	1
2bbmA(148)	1cll_(144)	139	2.22	1	144	2.28	1	138	1.8	1	138	1.80	1
2bbmA(148)	1top_(162)	147	2.40	3	145	2.28	3	137	1.78	3	137	1.78	3
1akeA(214)	2ak3A(226)	200	2.44	2	202	1.54	2	207	2.05	2	206	6.72	1
2ak3A(226)	1uke_(193)	182	2.90	2	188	2.97	0	184	2.36	1	184	3.08	0
1mcpL(220)	4fabL(219)	218	1.93	1	217	1.40	1	217	1.49	1	217	1.49	1
1mcpL(220)	1tcrB(237)	212	2.33	1	213	2.20	1	202	2.3	1	200	2.38	1
1lfh_(691)	1lfg_(691)	691	1.41	2	686	0.89	2	688	0.99	2	688	0.99	2
1tfd_(294)	1lfh_(691)	291	1.98	2	290	1.37	2	287	1.89	2	283	1.41	2
1b9wA(91)	1danL(142)	75	2.78	1	80	2.39	2	82	2.25	2	83	2.7	2
1qf6A(641)	1adjA(420)	323	4.43	1	351	2.68	1	326	2.45	3	320	2.47	2
2clrA(275)	3fruA(269)	253	2.71	2	245	3.06	0	254	2.57	3	252	4.31	0
1fmk_(438)	1qcfA(450)	424	1.25	2	433	2.27	0	413	2.71	0	413	2.44	1
1fmk_(438)	1tkiA(321)	231	3.28	2	238	3.07	0	241	2.58	3	242	3.14	2
1a21A(194)	1hwgC(191)	163	2.75	4	153	3.16	1	156	2.35	3	155	3.79	2

alignment had 72 aligned pairs. As shown in Figure 4, the sequential alignment methods (only DALI and FATCAT shown) have their alignment paths along the diagonal and do not agree with with the reference alignment (shown as circles).

3.2 Sequential Flexible Alignments

Table 2 shows the alignments of different methods on the FlexProt dataset [11] which is obtained from the database of macromolecular motions [26]. FlexSnap^F is our implementation of FATCAT with a simpler function for the score of AFP and a different function for the connection cost of two AFPs. In this version, $C(P_j \rightarrow P_i)$ calculates the connection cost of P_i with the rigid region to which P_j belongs. In the second version, FlexSnap^{F_2}, $C(P_j \rightarrow P_i)$ calculates the connection cost of P_i with only P_j. It is obvious that when considering the entire rigid region, we get much better alignments. Moreover, FlexSnap^F gives comparable results to the FATCAT method. In few cases it got slightly shorter alignments with much better *rmsd* as in the case of the third and fourth alignment pairs.

4 Conclusion

We have introduced FlexSnap, a chaining algorithm that reports both sequential and non-sequential alignments and allows twists (hinges). We assessed the

quality of the FlexSnap alignments by measuring its agreements with manually curated non-sequential alignments. Moreover, we employed the scoring function devised in FlexSnap in a FATCAT-like algorithm for sequential flexible alignments. The new algorithm for flexible sequential alignment, FlexSnap^F, gave competitive results against state-of-the-art flexible sequential alignment methods: FlexProt and FATCAT. Our future goal is to compile a list of manually curated flexible non-sequential alignments and measure the agreement of FlexSnap alignments with this dataset. Moreover, we would like to apply the algorithm on SCOP [27] and CATH [28] classifications to investigate how the introduction of flexibility would change the classification of some proteins.

References

[1] Wriggers, W., Schulten, K.: Protein domain movements: detection of rigid domains and visualization of hinges in comparisons of atomic coordinates. Proteins: Structure, Function, and Genetics 29, 1–14 (1997)

[2] Holm, L., Sander, C.: Protein structure comparison by alignment of distance matrices. J. Mol. Biol. 233(1), 123–138 (1993)

[3] Subbiah, S., Laurents, D.V., Levitt, M.: Structural similarity of dna-binding domains of bacteriophage repressors and the globin core. Curr. Biol. 3, 141–148 (1993)

[4] Alexandrov, N.N.: Sarfing the pdb. Protein Engineering 50(9), 727–732 (1996)

[5] Shindyalov, I.N., Bourn, P.E.: Protein structure alignment by incremental combinatorial extension (ce) of the optimal path. Protein Eng. 11, 739–747 (1998)

[6] Shatsky, M., Nussinov, R., Wolfson, H.J.: A method for simultaneous alignment of multiple protein structures. Proteins: Structure, Function, and Bioinformatics 56(1), 143–156 (2004)

[7] Yuan, X., Bystroff, C.: Non-sequential structure-based alignments reveal topology-independent core packing arrangements in proteins. Bioinformatics 21(7), 1010–1019 (2003)

[8] Zhu, J., Weng, Z.: Fast: A novel protein structure alignment algorithm. Proteins:Structure, Function and Bioinformatics 14, 417–423 (2005)

[9] Lindqvist, Y., Schneider, G.: Circular permutations of natural protein sequences: structural evidence. Curr. Opin. Struct. Biol. 7(3), 422–427 (1997)

[10] Milik, M., Szalma, S., Olszewski, K.A.: Common structural cliques: a tool for protein structure and function analysis. Protein Engineering 16(8), 543–552 (2003)

[11] Shatsky, M., Nussinov, R., Wolfson, H.J.: Flexible protein alignment and hinge detection. Proteins: Structure, Function, and Bioinformatics 48, 242–256 (2002)

[12] Ye, Y., Godzik, A.: Flexible structure alignment by chaining aligned fragment pairs allowing twists. Bioinformatics 19, II246–II255 (2003)

[13] Kolodny, R., Linial, N.: Approximate protein structural alignment in polynomial time. PNAS 101, 12201–12206 (2004)

[14] Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48, 443–453 (1970)

[15] Gerstein, M., Levitt, M.: Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of protein structures. In: Proc. Int. Conf. Intell. Syst. Mol. Biol., vol. 4, pp. 59–67 (1996)

[16] Orengo, C.A., Taylor, W.R.: Ssap: sequential structure alignment program for protein structure comparison. Methods Enzymol. 266, 617–635 (1996)

[17] Eidhammer, I., Jonassen, I., Taylor, W.R.: Protein Bioinformatics: An algorithmic Approach to Sequence and Structure Analysis. John Wiley & Sons Ltd., UK (2004)

[18] Eidhammer, I., Jonassen, I., Taylor, W.R.: Structure comparison and structure patterns. J. Comput. Biol. 7(5), 685–716 (2000)

[19] Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of np-completeness. W.H. Freeman, San Francisco (1979)

[20] Emekli, U., Schneidman-Duhovny, D., Wolfson, H.J., Nussinov, R., Haliloglu, T.: Hingeprot: Automated prediction of hinges in protein structures. Proteins 70(4), 1219–1227 (2008)

[21] Flores, S.C., Keating, K.S., Painter, J., Morcos, F., Nguyen, K., Merritt, E.A., Kuhn, L.A., Gerstein, M.B.: Hingemaster: normal mode hinge prediction approach and integration of complementary predictors. Proteins 73, 299–319 (2008)

[22] Kabsch, W.: A solution for the best rotation to relate two sets of vectors. Acta Crystallogr. A32, 922–923 (1976)

[23] Chwartz, J.T., Sharir, M.: Identification of partially obscured objects in two dimensions by matching of noisy characteristic curves. Int. J. Robotics Res. 6, 29–44 (1987)

[24] Mayr, G., Dominques, F., Lackner, P.: Comparative analysis of protein structure alignments. BMC Structural Biol. 7(50), 564–577 (2007)

[25] Zemla, A.: Lga - a method for finding 3d similarities in protein structures. Nucleic Acids Research 31(13), 3370–3374 (2003)

[26] Gerstein, M., Krebs, W.: A database of macromolecular motions. Nucleic Acids Res. 26(18), 4280–4290 (1998)

[27] Murzin, A., Brenner, S.E., Hubbard, T., Chothia, C.: Scop: A structural classification of proteins for the investigation of sequences and structures. J. Mol. Biol. 247, 536–540 (1995)

[28] Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B., Thornton, J.M.: Cath- a hierarchic classification of protein domain structures. structure 5(8), 1093–1108 (1997)