Formatt: Correcting Protein Multiple Structural Alignments by Sequence Peeking

An Honors Thesis for the Department of Computer Science

Shilpa Nadimpalli

Tufts University, 2011

Acknowledgements

Thanks to my advisor, Professor Lenore Cowen, for her ideas, suggestions, motivation, and guidance throughout the entire project. Thanks to Noah Daniels for significant software engineering advice and debugging assistance, and also thanks to Matt Menke for expert help with the Matt code base. Portions of this thesis appear in the paper *"S. Nadimpalli, N. Daniels, L. Cowen (2011). Formatt: Correcting Protein Multiple Structure Alignments by Sequence Peeking. Proceedings of the 2011 ACM Conference on Bioinformatics, Computational Biology, and Biomedicine."* I thank my co-authors for permission to include our joint work in this thesis. I would also like to thank my thesis defense committee members, Professor Juliet Fuhrman of the Tufts biology department and Professors Lenore Cowen and Ben Hescott of the Tufts computer science department, for their time and feedback. This work was funded in part by NIH grant 1R01GM08033001A1 (to Lenore Cowen).

Table of Contents

Abstract

Multiple protein alignments have several computational biology applications, including detection of protein homology, identification of conserved regions, or construction of HMMs as templates for protein families. Although it has been shown that structure-based methods produce better multiple protein alignments than pure sequence aligners, structural aligners that ignore all sequence information are prone to making easily-avoidable frame offset errors. We present Formatt, a multiple protein structural alignment program that also takes sequence similarity into account when constructing alignments. Formatt is based on the Matt purely geometric multiple structural alignment program. We show that Formatt is superior to Matt in alignment quality based on objective measures (most notably Staccato *Seq* and *Str* scores) while preserving the same advantages in core length and RMSD that Matt, as a flexible structural aligner, has as compared to other multiple structural alignment programs on popular benchmark datasets. Applications include producing better training data for threading methods.

1 Introduction

A classical problem in computational biology is the construction of multiple protein alignments, applications of which include detection of protein homology and inference of protein function, development of Hidden Markov Models as templates for protein families [28], identification of highly conserved or highly divergent regions in proteins, and others. Because of the widespread applications of multiple homologous protein alignments, coming up with the "best" or "most correct" alignment has been the goal of several alignment programs. The major approaches to constructing these alignments can be grouped into two main categories: first, sequence alignment programs, which typically have access only to and therefore use solely protein sequence information; and second, structural alignment programs, which, although they have access to both protein sequence and its 3D structure, typically utilize only protein structure. As has been demonstrated by Kim and Lee [9], structure-based methods produce better sequence alignments than methods based on sequence information alone, likely because structure is more conserved over a longer evolutionary distance than sequence is. But of course, structural information is much less commonly available for all the proteins in a set that need to be aligned. There is a new, emerging class of "hybrid" aligners which attempt to utilize both sequence and structural information.

1.1 Protein Structural Alignment Programs

Because it has been shown that structural aligners can produce better alignments than pure sequence aligners can, and because we are presenting an improved protein structural aligner in this paper, we will now describe structural alignment programs and one of their shortcomings in greater detail. For a succinct description of some of the approaches employed by sequence alignment programs, see the briefing by S. Batzoglou [1].

As mentioned previously, structural alignment programs assume an input of proteins' 3D structures in space along with these proteins' sequences. Thus most structural alignment programs will produce both a rigid body transformation that aligns the structures in space, and also a sequence alignment derived from that structural alignment that proposes homologous residue-residue correspondences. Hasegawa and Holms recently surveyed the current best structural alignment programs [6].

The quality of protein structural alignments is typically assessed using a purely geometric measure that is some function of the number of residues placed in alignment and an average RMSD (root mean square deviation) score for aligned residues, and can sometimes include a penalty for gaps. Since the best structural alignment programs use purely geometric measures to score alignments, it comes as no surprise that many structural aligners in use today begin by throwing out all sequence information and working only with the geometric location of the C_α atoms of the protein backbones. Because they ignore

$$\text{I.}\quad \begin{matrix} a_1 & a_2 & a_3 & — \\ b_1 & b_2 & b_3 & — \\ — & c_1 & c_2 & c_3 \end{matrix} \qquad \text{II.}\quad \begin{matrix} a_1 & a_2 & — & a_3 \\ b_1 & b_2 & b_3 & — \\ — & c_1 & c_2 & c_3 \end{matrix} \qquad \text{III.}\quad \begin{matrix} a_1 & a_2 & — & a_3 \\ b_1 & b_2 & — & b_3 \\ — & c_1 & c_2 & c_3 \end{matrix}$$

Figure 1: Example combinations of spatially consistent structural alignments between three proteins $a_1a_2a_3$, $b_1b_2b_3$, and $c_1c_2c_3$ where the following groups of amino acids have been found to be close in 3D space: (a_1, b_1), (a_2, b_2, c_1), (a_3, b_3, c_2), and (a_3, b_3, c_3). All three alignments I., II., and III. satisfy the geometric constraints, but since a structural superposition does not uniquely define a single sequence alignment, we would want to choose, in this case, the alignment that places the most similar amino acids in the same columns [27].

sequence information, structural aligners can sometimes produce suboptimal results in closely aligned regions, where simple frame shifts have little effect on the overall RMSD of an alignment, but do have a significant positive or negative effect on the sequence alignment of the region. A theoretical illustration of this "blindness" of structural aligners can be found in Figure 1, adapted from [27].

A logical fix to this problem would be to simply find a way to use the sequence information that structural alignment programs already have available to them to augment their alignments. However, although intuitively it seems that this extra sequence information should be used rather than discarded, the correct way to incorporate sequence information into structural alignment algorithms in order to improve their performance has remained elusive, since an optimization disagreement between minimizing RMSD and maximizing a sum-of-pairs score cannot be avoided. Finding an acceptable trade-off between minimizing the RMSD and improving a sequence score is often problem-dependent and difficult to generalize. Whereas a purely or heavily structure-based method might be used to identify regions of a protein that are highly conserved and critical for function, a sequence-based method might be more appropriate for phylogenetic reconstruction.

When a sequence alignment program and a structural alignment program produce different results on the same input set of proteins, it is unclear how to determine which alignment is actually correct, and by what reasonable measure. If the correct alignment is pre-defined solely based on the geometric location of the C_α atoms of the protein backbones, then this correct alignment can always be computed without ever looking at the protein sequences. At the opposite end of the spectrum, we could imagine a "true" correct alignment to be one that aligns residues that have evolved from residues in a common ancestor protein. Although we have no way to truly observe the course of evolution without time travel, such an alignment would presumably maximize a sum-of-pairs alignment score and might result in aligned regions with very little geometric similarity. Should these regions still be considered alignable?

Therefore our problem is two-fold; first, how, where, and in what circumstances should we incorporate sequence information to repair frame-offset errors in structural alignments, and second, how can we objectively assess the quality of such a "hybrid" alignment that

incorporates both sequence and structural protein information, particularly when hand-curated gold-standard benchmark reference alignments are unavailable to be compared against?

1.2 Objectively Assessing Alignment Quality

Methods for evaluating the quality of multiple protein alignments is important for the development of new alignment methods, and thus has received quite a bit of attention. Unfortunately, most existing algorithms for checking the validity and quality of multiple protein alignments are applicable only for sequence alignments, and vary in terms of goals and approaches. These algorithms look at (1) individual columns of an alignment, (2) the entire alignment as a whole, or (3) subsections of alignments when assessing quality. Some methods, such as norMD [29] and al2co [21] evaluate an alignment's quality column-by-column. Other quality assessment algorithms will look only at the entire alignment. The MUMSA program [13], for example, searches for regions in a multiple alignment which are identically aligned in many alignments produced by different aligners, or by the same aligner with different parameters. Using the assumption that these identically-aligned regions are more reliable than regions that are differently aligned by different alignment programs, MUMSA claims to assess the biological correctness of individual alignments by evaluating how well they match other alignments in these identically-aligned regions. Another method that evaluates the entire alignment is the Heads or Tails (HoT) method introduced by G. Landan and D. Graur [12]. This method reverses each of the sequences being aligned, realigns them using a specified aligner or parameters to an aligner, and then compares the original alignment to this new alignment of reversed sequences, checking the proportion of identical alignment columns and identically-paired residues, with the assumption that a strong method will maximize these two scores and thus have produced a reliable alignment. Another method, called WOOF, word-oriented objective function [2], evaluates alignments by identifying and scoring conserved amino acid patterns between pairs of sequences.

Despite the large body of research dedicated to assessing the quality of multiple sequence alignments objectively, many of these methods cannot be extended to effectively assess the quality of sequence-structure hybrid alignments. Several previous researchers have developed "hybrid" algorithms, including 3DCoffee [20], Promals3D [23], and Salign [14], that consider both sequence and structure when constructing protein alignments. These algorithms have all, to some extent, had to address the question of what their hybrid algorithm considers a "correct" alignment. An approach that seemed most intuitive and applicable to our problem is the Staccato scoring method, presented by Shatsky et al. [26], and described in further detail in the Methods section. However, with the notable exception of Salign, most of these papers describing hybrid aligners are actually trying to use *structural* information to improve *sequence* alignments, whereas the goal of this paper is to use *sequence* information to improve *structural* alignments. Even though the "correct" alignment in both scenarios is presumably the same, these are two very different

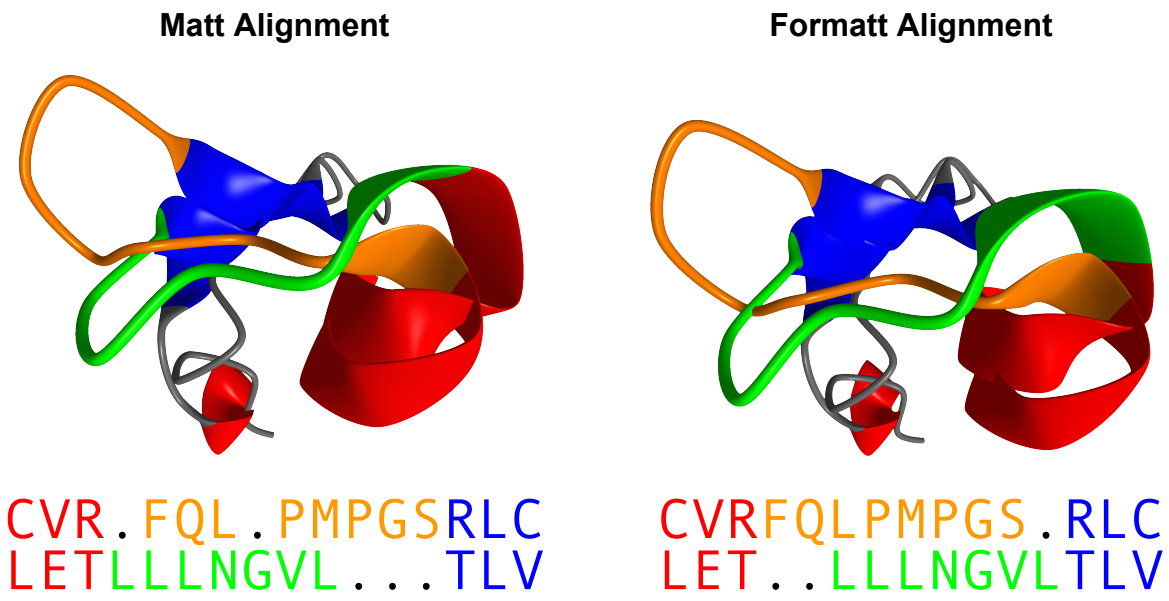

Figure 2: Example of Formatt's frame-offset repair on a subset (residues 37-50 of chain A of PDB ID 1c9f, and residues 64-76 of chain A of PDB ID 1d4b) of the Homstrad "CIDE-N" group. In both the structural (top) and sequence (bottom) alignments, the difference between Matt and Formatt are shown in orange and green; and the red and blue regions are α and β structures aligned identically by Matt and Formatt. Note that the Formatt alignment has fewer non-core residues (3) than Matt (5), and more core columns with sequence identity (3/12) than Matt (1/11).

problems, because whereas structural alignment programs can assume an input of sequence *and* structural information for proteins, sequence alignment programs cannot rely on structural information being available for all proteins.

1.3 Correcting Structural Aligners' Sequence Errors

Instead of asking if (partial) structural information can help sequence alignment algorithms, this thesis instead focuses on what is likely a substantially easier computational problem; does looking at sequence information help structural alignment algorithms, where 3D structural information *is* available for all the proteins in the set? The reason we believed this would help is that, anecdotally, for even the best structural alignment programs, it seemed that the resulting alignments for certain protein sets could be hand-"corrected" to produce an alignment that made more sense from a sequence point of view, with little or no loss in geometric fidelity. In fact, as reported by Landan et al. [12], in the rare case that a researcher actually inspects a multiple protein alignment, very often that alignment is "corrected by visual or manual inspection." An example of such an error that might be fixed manually can be found in Figure 2. In this case, two proteins were aligned by our group's own structure alignment program, Matt [15], and then corrected by the sequence alignment

step implemented in Formatt. You can see from the figure that the structural alignments produced are nearly identical, but the sequence alignment in Formatt's version has fewer gaps, and thus fewer non-core residues (3) than the sequence alignment in Matt's version (5). Additionally, whereas the depicted portion of the Matt alignment has only 1 core column where the residues are identical, the same portion of the Formatt alignment has 3. All other purely structural alignment algorithms that we tested will sometimes produce similar such errors.

To correct for exactly these "offset in sequence" register errors that purely structural aligners are prone to, we modified Matt to "peek" at the sequence at appropriate times in the algorithm. We present this modified version of Matt, called "Formatt" for "Frame Offset Repair for Matt," which initially follows the Matt algorithm exactly. Formatt uses the same aligned fragment-pair chaining method as Matt, first adding small, tightly-aligned blocks of 5 to 9 residues to an alignment, but differs from Matt in the following step of the algorithm, the "final extension phase," where these blocks are extended to complete the alignment. Whereas Matt uses solely geometric criteria to greedily align the protein backbones in the regions between blocks, which we will now refer to as "inter-block regions," Formatt considers both geometric and sequence similarity criteria in choosing which residues to align. Details of both the Matt and Formatt algorithms can be found in the Methods section.

It is important to note that the original Matt structural aligner that was modified to produce Formatt is specifically optimized for more distant homology [3]. Therefore, as was found in the original paper and again in this thesis, other aligners may perform better on highly homologous sequences. However, the hope is that Formatt's frame-offset correction will improve Matt's performance on closely homologous sequences while preserving Matt's performance advantage on remote homologs. We show in our Results section that this is indeed the case.

We test the performance of Formatt against the original Matt [15], against Mustang [10], another well-known multiple structure alignment program, and against Salign [14], which like Formatt incorporates sequence information into a structural alignment. We also considered 3DCoffee [19] and Promals3D [23] but found they were not competitive even on the closely-aligned structures in the popular Homstrad benchmark. Of course, as remarked above, to be fair to 3DCoffee and Promals3D, they can also produce alignments (which Formatt cannot) in the situation where structural information is only available for a subset of the protein sequences to be aligned, and were not optimized for the full-information structural alignment problem.

We tested the performance of Formatt against the performance of competitor aligners on the Homstrad benchmark set [17] and the SABmark Twilight Zone benchmark set [30], also referred to as just the "Twilight" set in this thesis. The Homstrad benchmark set contains several sets of closely homologous proteins, and, unlike the SABmark benchmark sets, also contains the "correct," gold-standard reference alignment for each of these sets, which were curated by hand. We chose the SABmark Twilight set to capture the alignment of more distantly related proteins. Since there are no available "correct" alignments of these

protein sets to compare against, we use the objective Staccato *Seq* and *Str* scores as introduced by Shatsky, Nussinov and Wolfson [27] to measure alignment quality. Although Mustang and Salign produce reasonable, and in many cases more correct, alignments on Homstrad than either Matt or Formatt, neither Mustang nor Salign produce alignments on SABmark Twilight with a reasonable RMSD, whereas both Formatt and Matt *do* produce alignments with tight RMSDs on these sets.

We make available Formatt source code, freely available for download under the Gnu Public License, at http://bcb.cs.tufts.edu/formatt, where we also make available Homstrad and SABmark benchmark reference alignments aligned by Formatt.

2 Methods

2.1 Matt

The Matt structural aligner [15] is an aligned fragment-pair chaining method. Like other structural aligners in this class, Matt first finds blocks of between 5 and 9 residues in each chain that share very close spatial alignment. Then, Matt greedily adds these blocks into a final multiple structural alignment, and extends these aligned blocks, keeping close spatial alignment, in a "final extension phase" by adding adjacent residues into the inter-block regions. A depiction of a Matt "bent" alignment, that is, before these closely-aligned 5-9 residue blocks have been extended into the inter-block regions, can be found in Figure 3.

What separates Matt from other fragment-pair chaining methods is that Matt initially adds tightly-aligned blocks to the final alignment with disregard to geometric consistency, allowing, in essence, impossible bends, translations, and twists between aligned blocks prior to the final extension phase. Thus the name, Matt, which stands for Multiple Alignment with Translations and Twists. This way, Matt is able to detect regions in close spatial contact and incorporate them into an alignment in situations where other aligners would have immediately, and oftentimes erroneously, disallowed such blocks from entering an alignment.

Matt places residues from the input protein sequences into alignment based on the resulting RMSD. That is, residues are aligned if the distance between them, spatially, is minimized. Of course, it is easy to see how placing residues in alignment based solely on RMSD could be problematic; any residue in a column by itself would have an RMSD of 0. On the other hand, maximally long alignments in which all residues from all chains participate in the overall alignment could be achieved without regard to RMSD. We define the "core" of a multiple protein structural alignment to be the subset of columns in the alignment with no gaps present for any of the chains. Therefore, Matt finds the optimal trade-off between minimizing the average core RMSD and maximizing the number of residues in alignment. This balance was achieved by finding a linear combination of RMSD and core

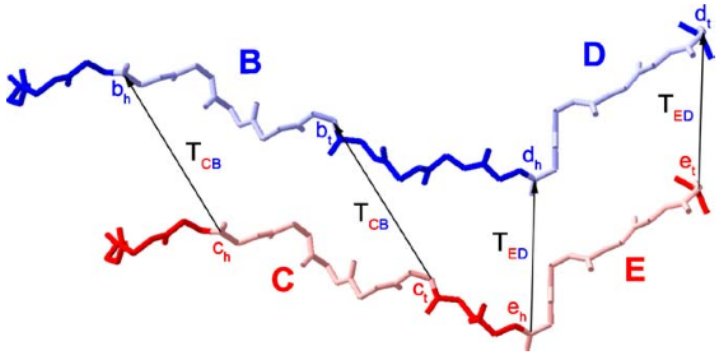

Figure 3: Segment of two chains participating in an alignment after the Matt "bent" phase. The solid blue and red portions of the two chains are found in be in very close spatial alignment, and were added into the final alignment. These blocks are extended to incorporate portions of the translucent sections labeled B, C, D, and E in the "final extension phase" according to purely geometric measures in the original Matt algorithm, and are extended according to geometric and sequence measures in the modified Format algorithm. This figure comes from [15].

length that optimally separated SABmark [30] positive from decoy chains at the superfamily level of homology.

In Matt's final extension phase, the algorithm loops through all inter-block regions, first adding new blocks into these regions of length 4, then 3, then 2, then 1. This entire process is continued until no new changes to the alignment are made, which in practice is between 1 and 3 loops. The value of 4 as the longest length of a block to be added was arbitrarily chosen, again in an effort to achieve longer sections of core residues. Each "block" that is added, either before the final extension phase or during it, must include a residue from each chain in the alignment. In other words, partial alignments, or columns with gaps, are not allowed to participate in an alignment. Prior to outputting an alignment, Matt aesthetically fixes any remaining inter-block regions with partial alignments, but this does not affect the rigid body transformation that aligns the structures in space, but rather just the sequence alignment derived from it.

2.2 Improving upon Matt

The chief limitation of the Matt procedure is that the inter-block regions in between the original, closely-aligned, 5-9 amino acid blocks are still aligned purely according to this balance between core length and RMSD, and thus the final alignment may choose arbitrarily between different possible alignments of similar RMSD values, as previously shown

in Figure 1. This can lead to certain alignments with otherwise obvious sequence similarity being discarded due to negligible increases in RMSD.

By preserving sequence information instead of immediately discarding it at the start of the algorithm, and allowing the input from a pure sequence alignment tool to influence the final alignment, we aim to improve the alignments of these inter-block regions. We decided that this extension phase would be the ideal point in the algorithm to allow sequence information to influence an alignment, because logically only structural information should play a role in detecting the original closely-aligned blocks in the bent alignment. Incorporating sequence information prior to the final extension phase could damage this original alignment if the proteins being aligned have very low sequence similarity. On the other hand, choosing to incorporate sequence information at later steps in the algorithm would have a negligible effect on the overall alignment, and frame-offset errors that were introduced in the final extension phase would not be fixed.

Therefore, Formatt improves upon the Matt algorithm by first producing an initial bent alignment, identical to Matt's, of 5 to 9 amino acid blocks, and then extending these blocks using sequence information as follows.

2.2.1 Approach 1: Align entire inter-block regions using sequence

The very first version of Formatt performed the following steps to generate an alignment:

1. A "window length" is calculated for each inter-block region, equivalent to the minimum length of the chains participating in the inter-block region. For example, in an alignment with three chains, called A, B, and C, residues $A_{13}...A_{20}$, $B_8...B_{12}$, and $C_{11}...C_{19}$ could participate in one inter-block region. In this inter-block region, the maximum length is 9, occurring in chain C, the minimum length is 5, occurring in chain B, and the window length would also be 5, equivalent to the minimum length.

2. The RMSD of all possible alignments of this particular window length in each inter-block region is measured. Specifically, we implement a sliding window that checks all possible orientations of window-length subchains in each chain in the inter-block region against each other.

3. If any window orientation results in a region with average RMSD < 5 Angstroms (Å), the entire inter-block region is passed to a sequence aligner. We output the amino acid sequences from this region to a temporary file and run an existing sequence aligner to produce an alignment. We currently use Muscle [5] but in principle, any multiple sequence alignment application could be used.

4. The resulting sequence alignment for the inter-block region is used in place of the structural alignment extension that Matt originally relied upon.

5. Otherwise, we leave this inter-block region untouched. It is eventually aligned using the purely geometric extension method employed by original Matt.

This approach was based on the following two assumptions about the structure of inter-block regions, which turned out to be problematic.

- Most inter-block regions contain subchains with lengths of about 5 to 15 residues each. This assumption was supported by a histogram of inter-block region lengths.
  - Although the vast majority of inter-block regions were fairly short, it turns out that only about half of the Homstrad protein sets we were testing on had several, short inter-block regions upon entering the final extension phase, and the other portion of sets entered this phase with only a few, very long inter-block regions. When we instead looked at the average inter-block region length for each Homstrad protein set, we discovered that the average inter-block region length was much larger than we originally assumed.
- The lengths of the subchains participating in each inter-block region were nearly equivalent in length, differing only by 1 or 2 residues.
  - There actually appeared to be a few inter-block regions in particular protein sets where a single subchain was very short or very long compared to the lengths of the other subchains. We then passed such entire inter-block regions to an external sequence aligner, disregarding which window orientations resulted in a tight RMSD. As a result, shorter subchains were placed at positions that maximized a sequence score, and in several cases yielded high and undesirable RMSD values as a result.

Because of the flawed assumptions about the structure of inter-block regions, this approach turned out to be inadequate. The problem of having long inter-block regions was a major one, and replacing entire inter-block structural alignments with sequence alignments in the event that one small window had a good RMSD was clearly not the ideal approach.

2.2.2 Approach 2: Align segments of inter-block regions using sequence

We then modified Formatt's original approach to deal with larger inter-block regions. The new version of the algorithm was very similar to the previous version, except steps 3 and 4 were updated. Instead of passing the entire inter-block region to a sequence aligner in the event that a certain window orientation had an RMSD < 5 Å, Formatt instead passed only that particular window with the tightest RMSD to an external sequence aligner. The resulting sequence alignment was then incorporated back into the final alignment, and the resulting new inter-block regions were again considered for potential sequence alignment.

This alleviated the problem we were seeing before, where seemingly "junk" alignments were produced when a sequence aligner was used to align a large region, particularly when that region had only a short segment with a low RMSD and otherwise little to no sequence identity.

However, we encountered a new problem. Because Formatt was now choosing which subsection of an inter-block region to pass to an external sequence aligner, there arose a new opportunity for frame offset issues to occur. The new issue that started to occur was that in certain inter-block regions, the "winning" window frame with the tightest RMSD did not contain the best possible sequence alignment, and if the frame had shifted one or two positions to the right or left for any of the participating subchains, a significantly better sequence alignment was possible. Thus, we recreated our original frame-offset issue in a smaller form.

Additionally, choosing the ideal window size for this purpose was also complicated. The range of lengths of subchains in each inter-block region, the RMSD over the entire region *and* the RMSD for each sliding window frame, and also the maximum length of the participating subchains all seemed to play an important, yet inconsistent, role in choosing an ideal window size.

2.2.3 Approach 3: Only align inter-block regions of certain lengths using sequence

The third approach produced much better results, which are the results presented in this paper. There is one more approach detailed in the following subsection which is currently being implemented, but for which no results have yet been calculated. Current difficulties with this approach are discussed in the next section and in the Future Work section.

Because at this point it seemed that only "large" inter-block regions were problematic and difficult to deal with, we modified Formatt to instead follow a different set of steps:

1. Given an inter-block region where the length of the longest subchain is less than some previously specified window size, we measure the optimal RMSD of the region using the same sliding window technique described in the first and second approaches.
2. If the RMSD of this region is less than Matt's original spatial alignment threshold (5 Å), we output the amino acid sequences from this region to a temporary file, and run an existing external sequence aligner on those sequences. (As previously mentioned, we currently use Muscle.)
3. We then incorporate this sequence alignment into the overall structural alignment in place of whatever structural alignment could have been calculated by original Matt.
4. Otherwise, in the event that the RMSD of this inter-block region is greater than the cutoff of 5 Å, then this region is not passed to a multiple sequence aligner, and in-

stead we use the same purely structural extension method as would have been done originally.

5. Inter-block regions of longer than our previously specified window size are greedily aligned based solely on RMSD; they are, in practice, unlikely candidates for a good sequence alignment.

The choice of window size to pass to the sequence aligner in Step 1 above determines when sequence rather than RMSD-based structure alignment is used to align inter-block regions. As noted in Step 5, inter-block regions longer than the window size are greedily aligned based purely on RMSD. We present results for window size choices of 5, 10, 15, and 20 residues.

2.2.4 Approach 4: *Choose* between a sequence and structural alignment for each inter-block region

As you will see in the Results section, there were a select few cases where our new approach seemed to be doing *worse* than original Matt. To deal with this problem, we plan to implement a method that calculates a sequence alignment *and* whatever original structural alignment Matt may have produced for *every* inter-block region (i.e. no longer only for inter-block regions of maximum length less than the window sizes of 5, 10, 15, or 20), compare them using an objective measure that takes both sequence and structural similarity into account, and select whichever alignment scores higher to incorporate into the final structural alignment.

Calculation of the Objective Staccato *Cons* Score

The objective score that will be implemented in the newest version of Formatt is the Staccato *Cons* score, as described by Shatsky et al. [27], where *Cons*(c) $\in$ [0,9]. Formatt calculates the *Cons*(c) score for each column in an alignment, where c is a single column, and averages the score of all columns in the inter-block region to get an overall score for that region. As mentioned before, alignments produced both by the purely geometric method and by an external sequence aligner for each inter-block region are compared using the objective *Cons* score, and the "winning" alignment is selected. The calculation for this score is reproduced below:

$$Cons(c) = w * Cons_{seq}(c) + (1 - w) * Cons_{str}(c)$$

The value of w is set to 0.5 for our testing. This value is a parameter to Formatt and can be adjusted so that either sequence or structure is weighted more heavily if a user has some prior knowledge about the input data. If the user knows that the proteins being aligned are more closely related, the value of w should be increased from 0.5, and if they

are more distantly related, the value of w should decrease, so that structure is weighted more heavily. The calculation for the $Cons_{str}(c)$ portion of the overall *Cons*(c) score is shown below:

$$Cons_{str}(c) = \begin{cases} 9 & \textit{if (rmsd(c) > 22.62Å)}, \\ \dfrac{1}{f + \dfrac{1-f}{rmsd(c)}} & \textit{otherwise}. \end{cases}$$

In the above $Cons_{str}(c)$ equation, the value of f is operationally set to 0.07. We calculate the $Cons_{seq}(c)$ score using the Blosum62 scoring matrix, so the values of each column can range between -4 to 5.75. We map these values to the range [0,9] using the following equation.

$$Cons_{seq}(c) = 9 * \left(1 - \frac{Cons^*_{seq}(c) + 4}{9.75}\right)$$

The actual $Cons^*_{seq}(c)$ score calculation is described by the following equations, where N is the number of chains in the alignment, and c_i is the amino acid residue from chain i in column c.

$$Cons^*_{seq}(c) = \sum_{i=1}^{N} \sum_{j>i}^{N} w_i w_j Score^*_{SM}(c_i, c_j)/W$$

$$Score^*_{SM}(a,b) = \begin{cases} Score_{SM}(a,b) & \textit{if (a} \neq \textit{b)}, \\ \displaystyle\sum_{i=1}^{20} Score_{SM}(i,i)/20 & \textit{if (a = b)}. \end{cases}$$

The $Score^*_{SM}(a,b)$ is merely a modified scoring matrix such that only the matrix diagonal is changed, where the original scoring matrix is represented by $Score_{SM}(a,b)$). In the original Staccato and our implementation, the Blosum62 scoring matrix is used. If we only used an unaltered sum-of-pairs scoring method, sequences with high similarity would have a significantly higher $Score^*_{SM}$, so to overcome this overweighting, the following weights w_i and normalized weight W are used to balance the weighting scheme:

$$w_i = \sum_{j \neq i}^{N} \frac{1 - \frac{PercentIdentity(S_i, S_j)}{100}}{N-1}$$

$$W = \sum_{i=1}^{N} \sum_{j>i}^{N} w_i w_j$$

The overall *Cons* score will range between the values 0 and 9, and we want to maximize this score in order to select the "best" alignment. We conjecture that this approach must improve Formatt's performance, because the event where considering sequence similarity is detrimental to an alignment will be eliminated.

2.3 Validation

In order to quantitatively assess Formatt's performance, we evaluate it against two well-known benchmark sets, Homstrad [17] and SABmark [30].

Homstrad Benchmark Set

The Homstrad multiple-alignment benchmark is a manually curated set of 1,028 alignments, each of which contains between 2 and 41 structures. Homstrad alignments consist of closely related, homologous protein families, where proteins in each set have at least 30% sequence identity on average. The original alignments were derived using COMPARER, then annotated using JOY in a format representing the local structural environment of each amino acid residue. The alignments were tweaked manually where necessary. For Homstrad alignments, we can assume the manually curated alignments form a gold-standard set of "correct" alignments.

We test on the 414 Homstrad alignments with more than 2 (i.e. between 3 and 41) structures in the alignment, because these necessitate a multiple rather than a pairwise structure alignment program. Of these 414 sets, 16 sets require fused or reversed chains to be aligned, and unfortunately there is no automatic way to input these modified chains without actually manually editing the appropriate PDB files. Therefore Matt and Formatt, which do not yet have the capability of automatically editing the input PDB files accordingly, were only run on 398 sets. Other structural aligners have the same problem with aligning fused or reversed chains.

SABmark Benchmark Set

The SABmark benchmark, unlike the Homstrad benchmark, does not contain reference alignments, and contains only sets of remotely homologous proteins to be aligned. The SABmark benchmark covers the entire fold space according to the SCOP classification, and is divided into two datasets: the superfamily and Twilight Zone datasets, which each contain subsets of 3 to 25 remotely homologous protein structures. Each Twilight Zone set represents a SCOP fold, and the proteins in each set have very low sequence identity (0-25%).

We test Formatt and its competitors on the 209 subsets in the Twilight set. As previously mentioned, we do not have a gold-standard set of "correct" alignments as we did with the Homstrad sets, and must instead determine alignment quality solely by objective means, such as core length, average core pairwise RMSD, as well as the Staccato scores, as introduced by [27].

Staccato Objective Scores

We review how the Staccato scores are computed next. Note that our calculation of Staccato scores diverge from the algorithms presented in the original paper in one important respect: we only consider core positions, or columns without any gaps, in the alignment when scoring a multiple alignment.

The Staccato *Seq* score measures sequence alignment quality and is a normalized sum-of-pairs score based upon the BLOSUM62 matrix. More formally:

1. Let $seq = 0$
2. Let P be a set of protein chains p_1 through p_n participating in a multiple alignment A
3. For every column $c_j \in A$
   (a) Let $r_{i,j}$ be the residue at position j of chain p_i
   (b) If $\forall i \in \{1..n\}$, $r_{i,j}$ is not a gap
      i. Let B be the BLOSUM62 score of $r_{i,j} \forall i \in \{1..n\}$
      ii. $seq = seq + B$
4. return seq

The Staccato *Str* score is a measure of what percentage of core residues in an alignment have an RMSD of < 3 Å, with core residues defined as those columns in the multiple alignment for which every chain has a residue, or equivalently, those columns without gaps. More formally,

1. Let $str = 0$
2. Let P be a set of protein chains p_1 through p_n participating in a multiple alignment A
3. For every column $c_j \in A$
   (a) Let $r_{i,j}$ be the residue at position j of chain p_i
   (b) If $\forall i \in \{1..n\}$, $r_{i,j}$ is not a gap
      i. Let R be the RMSD of $r_{i,j} \forall i \in \{1..n\}$
      ii. If $R < 3.0$ Å
         A. $str = str + 1$
4. return str

3 Results

Table 1: Performance on Homstrad Benchmark Set (414 sets)

Aligner	Num. of Results	Average Core				
		Length	RMSD	% Correct	*Seq* score	*Str* score
Homstrad	414	126.8	2.71	(100%)	39.2	84.4
Mustang	410	152.8	3.60	79.3%	37.5	82.1
Matt	398	178.4	1.72	73.4%	32.0	86.8
Promals3D	383	185.8	—	18.6%	33.7	—
Salign	277	172.6	2.29	78.1%	37.4	78.9
Formatt (5)	398	178.6	1.79	73.5%	32.0	85.6
Formatt (10)	398	178.9	1.83	73.4%	32.3	84.7
Formatt (15)	398	179.4	1.93	73.4%	32.6	83.8
Formatt (20)	398	179.6	1.95	73.4%	32.7	83.5

As can be seen in Table 1, all the aligners do a reasonable job producing alignments for the 414 Homstrad multiple protein sets, although no aligner actually produces results for all 414 sets. It is also important to notice that Salign only produced results on 277 out of 414 sets, and therefore the resulting long average core length and high percent correct might change if alignments for all Homstrad sets were available. The Homstrad gold standard has the smallest average core length, followed by Mustang and Salign. Matt's average core length is longer, and each version of Formatt with progressively longer sequence alignment windows achieves a progressively longer core length. On the other hand, as the Formatt alignment window increases, RMSD increases slightly from Matt as well, as would be expected. Formatt's and Matt's percent correct, however, according to the gold standard, underperforms the other methods, though the difference between Formatt and Matt by this measure is negligible. Thus, we conclude that Formatt and Matt by the most objective measure (both core length and RMSD) outperform other methods on the Homstrad benchmark set, but underperform them according to the Homstrad hand-curated alignments. On

Table 2: Performance on SABmark Twilight Benchmark Set (209 sets)

Aligner	Num. of Results	Average Core			
		Length	RMSD	*Seq* score	*Str* score
Mustang	204	63.4	11.83	-9.9	49.6
Matt	209	67.1	2.64	-15.4	64.9
Promals3D	196	75.4	—	1.6	—
Salign	155	60.0	22.15	-15.9	45.4
Formatt (5)	209	67.0	2.64	-15.4	64.7
Formatt (10)	209	67.0	2.68	-15.3	64.1
Formatt (15)	209	67.0	2.71	-15.2	63.5
Formatt (20)	209	67.0	2.74	-15.1	63.2

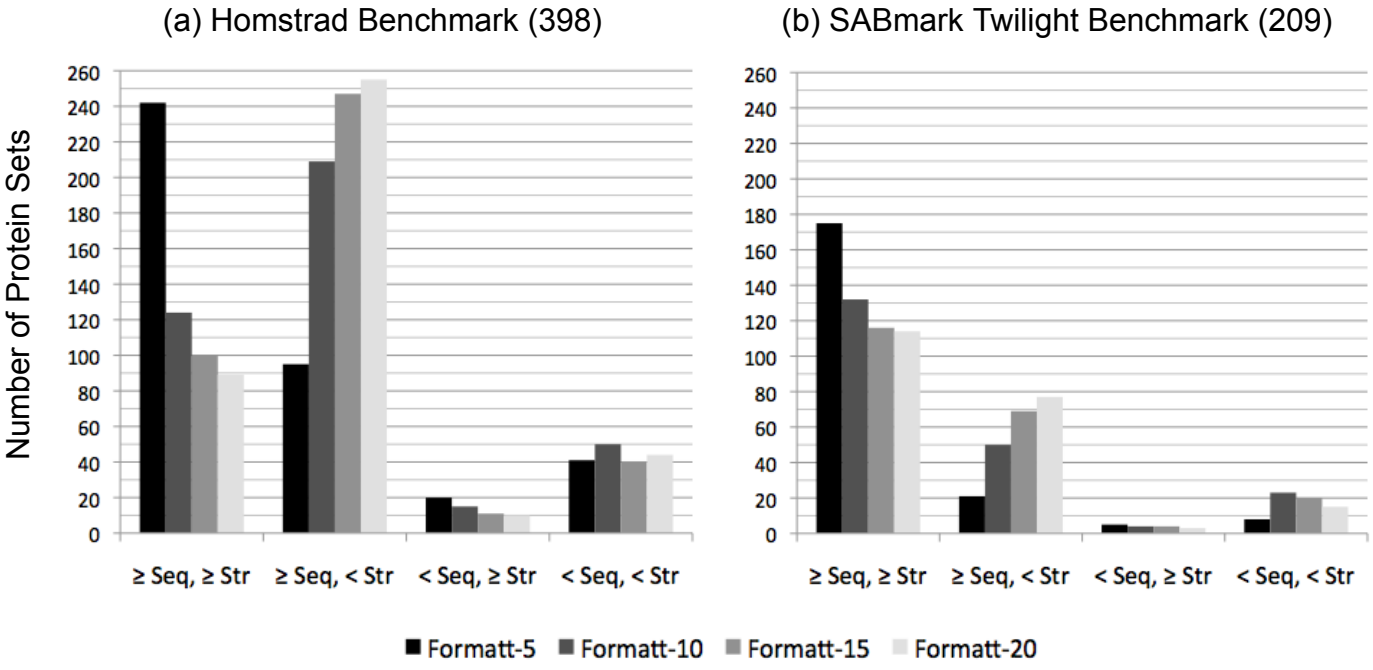

Figure 4: Histograms showing the effect of different Formatt window lengths on Formatt's performance compared to Matt's, on the Homstrad (a) and SABmark Twilight (b) multiple alignments. Numbers in parentheses indicate the total number of protein sets that were tested on for each benchmark. For the vast majority of both Homstrad and SABmark Twilight protein sets, Formatt with a window size of 5 outperforms Matt on the Staccato *Seq* score, while rarely performing worse on the *Str* score. As the window size increases, Formatt outperforms Matt more frequently on the *Seq* score but less frequently on the *Str* score.

the other hand, it is clear that in comparing Formatt to Matt, Formatt increases core length for a small penalty in RMSD.

We also tested Promals3D on the Homstrad benchmark set. Note that Promals3D outputs only a sequence alignment without coordinates, so an RMSD and *Str* score could not be calculated. However, when we compared the Promals3D to the Homstrad gold-standard alignments, the average percentage correct was only 18.6%. We tested a a subset of the Homstrad benchmark set against 3DCoffee and the results were even worse, and unfortunately 3DCoffee was also unable to run on a majority of the Homstrad sets. Although at first we assumed this could have been a result of neither Promals3D nor 3DCoffee being able to adequately handle malformed PDB files, the problems continued even when aligning just sequences. Thus, we conclude that Promals3D and 3DCoffee are not producing competitive alignments on this benchmark.

Table 2 shows that Matt and Formatt outperform Mustang and Salign both in terms of core length and RMSD on the SABmark Twilight benchmark set. Here, however, we do

not have gold-standard reference alignments. It is also less immediately clear from Table 2 whether Formatt or Matt alignments are to be preferred. To further study this question, we look at the Staccato scores suggested by [27] on both Homstrad and SABmark. Figure 4a shows that on Homstrad, Formatt with a window size of 5, for example, has a Staccato *Seq* score and *Str* score greater than or equal to Matt on 242 out of 398 groups, and a *Seq* score greater than or equal to Matt but a *Str* score less than Matt on 95 groups. Figure 4b shows that on the SABmark Twilight set, Formatt with a window size of 5 has a Staccato *Seq* score and *Str* score greater than or equal to Matt on 175 out of 209 groups, and a *Seq* score greater than or equal to Matt but a *Str* score less than Matt on 21 groups. Thus, by the objective Staccato measures, Formatt's alignments are superior to Matt's alignments. As Formatt window size increases, the Staccato *Seq* score goes up in a tradeoff against the Staccato *Str* score.

4 Discussion

We have introduced Formatt and shown that incorporating sequence information can improve the quality of structural alignments, both in terms of gold-standard alignment benchmarks, and in terms of objective measures of sequence and structural alignment quality (e.g. Staccato *Seq* and *Str* scores [27]). We were particularly interested in "correcting" Matt structural alignments to better capture correct sequence homology because of our extensive use of the Matt structural alignment program in the training phase as we build HMMs [11] and Markov Random Fields [16] from sets of solved protein structures that all fold into the same shape, to learn to recognize new protein sequences that match these models. More consistent alignments lead to better structural templates, and therefore better motif recognition programs. This is the same problem domain that motivated the work on the Salign program as well [14].

Formatt is a variant of the Matt [15] multiple structure alignment program, one of a new generation of structural alignment programs that incorporate flexibility into multiple protein structure alignments. Other recent pairwise and multiple structure alignment programs that also incorporate some form of flexibility into alignments include FlexProt [25], Fatcat [31], Posa [32], Rapido [18], and FlexSnap [24]. It would be interesting to see if some form of sequence alignment could be incorporated into these programs as well, and whether it could improve their structural alignments.

5 Future Work

As mentioned above, our current implementation of Formatt was tested using only two different popular sequence alignment methods, namely Clustal-W [7] and Muscle [5]. We only reported results for Muscle, since Formatt's resulting percent correct on Homstrad

was consistently higher over Clustal-W. However, there are many newer multiple sequence alignment programs that have recently been shown to perform well on more distantly homologous sequences, such as ProbCons [4], MUMMALS [22] and MAFFT [8]. We conjecture that substituting some of these programs in for Muscle for Formatt's multiple sequence alignments might improve Formatt results still further. So far, we have tested the replacement of Muscle in Formatt with Clustal-W, ProbCons and MAFFT, and results appear in Table 3.

Table 3: Sequence Aligner Effect on Formatt's Performance on Homstrad Benchmark

Window Length	**Aligner**	**Average Core**				
		Length	RMSD	% Correct	*Seq* score	*Str* score
5	Muscle	178.6	1.79	73.5%	32.0	85.6
	Clustal-W	178.0	1.78	73.0%	32.3	85.9
	MAFFT	177.7	1.78	73.0%	32.2	85.8
	ProbCons	174.7	1.76	71.8%	32.1	86.3
10	Muscle	178.9	1.83	73.4%	32.3	84.7
	Clustal-W	178.0	1.82	72.8%	32.9	85.9
	MAFFT	176.4	1.82	72.1%	32.9	85.0
	ProbCons	168.8	1.71	69.5%	32.8	87.4
15	Muscle	179.4	1.93	73.4%	32.6	83.8
	Clustal-W	178.5	1.91	72.6%	33.3	84.0
	MAFFT	176.4	1.92	71.7%	33.3	84.2
	ProbCons	165.2	1.66	68.3%	33.3	88.1
20	Muscle	179.6	1.95	73.4%	32.7	83.5
	Clustal-W	178.7	1.94	72.6%	33.3	83.8
	MAFFT	178.2	1.77	73.3%	32.2	86.2
	ProbCons	163.9	1.65	67.9%	33.4	88.4

As we already knew, Formatt using Muscle outperforms the version of Formatt using Clustal-W in regard to percent correct on Homstrad in nearly all cases. However, it appears that using MAFFT, ProbCons, and Clustal-W improve both the *Seq* and *Str* scores on Homstrad, yet the core length also reduces. It is therefore hard to say whether these particular aligners are improving Formatt. Because of this, we chose not to also run these variant versions of Formatt on SABmark before we can first assess the performance gains, if any, on Homstrad. However, this question leads us to the more pressing issue, the next major modification to Formatt as a future step.

As mentioned in the Methods section, we are attempting to modify Formatt so that for every inter-block region, regardless of the length of that inter-block region (which we currently take into consideration), we calculate both a sequence alignment and a structural alignment, and then choose the better of these alignments according to the Staccato *Cons* score. This way, we will select an alignment that most optimally maximizes the core length, minimizes the core RMSD, and maximizes both the Staccato *Seq* and *Str* scores. If we are consistently choosing the "best" alignment for each interblock region, it will be much

easier to see whether changing the sequence aligner actually improves the performance of Formatt. Furthermore, we suspect that implementing this new final extension phase will improve all alignments of protein sets for which Formatt is currently doing worse than Matt. In essence, Formatt will never perform worse than Matt.

There are a few minor setbacks to implementing the newest version of Formatt with the Staccato *Cons* score. Specifically, calculating what original Matt would have done for a given inter-block region is difficult to determine. In the original Matt code, the addition of somewhat tightly-aligned blocks of lengths 4, 3, 2, and 1 are added to the *entire alignment* in that order, and not to each inter-block region in that order. That is, all new blocks of length 4 are added to all existing inter-block regions in the alignment, where appropriate, before any blocks of length 3 are added. If we update this algorithm to look individually at inter-block regions and add blocks there instead, core length increases dramatically, but the RMSD also increases dramatically. It is unclear why we are seeing these results.

Furthermore, in several cases, a sequence alignment for one inter-block region can change the resulting structural alignments in subsequent inter-block regions. That is, if we calculate a structural alignment in inter-block regions i, j, and k, and then add some sort of sequence alignment into inter-block region i, the structural alignments, specifically the RMSDs of these structural alignments, that we previously calculated for j and k can also change. Any *Cons* score, which depends on the RMSD of an alignment, that was initially calculated would no longer be valid in subsequent steps of the algorithm as sequence alignments are incorporated. Therefore, a simple solution to our previously-mentioned problem, merely calculating all of the structural alignments for each inter-block region in one go and then evaluating these will not work.

We are currently attempting to duplicate the results of the original Matt final extension phase without using the same algorithm as before. This will hopefully make it much easier to calculate a reliable structural alignment for a given inter-block region that doesn't drastically change from what is expected with changes in other inter-block regions.

References

[1] S. Batzoglou. The many faces of sequence alignment. *Briefings in Bioinformatics*, 1:6–22, 2005.

[2] Robert G. Beiko, Cheong Xin Chan, and Mark A. Ragan. A word-oriented approach to alignment validation. *Bioinformatics*, 21(10):2230–2239, 2005.

[3] N Daniels, A Kumar, L Cowen, and M Menke. Touring protein space with Matt. *Bioinformatics Research and Applications*, 6053/2010:18–28, Jan 2010.

[4] C.B. Do, M.S. Mahabhashyam, M. Brudno, and S. Batzoglou. Probcons: probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15:220–240, 2005.

[5] R.C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucl. Acids Res.*, 32:1792–1797, 2004.

[6] H. Hasegawa and L. Holm. Advances and pitfalls of protein structural alignment. *Current Opinion in Structural Biology*, 19(3):341 – 348, 2009.

[7] D. Higgins, J. Thompson, T. Gibson, J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.

[8] K. Katoh and H. Toh. Recent developments in the MAFFT multiple sequence alignment program. *Briefings in Bioinformatics*, 9:286–298, 2008.

[9] C. Kim and B. Lee. Accuracy of structure-based sequence alignment of automatic methods. *BMC Bioinformatics*, 8:355, 2007.

[10] A.S. Konagurthu, J.C. Whisstock, P.J. Stuckey, and A.M. Lesk. MUSTANG: A multiple structural alignment algorithm. *Proteins: Struct. Func. Bioinform.*, 64:559–574, 2006.

[11] A. Kumar and L. Cowen. Recognition of beta structural motifs using hidden Markov models trained with simulated evolution. *Bioinformatics*, 26:i287–i293, 2010.

[12] Giddy Landan and Dan Graur. Heads or tails: a simple reliability check for multiple sequence alignments. *Mol. Biol. Evol.*, 24(6):1380–1383, 2007.

[13] Timo Lassmann and Erik L. L. Sonnhammer. Automatic assessment of alignment quality. *Nucleic Acids Res.*, 33(22):7120–7128, 2005.

[14] M.S. Madhusudhan, B. M. Webb, M. A. Marti-Renom, Narayanan Eswar, and Andrej Sali. Alignment of multiple protein structures based on sequence and structure features. *Protein Engineering, Design and Selection*, pages 1–6, 2009.

[15] M. Menke, B. Berger, and L. Cowen. Matt: Local flexibility aids protein multiple structure alignment. *PLoS Computational Biology*, 4(1):e10, 2008.

[16] M. Menke, B. Berger, and L. Cowen. Markov random fields reveal an N-terminal double propeller motif as part of a bacterial hybrid two-component sensor system. *PNAS*, 107:4069–4074, 2010.

[17] K. Mizuguchi, C.M. Deane, T. L. Blundell, and J.P. Overington. HOMSTRAD: a database of protein structure alignments for homologous families. *Protein Sci.*, 11:2469–2471, 1998.

[18] R. Mosca, B. Brannetti, and T. R. Schneider. Alignment of protein structures in the presence of domain motions. *BMC Bioinformatics*, 9:352, 2008.

[19] C. Notredame, D.G. Higgins, and J. Heringa. T-coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205 – 217, 2000.

[20] O. O'Sullivan, K. Suhre, C. Abergel, D.G. Higgins, and C. Notredame. 3DCoffee: combining protein sequences and structures with multiple sequence alignments. *J. Mol. Biol*, 340:385–395, 2004.

[21] J. Pei and N. V. Grishin. Al2co: calculation of positional conservation in a protein sequence alignment. *Bioinformatics*, 17:700–712, 2001.

[22] J. Pei and N.V. Grishin. MUMMALS: multiple sequence alignment improved by using hidden Markov models with local structure information. *Nucleic Acids Res.*, 34:4364–4374, 2006.

[23] J. Pei, B. H. Kim, and N. V. Grishin. PROMALS3D: a tool for multiple protein sequence and structure alignments. *Nucleic Acids Res.*, 36:2295–2300, 2008.

[24] S. Salem, M. J. Zaki, and C. Bystroff. FlexSnap: Flexible non-sequential protein structure alignment. *Algorithms in Molecular Biology*, 12(5), 2010.

[25] M. Shatsky, R. Nussinov, and H.J. Wolfson. Flexible protein alignment and hinge detection. *Proteins*, pages 242–256, 2002.

[26] M. Shatsky, R. Nussinov, and H.J. Wolfson. A method for simultaneous alignment of multiple protein structures. *Proteins: Struct. Func. and Bioinform.*, pages 143–156, 2004.

[27] M. Shatsky, R. Nussinov, and H.J. Wolfson. Optimization of multiple-sequence alignment based on multiple-structure alignment. *Proteins: Structure, Function and Bioinformatics*, 62:209–217, 2006.

[28] E. Sonnhamer, S. Eddy, E. Birney, A. Bateman, and R. Durbin. Pfam: multiple sequence alignments and HMM-profiles of protein domains. *Nucleic Acids Res.*, 26(1):320–322, 1998.

[29] J. D. Thompson, F. Plewniak, R. Ripp, J.C. Thierry, and O. Poch. Towards a reliable objective function for multiple sequence alignments. *J. Mol. Biol.*, 21:937–951, 2001.

[30] I. VanWalle, I. Lasters, and L. Wyns. SABmark–a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics*, 21:1267–1268, 2005.

[31] Y. Ye and A. Godzik. Flexible structure alignment by chaining aligned fragment pairs allowing twists. *Bioinformatics*, Suppl 2:II246–II255, 2003.

[32] Y. Ye and A. Godzik. Multiple flexible structure alignment using partial order graphs. *Bioinformatics*, 21:2362–2369, 2005.