MUSTANG: A Multiple Structural Alignment Algorithm

Arun S. Konagurthu,[1,2] James C. Whisstock,[2,3] Peter J. Stuckey,[1,4,*] and Arthur M. Lesk[2,5,*†]
[1]*Department of Computer Science and Software Engineering, The University of Melbourne, Parkville, Melbourne, Victoria, 3010 Australia*
[2]*Victorian Bioinformatics Consortium, Department of Biochemistry and Molecular Biology, Monash University, Victoria, 3800 Australia*
[3]*ARC Centre for Structural and Functional Microbial Genomics, Monash University, Victoria 3800, Australia*
[4]*National ICT Australia (NICTA) Victoria Laboratory at The University of Melbourne, Australia*
[5]*Department of Biochemistry and Molecular Biology, and the Huck Institutes of the Life Sciences, Genomics, Proteomics, and Bioinformatics Institute, The Pennsylvania State University, University Park, Pennsylvania 16802, USA*

ABSTRACT **Multiple structural alignment is a fundamental problem in structural genomics. In this article, we define a reliable and robust algorithm, MUSTANG (MUltiple STructural AligNment AlGorithm), for the alignment of multiple protein structures. Given a set of protein structures, the program constructs a multiple alignment using the spatial information of the C_α atoms in the set. Broadly based on the progressive pairwise heuristic, this algorithm gains accuracy through novel and effective refinement phases. MUSTANG reports the multiple sequence alignment and the corresponding superposition of structures. Alignments generated by MUSTANG are compared with several hand-curated alignments in the literature as well as with the benchmark alignments of 1033 alignment families from the HOMSTRAD database. The performance of MUSTANG was compared with DALI at a pairwise level, and with other multiple structural alignment tools such as POSA, CE-MC, MALECON, and MultiProt. MUSTANG performs comparably to popular pairwise and multiple structural alignment tools for closely related proteins, and performs more reliably than other multiple structural alignment methods on hard data sets containing distantly related proteins or proteins that show conformational changes. Proteins 2006;64:559–574.**
© 2006 Wiley-Liss, Inc.

Key words: protein evolution; multiple protein structural alignment; dynamic programming; superposition; maximal fragment pairs

INTRODUCTION

Alignment of amino acid sequences of proteins is crucial for many purposes in biology, including the study of evolution in protein families, identification of patterns of conservation in sequences that fold into similar structures, homology modeling, and protein crystal structure solution by molecular replacement.

Alignments can be based on sequences, structures, or both. Pairwise sequence alignment is unreliable if the proteins are too far diverged. Although multiple sequence alignments are more accurate, even they are inadequate when dealing with distantly related proteins sharing little sequence similarity. Because structure changes more conservatively than sequence during protein evolution, it is necessary to appeal to the three-dimensional structures to align the sequences of distantly related proteins.

The need for accurate, reliable alignment algorithms is increasingly important given the exponential rise in the number of known protein three-dimensional structures: 33,152 structures in the Protein Data Bank holdings as of 18 October 2005. Interest in evolutionary studies of the wide range of species from which the proteins are isolated requires methods capable of treating very distantly related molecules. Crystallographic packages such as PHASER[1,2] can use ensembles of aligned structures as molecular replacement (MR) probes. This is crucial for solving difficult MR problems where the sequence identity of available MR probes is low.[3] In addition, high-quality multiple alignments, based on structures, will improve the sensitivity of iterative database searching procedures such as PSI-BLAST.[4]

Algorithms and software for pairwise sequence alignment[5–7] and multiple sequence alignment[8–10] have been developed. (See Notredame[11] for review of sequence alignment algorithms.) Several procedures for pairwise structural alignment (e.g., DALI)[12] are also known. These methods are now mature components of the technology of bioinformatics, readily available and in widespread use. The aim of our work is to contribute a robust and reliable algorithm and software for *multiple structural alignment.*

Formulation of the Problem

An alignment is an assignment of residue–residue correspondences. The main purpose of alignment has been to identify homologous residues; that is, residues encoded by bases derived from the same position in the genome sequence of a common ancestor.

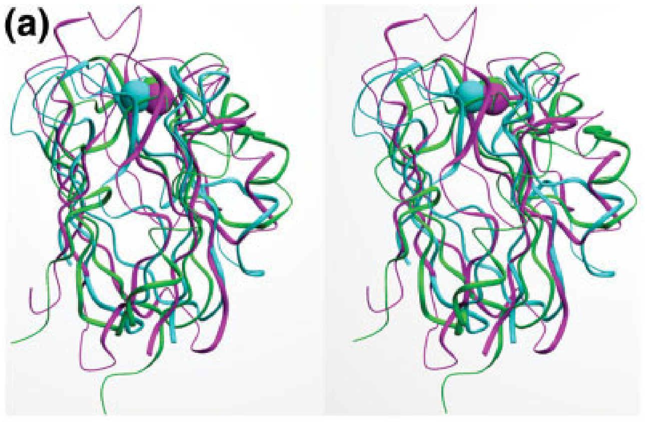
*Combined last authors.

†Correspondence to: Arthur Lesk, 512A Wartik Laboratory, The Pennsylvania State University, Department of Biochemistry and Molecular Biology, University Park, PA 16802. E-mail: aml2@mrc-lmb.cam.ac.uk

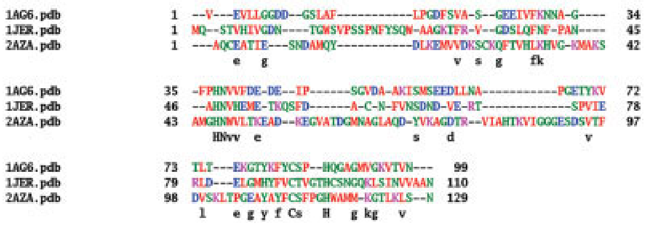
Received 24 October 2005; Accepted 28 November 2005

Published online 30 May 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/prot.20921

(a)



(b)



```
1AG6.pdb    1 --V---EVLLGGDD--GSLAF-----------LPGDFSVA-S--GEEIVFKNNA-G---- 34
1JER.pdb    1 MQ--STVHIVGDN----TGWSVPSSPNFYSQW-AAGKTFR-V--GDSLQFNF-PAN---- 45
2AZA.pdb    1 ---AQCEATIE---SNDAMQY-----------DLKEMVVDKSCKQFTVHLKHVG-KMAKS 42
                    e   g                           v  s  g    fk

1AG6.pdb   35 -FPHNVVFDE-DE--IP------SGVDA-AKISMSEEDLLNA-----------PGETYKV 72
1JER.pdb   46 --AHNVHEME-TKQSFD------A-C-N-FVNSDND-VE-RT-------------SPVIE 78
2AZA.pdb   43 AMGHNWVLTKEAD--KEGVATDGMNAGLAQD-YVKAGDTR--VIAHTKVIGGGESDSVTF 97
                 HNvv  e                        s   d                  v

1AG6.pdb   73 TLT---EKGTYKFYCSP--HQGAGMVGKVTVN---  99
1JER.pdb   79 RLD---ELGMHYFVCTVGTHCSNGQKLSINVVAAN 110
2AZA.pdb   98 DVSKLTPGEAYAYFCSFPGHWAMM-KGTLKLS--N 129
               l    e g y f Cs   H   g kg   v
```

Fig. 1. (**a**) Multiple superposition of three small copper-binding proteins: spinach plastocyanin (PDB code:1ag6), *Alcaligenes denitrificans* azurin (2aza), and cucumber stellacyanin (1jer). (1ag6, cyan; 1jer, green; 2aza, magenta) based on MUSTANG's alignment. Regions aligned by MUSTANG (**b**) are shown as thick ribbons. These prominently include the double–β-sheet structure. The nonalignable regions are shown in thin lines. The respective copper ions bound to the three proteins are shown at the top in large spheres. (**b**) Alignment of the copper-binding proteins that MUSTANG produced. Colors indicate the chemical nature of the amino acid. Red = small hydrophobic including aromatic; blue = acidic; magenta = basic; and green = basic amino acids with hydroxyl groups and/or amine groups. The markup row below each stretch of the multiple alignment indicates completely conserved residues (in UPPERCASE) and partially conserved residues (in lowercase) in each column of the alignment.

For closely related proteins, sequence-based alignments and structure-based alignments give consistent answers (with few exceptions), reflecting evolutionary divergence. For distantly related proteins, however, sequence-based and structural alignments reflect different types of residue correspondences.

In a structure-based alignment, corresponding residues have similar structural contexts. Many homologous proteins share a common core structure, containing regions in which the chain retains the topology of its folding pattern, but varies in geometric details.[13] This retained similarity makes it possible to align the residues of the core. The common core generally comprises a central cluster of secondary structural elements, including the binding site. Peripheral regions outside the common core may refold entirely. It may not be possible to align the refolded peripheral regions by any method, based on sequence or on structure (see Fig. 1).

A structural alignment can therefore distinguish between alignable and nonalignable regions of proteins. This is not possible in a pairwise sequence alignment. Therefore, a structural alignment should not be viewed as merely an extension of sequence-based alignment to more highly diverged proteins. It is providing information about conformational similarities and differences.

Just as the user of a sequence alignment program can control the gappiness by adjusting gap penalties, changing parameters can make a structural alignment method more or less tolerant of conformational difference in deciding what is alignable and what is not. This makes it difficult to compare the results of different alignment methods, in cases where different results do not appear starkly right or wrong, but merely show different degrees of conservativism in choosing what is alignable (see Results and Discussion). This question is discussed in Irving et al.[14]

It is possible to design mixed methods that combine indications from both sequence and structure; see, for example, Šali and Blundell.[15] However, sequences and structures sometimes give conflicting alignment indications. For example:

1. **Serpins in different conformational states.** Protease inhibitors in the Serpin family can take on native and latent states of identical primary structure, but different folding topology.[16] Because the sequences are identical, the sequence alignment is trivial. However, a substantial part of the molecule conserves neither global nor even local structural similarity, and a structural alignment would report the structurally nonconserved regions as nonalignable.
2. **Domain swapping.** It is not uncommon for a protein to exist as a monomer, but a close relative to form a domain-swapped dimer. In such cases, a sequence alignment would align the first protein with one domain of the dimer. A structure-alignment method might align the sequence of the monomeric protein with a combination of fragments from both domains of the dimer.
3. **In some cases, evolution may alter the conformation of regions** even if the sequence divergence is low.

These considerations imply that the first challenge in multiple structural alignment is a proper formulation of the problem.

Multiple structural alignment inherits the difficulties in its formulation from the pairwise structural alignment case.

The structural alignment problem can be formulated in three broad ways:

1. Reduction of the problem to one dimension by encoding the individual conformations of successive residues, and then applying sequence alignment techniques.[17,18] However, this method is unreliable especially when elements of secondary structure are deleted, as occurs, for example, in the cytochrome *c* family, or even when there is a major difference in the lengths of secondary structural elements.
2. Extracting maximal common substructures that can be globally well superposed by rigid-body translations

and rotations.[19] Such methods are not robust if there are conformational changes.

3. Methods based on similarities of distance maps[12] or contact maps.[20] Lesk and Chothia[21] noted that the residue-contact pattern is the most deeply conserved feature of distantly related proteins. Structural alignment methods based on this observation, notably DALI, provide the best results for very distantly related proteins.[22]

Most of the published methods for pairwise structural alignments use either rigid-body superpositions of substructures, which minimize metrics such as root-mean-square deviations (RMSD),[23–26] or dynamic programming.[27] Some methods alternate them, for example, Gerstein and Levitt.[28] A few pairwise methods use geometric techniques to find similar substructures.[29] DALI, which compares distance maps, is the most popular and widely used pairwise structural alignment tool.[12] (See Koehl[30] for a review.)

Were the difficulty of formulating the question not problematic enough, multiple structural alignment is inherently computationally harder than multiple sequence alignment, which itself has been shown to be Non-deterministic Polynomial-time hard (NP-hard).[31,32] Goldman and colleagues[33] have shown that even simplified versions of the structural alignment problem are NP-hard. Caprara and coworkers[20] observe that the theory that can provide rigorous support for structural comparisons is almost nonexistent, as the problems are a blend of continuous–geometric and combinatorial–discrete mathematics.

Review of Previous Work on Multiple Structural Alignment

Most previously published methods for multiple structural alignment resort to approximate pairwise heuristics to find a multiple alignment in polynomial time. Existing methods vary in both their algorithmic and weighting schemes.

Previous work on the multiple structural alignment problem includes the following: Guda et al.[34,35] use combinatorial extension (CE)[36] to perform all pairwise alignments from which a seed alignment is generated which then is iteratively refined using a Monte Carlo optimisation technique. POSA[37] uses a progressive pairwise heuristic that combines the nonlinear multiple alignment approach of Lee and coworkers[38] and the pairwise structural alignment method of Ye and Godzik.[39] Ye and Janardan[40] give an algorithm that is an approximation to the optimal alignment problem that minimises the sum-of-pairs distance using a consensus structure. This is similar to the bounded-error approximation method for sum-of-pairs sequence alignment.[41]

Shatsky and colleagues[42] attempt to find the largest geometric core of a subset of proteins in a set using a plane sweeping technique from computational geometry. Dror and colleagues,[43] and Leibowitz and coworkers[44] use a geometric hashing technique to detect structurally similar subunits, also for subsets of input structures. Russell and Barton,[45] Sutcliffe and coworkers,[46] Lesk,[47] and Lesk and Fordham[48] use rigid-body superpositions. Taylor and colleagues[49] and Taylor and Orengo[50] use iterative double dynamic programming, progressively. Gerstein and Levitt[28] use an algorithm that alternates between dynamic programming and rigid-body superpositions. The multiple alignment methods of Šali and Blundell,[15] Ding and colleagues,[51] and May and Johnson[52] also use the same algorithmic scheme of progressive alignment.[53]

Once an alignment (set of equivalences) has been determined, algorithms for simultaneous rigid-body superposition of multiple structures appear in Shapiro et al.[54] and Diamond.[55]

MUSTANG: A MUltiple STructural AligNment AlGorithm

In this article, we present an algorithm for multiple structural alignment, MUSTANG (see Materials and Methods). MUSTANG aligns residues on the basis of similarity in patterns of both residue–residue contacts and local structural topology. In this respect it seeks to extend the spirit, and thereby the success, of DALI, from pairwise to multiple structural alignment.

MUSTANG uses a progressive pairwise framework to build the final multiple structural alignment. At the core of the method is a robust scoring scheme for pairwise alignments. This makes possible the use of a simple dynamic programming algorithm for all pairs of structures in the set, to gather accurate pairwise residue–residue equivalences, without the need for gap penalties, which are known to be troublesome. Before the final multiple alignment is constructed along a guide tree, a special extension phase is undertaken in which the pairwise scores of correspondences are recalculated in the context of all the remaining structures. This significantly reduces the problem of making incorrect greedy choices while building the final alignment. The design of the MUSTANG scoring scheme lends it the ability to handle conformational flexibilities such as hinge rotations.

Results of numerous comparisons with other methods clearly show that MUSTANG is very reliable in generating quality multiple structural alignments even on very distantly related data sets containing structural deformations.

MATERIALS AND METHODS

Notation

Let $\mathcal{S} = (S^1, \ldots, S^n)$ be a set of n protein structures. Each structure is represented by the set of coordinates of the C_α atoms, in order from N- to C-terminus. (In this paper the word *residue* refers to the C_α coordinates of a residue. The numbers of residues in the structures are $(L_1, \ldots, L_n)$ respectively. S^i_j denotes the jth residue of the ith structure, for $i = (1, \ldots, n)$ and $j = (1, \ldots, L_i)$.

A multiple structural alignment $\mathcal{A}$ of $\mathcal{S}$ is a set of equivalences represented by an $n \times l$ matrix, $A = (\mathbf{a}_{ij})_{1 \le i \le n,\ 1 \le j \le l}$, such that:

1. $\max(L_1, \ldots, L_n) \le l \le L_1 + \ldots + L_n$.
2. Each element of A is either one of the residues S^i_k, or a

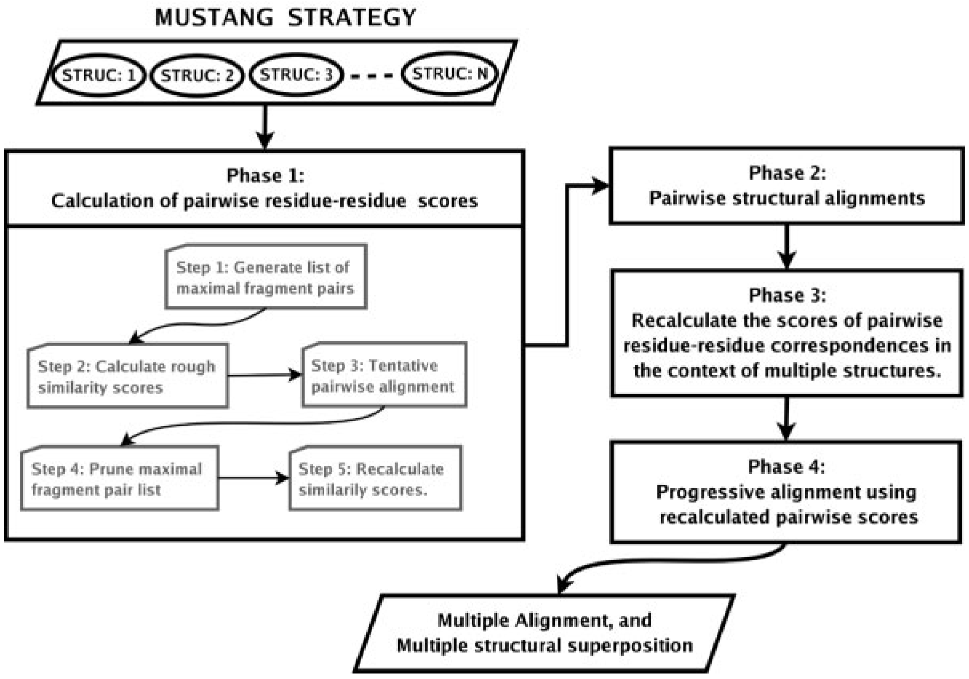

MUSTANG STRATEGY
STRUC: 1
STRUC: 2
STRUC: 3
STRUC: N
Phase 1: Calculation of pairwise residue-residue scores
Step 1: Generate list of maximal fragment pairs
Step 2: Calculate rough similarity scores
Step 3: Tentative pairwise alignment
Step 4: Prune maximal fragment pair list
Step 5: Recalculate similarily scores.
Phase 2: Pairwise structural alignments
Phase 3: Recalculate the scores of pairwise residue-residue correspondences in the context of multiple structures.
Phase 4: Progressive alignment using recalculated pairwise scores
Multiple Alignment, and Multiple structural superposition

Fig. 2. An overview of the MUSTANG algorithm.

special null vector called *gap* and denoted by the symbol '–'.

3. The ith row of A contains the ordered set of C_α positions of structure i, possibly with gaps interspersed. (In particular, this implies that the alignment preserves the order of the residues.)

Given the matrix of equivalences A, let $O_i(R_i, T_i)$, $1 \le i \le n$, be a set of rigid-body transformations (each comprising a proper rotation matrix R_i, $\det(R_i) = +1$, and a translation vector T_i) associated with each row in A, calculated to effect an optimal superposition of the structures.[54,55]

The MUSTANG Algorithm

An overview of MUSTANG is presented in Figure 2. The program extracts the C_α atoms from the n protein structures in the input. Note that, consistent with our focus on the more difficult problems arising from very distantly related structures, MUSTANG completely ignores the amino-acid sequences of the proteins.

The following four major phases are used to construct a multiple alignment:

1. Calculation of scores of pairwise residue–residue correspondences.
2. Pairwise structural alignments.
3. Recalculation of scores of residue–residue correspondences in the context of multiple structures.
4. Progressive alignment along the guide tree.

Phase 1: Calculation of scores of pairwise residue–residue correspondences

The RMSD after optimal rigid-body superpositions has been one of the earliest themes for measuring the degree of similarity between point sets. Distance matrices, on the other hand, containing all pairwise distances, have also been used to compare protein conformations.[56,57] Havel and coworkers[58] gave a method to recover a three-dimensional point set from its distance matrix (up to overall chirality), showing that the distance matrix contains complete structural information. In a preliminary step, MUSTANG calculates the complete distance matrix between the C_α atoms for all structures submitted, to provide data used in the alignment procedure.

The basic idea behind MUSTANG's scoring scheme for the initial pairwise structural alignment step is to use optimal superpositions and values of RMSD to detect similar contiguous substructures within two structures, and, using a similarity measure based on comparing the contact patterns in the substructures, to obtain a numerical statement of quality for each possible residue–residue correspondence between two structures. We use the results of these residue–residue scoring schemes, derived independently for each pair of structures, to align each pair of structures by a simple dynamic programming algorithm, which runs in time quadratic in sequence length.

In most sequence alignment methods, gap penalties are introduced to control the insertions and deletions. This is required because at every step the dynamic programming method considers only a pair of residues and a score based on the *log-odds-of-exchange* of the two residues. MUSTANG, however, derives a scoring scheme in which the score associated with aligning a pair of residues is influenced by the maximal similar substructural fragments to which they belong, first locally and then globally. In this approach, gap penalties are not needed.

MUSTANG's scoring procedure involves the following steps:

Step 1 Generate a list of all maximal similar contiguous substructures.
Step 2 Calculate rough similarity scores for individual residue–residue correspondences.
Step 3 Generate a tentative pairwise alignment.
Step 4 Prune the list of maximal similar substructures.
Step 5 Recalculate similarity scores for individual residue–residue correspondences.

Step 1. Generate a list of all maximal similar substructures. The fragment, or contiguous substructure, of length l, starting at position j in the ith structure, and extending to position $j + l - 1$, will be denoted by $f_j^i(l)$. (We do not require that the chain remain unbroken within a contiguous substructure, but only that the residues appear consecutively.) Two equal-length fragments $f_j^i(l)$ and $f_{j'}^{i'}(l)$ are said to be structurally similar if they can be superposed — with successive atoms in each sequence corresponding to each other — by a proper orthogonal transformation, with a RMSD no greater than some predefined threshold, ε. MUSTANG uses the method of Kearsley[25] that solves the optimal least-squares superposition problem as an eigenvector problem in quaternion parameters with a complexity of $O(l)$, where l is the number of points being superposed. To reduce noise, we demand that $l \geq l_{min}$ for all similar fragments. The definition of similar fragments therefore depends on two parameters, ε and l_{min}. After testing several values for ε and l_{min}, we empirically determined that the values $l_{min} = 6$ and $\varepsilon = 1.75$ Å gave the best results.

A *maximal fragment pair* (MFP) $m_{jj'}^{ii'}(l)$ is defined as a set of two similar fragments $f_j^i(l)$ and $f_{j'}^{i'}(l)$ that are not contained in longer similar fragments that share the same N-terminus. In other words, a similar fragment pair is maximal if adding the successor C_α atoms to the C-termini of both fragments produces extended fragments that cannot be superposed with RMSD $\leq \varepsilon$.

The goal of this step is to construct a list $\mathscr{L}$ of all MFPs for all pairs of structures. The procedure is as follows: For each pair of structures i and i', the program superposes all pairs of fragments, $f_j^i(l_{min})$ and $f_{j'}^{i'}(l_{min})$, with the N-terminal residue j of each fragment taking all values $1 \leq j \leq L_i - l_{min} + 1$ where L_i is the length of structure i, (and similarly for structure i'). If a pair of fragments can be optimally superposed with RMSD $\leq \varepsilon$ (i.e., the fragments are similar) the program iteratively tries to *extend* them by appending successive positions to the C-termini of both fragments, to determine the largest extended fragments that still fit to $\leq \varepsilon$. Note that we do not extend similar fragments by adding residues to the N-termini. This procedure will conclude by determining an MFP of some length l, $m_{jj'}^{ii'}(l)$, where $l_{min} \leq l \leq \min(L_i - j, L_{i'} - j') + 1$. Each such MFP is stored in the list $\mathscr{L}$. Note that because each MFP is determined starting from a pair of fragments the starting points (that is, the N-terminal residues) of which are not both the same, the list cannot contain any MFP that is properly contained in another. It *is* possible for MFPs to overlap unless the length of the extension of a new MFP over an existing one is less than or equal to $\lceil l_{min}/2 \rceil$.

We use a generous similarity threshold $\varepsilon = 1.75$ Å, in order to avoid excluding fragments of potential interest. A concomitant problem is that in some cases a MFP can include, in its terminal regions, a few correspondences that are, in reality, mismatches. Such cases are pronounced in the MFPs ending in between secondary structural regions such as helices and strands which have substructures that fit sufficiently well that mismatches at the termini do not adequately raise the RMSD. To overcome this problem, we demand that every MFP ending in between helices or strands in both the compared structures is truncated from the C-terminus of the fragment back to the the position where the MFP first entered into the secondary structural region in the terminal part of either of the structures. To be able to do this, MUSTANG must identify with some accuracy the start and end positions of helices and strands. MUSTANG uses the method described in Richards and Kundrot[59] to determine secondary structures using only the information of the C_α coordinates. Restricting the determination of secondary structure to helices and strands (because these are the regions where the major mismatches in MFPs occur) containing contiguous segments of more than four residues, the program determines the initial and final positions of all helices and strands in a structure through the use of the distance masks available in Richards and Kundrot.[59] Note that the secondary structural information is only used while building the list of MFPs and never again in subsequent stages of alignment construction.

The entire step 1 has a theoretical worst case time complexity of $O(k^3)$. In practice we find it to be $O(k^2)$ [$k = \max(L_i, L_{j'})$]. The worst case space complexity of $\mathscr{L}$ is $O(L_i L_j)$.

We could proceed to Step 5 (the recalculation of similarity scores) directly using the list of MFPs, $\mathscr{L}$. However the computation in Step 5 grows quadratically with the size of $\mathscr{L}$ as $O(L_i^2 L_j^2)$. This can be prohibitively expensive. Steps 2 to 4 reduce the size of $\mathscr{L}$ in order to speed up Step 5.

Step 2: Calculate rough similarity scores for individual residue–residue correspondences. We now discuss the calculation of initial scores for the individual correspondences available in the list $\mathscr{L}$. The basic idea of this step is to develop a rough pairwise residue–residue scoring scheme to generate a tentative pairwise alignment (Step 3) which is used to prune $\mathscr{L}$ (Step 4).

Let $W^{ii'} = (w_{jj'}^{ii'})_{1 \leq j \leq L_i, 1 \leq j' \leq L_{i'}}$ $(1 \leq i \neq i' \leq n)$ be a matrix of scores for every possible set of residue–residue correspondences between the two structures S^i, and $S^{i'}$. W is initially set to zero. For all MFPs $m_{jj'}^{ii'}(l) \in \mathscr{L}$, the score of every correspondence between the residues S_{j+t}^i and $S_{j'+t}^{i'}$, $w_{j+t,j'+t}^{ii'}$ $(0 \leq t \leq l - 1)$ is calculated as

$$w^{ii'}_{j+t,j'+t} = \sum_{\forall 0 \le p \le l-1} \sum_{\forall p+1 \le q \le l-1} \phi(i, i', j, j', p, q) + \sum_{\forall 0 \le q \le l-1} \phi(i, i', j, j', t, q). \quad (1)$$

where $\phi(i, i', j, j', x, y)$, a slight modification of *elastic similarity function* introduced in Holm and Sander,[12] is calculated as

$$\phi(i, i', j, j', x, y) = \begin{cases} \left(\theta - \dfrac{|d^{i}_{j+x,j+y} - d^{i'}_{j'+x,j'+y}|}{d^*}\right) \times \omega(d^*), & x \neq y \\ 0, & x = y \end{cases} \quad (2)$$

where d^{i}_{jk} is the distance between the C_α atoms of residues j and k in ith structure, $d^* = (d^{i}_{j+x,j+y} + d^{i'}_{j'+x,j'+y})/2$, $\theta = 0.20$, and ω is an envelope function calculated as $\omega(d) = e^{-(d^2/400)}$. See Holm and Sander[12] for details.

The $w^{ii'}_{j+t,j'+t}$ shown in Equation 1 has two components. The first denotes the complete similarity score of comparing the two fragments in the $m^{ii'}_{jj'}(l)$. The latter is the similarity score of the individual correspondence between S^{i}_{j+t} and $S^{i'}_{j'+t}$ within the structural environment of the MFP.

In most data sets, this initial scoring scheme itself is fairly accurate in aligning major chunks of the conserved core in the pairwise alignments. But a major disadvantage of this initial scoring scheme is that the alignment that is produced using these scores will simply be a concatenation of maximal local alignments without taking into consideration the arrangement of the various maximal local fragments in space. Therefore, this step is used by MUSTANG only as an effective way to speed-up Step 5 where the proper scoring matrix is derived to calculate pairwise structural alignments.

Before moving on to Step 3 all the weights $w^{ii'}_{jj'}$ which remain 0 are changed to $-\infty$ to avoid the possibility of a correspondence between such positions in the tentative alignment step. This is justified because there has been no evidence of structural similarity in this region.

Step 3: Tentative pairwise alignment. A tentative alignment using the rough scores of all possible pairwise correspondences is generated using a simple dynamic programming algorithm. Let $\Delta = (\delta_{jk})_{0 \le j \le L_i, 0 \le k \le L_{i'}}$ $(1 \le i \neq i' \le n)$ be a dynamic programming matrix with the following recursive update rules:

$$\delta_{jk} = 0, \qquad j = 0 \vee k = 0$$

$$\delta_{jk} = \max \begin{cases} \delta_{j-1,k-1} + w^{ii'}_{jk} \\ \delta_{j,k-1} \\ \delta_{j-1,k} \end{cases} ,1 \le j \le L_i, 1 \le k \le L_{i'} \quad (3)$$

The matrix Δ is filled from from δ_{00} to $\delta_{L_i L_{i'}}$ in either row-major, or column-major or anti-diagonal fashion. Each element in the matrix holds a pointer to mark one of the preceding elements that contributed its scoring using the update rules given above. The tentative alignment is generated based on the backtracking from the element $\delta_{L_i,L_{i'}}$ to δ_{00}.

Let $\mathcal{T}$ be the data structure that stores the set of equivalences defined by the resulting tentative pairwise alignment.

Step 4: Pruning the list of maximal similar substructures. For any two structures S^i and $S^{i'}$ the computation involved in the Step 5 grows quadratically with the size of the list $\mathcal{L}$ and hence has a theoretical complexity of $O(L_i^2 L_{i'}^2)$. Because a large number of MFPs $\in \mathcal{L}$ are redundant, the goal of this step is to reduce the size of the $\mathcal{L}$ to speed up Step 5.

Given the set of pairwise equivalences in the tentative alignment $\mathcal{T}$, the list of MFPs $\mathcal{L}$, is now pruned into a new list, $\mathcal{L}'$ as follows:

Every MFP of the form $m^{ii'}_{jj'}(l)$ is deleted if there exists no correspondence between S^{i}_{j+t} and $S^{i'}_{j'+t}$ $(0 < t < l - 1)$ in the MFP such that:

(1) (assuming that S^{i}_{j+t} has an equivalence with some $S^{i'}_{y}$ in the tentative alignment $\mathcal{T}$) if $(j + t) \le y$ then $(j + t) + \lambda > y$ or else $(j + t) - \lambda < y$, or
(2) (assuming that $S^{i'}_{j'+t}$ has an equivalence with some S^{i}_{x} in the tentative alignment $\mathcal{T}$) if $(j' + t) \le x$ then $(j' + t) + \lambda > x$ or else $(j' + t) - \lambda < x$

where λ is some positive constant that controls the degree of pruning of $\mathcal{L}$. In other words, only those MFPs are considered that are within a range of $\pm\lambda$ from any of correspondences in the tentative pairwise alignment. Those that are outside this range are pruned. The value of λ has been quite generously set to 30 which has produced accurate alignments in all the cases on which the program was tested. Such a step drastically reduces the size of $\mathcal{L}$ and hence will speed up Step 5.

Step 5: Recalculate similarity scores for individual residue–residue correspondences. At this stage we have the matrix $W^{ii'}$ of initial scores of all pairwise correspondences between the residues in ith and i'th structures, as well as a (pruned) list of MFPs, $\mathcal{L}'$. In this step the scores in $W^{ii'}$ are recalculated using $\mathcal{L}'$ and added over the initial ones as follows:

Every pair of MFPs $\in \mathcal{L}'$ of the form $m^{ii'}_{jj'}(l')$ and $m^{ii'}_{kk'}(l')$ is superposed jointly. That is, the sets of points superposed are the concatenations of the lists of points in the two corresponding fragments chosen from each structure. Let the RMSD of the joint superposition be ρ. If ρ is no greater than a second threshold ε', the scores of all individual correspondences in both the above MFPs, $w^{ii'}_{j+t,j'+t}$ and $w^{ii'}_{k+t',k'+t'}$ $(0 \le t(t') \le l(l') - 1)$ are calculated as,

$$w^{ii'}_{j+t,j'+t} = w^{ii'}_{j+t,j'+t} + (\varepsilon' - \rho)^2 \times \left(\begin{array}{l} \displaystyle\sum_{0 \le p \le l-1} \sum_{0 \le q \le l'-1} \bar{\phi}(i,i',j,j',k,k',p,q) \\ \displaystyle + \sum_{0 \le q \le l'-1} \bar{\phi}(i,i',j,j',k,k',t,q) \end{array} \right) \quad (4)$$

$$w^{ii'}_{k+t',k'+t'} = w^{ii'}_{k+t',k'+t'} + (\varepsilon' - \rho)^2 \times \left(\begin{array}{l} \sum_{0 \le p \le l-1} \sum_{0 \le q \le l'-1} \bar{\phi}(i,i',j,j',k,k',p,q) \\ + \sum_{0 \le p \le l-1} \bar{\phi}(i,i',j,j',k,k',p,t') \end{array} \right)$$

where $\bar{\phi}(i,i',j,j',k,k',x,y)$ in this step is calculated slightly differently from Equation 2 as,

$$\bar{\phi}(i,i',j,j',k,k',x,y) = \begin{cases} \left(\theta - \frac{|d^{i}_{j+x,k+y} - d^{i'}_{j'+x,k'+y}|}{d^*}\right) \times \omega(d^*), & cond1 \\ \theta, & otherwise \end{cases} \qquad (5)$$

where $cond1 \equiv (j + x) \neq (k + y) \wedge (j' + x) \neq (k' + y)$, $d^*=(d^{i}_{j+x,k+y}+d^{i'}_{j'+x,k'+y})/2$, the constant θ, distance d^{i}_{jk}, and the envelope function ω have the same definitions as used in Equation 2.

For increased accuracy we avoid joint superpositions of any two MFPs that have in common more than 80% of either of the fragments. This constraint prevents accumulation of unnecessary weights.

The value of ε' has been quite liberally fixed at 6.5. This generous threshold in conjunction with the quadratic reward factor $(\varepsilon' - \rho)^2$ in Equation 4 gives MUSTANG its ability to align efficiently structures with conformational changes.

Phase 2: Gathering equivalences between residues from all pairs of structures in the ensemble (pairwise structural alignments)

Pairwise structural alignments are generated here using the scoring matrix $W^{ii'}$ ($\forall 1 \le i < i' \le n$). The dynamic programming method discussed in Step 3 above with update rules shown in Equation 3 is used again. Let $\mathcal{P}^{ii'}$ be the data structure created by storing the equivalences derived through every pairwise alignment in the ensemble.

Phase 3: Recalculation of scores of residue–residue correspondences in the context of multiple structures (extension phase)

The progressive pairwise alignment approach for constructing multiple alignments involves merging a series of pairwise alignments along a guide tree. A lot of incorrect greedy choices are made in such a procedure. Also, the order in which the pairwise alignments are forced affects the accuracy and quality of the final multiple alignment. The *extension phase* used in MUSTANG helps reduce the number of greedy choices made in the final progressive pairwise alignment phase. The strategy MUSTANG uses is close in spirit to the ones used previously in sequence comparison methods such as Notredame and colleagues,[9] Morgenstern,[60] and Neuwald and co-workers,[61] but the details of the procedure differ significantly.

A new matrix $\overline{W}^{ii'} = (\overline{w}^{ii'}_{jj'})_{1 \le j \le L_i, 1 \le j' \le L_{i'}}$ $(1 \le i \neq i' \le n)$ is generated in this phase containing scores for every possible residue–residue correspondence between the two structures. The scores at the start are initialised to zero. First, for every equivalence between the residues S^{i}_{x} and $S^{i'}_{y}$ in the pairwise alignment $\mathcal{P}^{ii'}$ we assign $\overline{w}^{ii'}_{xy} = w^{ii'}_{xy}$.

We next establish transitive correspondences between every pair of structures S^i and $S^{i'}$ $(1 \le i \neq i' \le n)$ through all possible intermediate structures S^j $(1 \le j \neq i \wedge j \neq i' \le n)$. Given any equivalence between S^{i}_{x} and $S^{i'}_{y}$ (or a *gap*) $\in \mathcal{P}^{ii'}$, let S^{i}_{x} be equivalent to S^{j}_{z} (and not equivalent to a *gap*) in $\mathcal{P}^{ij}$, and let S^{j}_{z} in turn be equivalent to $S^{i'}_{y'}$ (and not equivalent to a *gap*) in $\mathcal{P}^{ji'}$, then we say that there is a (transitive) correspondence between S^{i}_{x} and $S^{i'}_{y'}$ through the intermediate S^{j}_{z}. The score of the transitive correspondence, $\overline{w}^{ii'}_{xy'}$, is calculated as,

$$\overline{w}^{ii'}_{xy'} = \max(w^{ii'}_{xy'}, w^{ii'}_{x'y'}),\ S^{i}_{x} \equiv S^{i'}_{y} \wedge S^{i'}_{y'} \equiv S^{i}_{x'} \wedge \overline{w}^{ii'}_{xy'} < \max(w^{ii'}_{xy}, w^{ii'}_{x'y'}) \qquad (6)$$

$$\overline{w}^{ii'}_{xy'} = w^{ii'}_{xy},\ S^{i}_{x} \equiv S^{i'}_{y} \wedge S^{i'}_{y'} \equiv gap \wedge \overline{w}^{ii'}_{xy'} < w^{ii'}_{xy} \qquad (7)$$

$$\overline{w}^{ii'}_{xy'} = w^{ii'}_{x'y'},\ S^{i}_{x} \equiv gap \wedge S^{i'}_{y'} \equiv S^{i}_{x'} \wedge \overline{w}^{ii'}_{xy'} < w^{ii'}_{x'y'} \qquad (8)$$

$$\overline{w}^{ii'}_{xy'} = 1,\ S^{i}_{x} \equiv gap \wedge S^{i'}_{y'} \equiv gap \wedge \overline{w}^{ii'}_{xy'} < 1 \qquad (9)$$

$$\overline{w}^{ii'}_{xy'} = \overline{w}^{ii'}_{xy'} + 1,\ otherwise \qquad (10)$$

(In Equations 6 to 10, read $X \equiv Y$ as X is equivalent to Y.)

Equations 6 to 9 deal with inconsistent transitive correspondences (with respect to the original pairwise equivalences). With the first evidence of an inconsistent transitive correspondence between the residues S^{i}_{x} and $S^{i'}_{y'}$, the score of the correspondence is scaled up to enable it to be on a level ground with the score of the original pairwise equivalence. This will allow the new correspondence to compete actively in the dynamic programming algorithm. Equation 10, on the other hand, will increment the score of a transitive correspondence by one, whether the correspondence is reinforcing a pairwise equivalence or providing subsequent evidence of an inconsistent correspondence.

In summary, the more intermediate structures supporting the alignment of a pair of residues, the higher the increments to its score and hence the greater the chances that the pair will correspond in the final alignment.

Figure 3 shows an instructive but idealized example of the extension phase. Figure 3a shows four (planar) structures in different colors with residues denoted in small circles numbered starting from one, left to right. Let *A, B, C, D* denote the structures coloured in blue, green, red, and grey, respectively. Let A_i, $1 \le i \le 6$ denote the respective residues of structure A (and similarly for other structures).

Figure 3b shows all possible pairwise equivalences (alignments) of these structures which, let us assume, MUSTANG generated in phase 2. Figure 3c shows the extension step in which the pairwise weights between A and B are recalculated in the context of remaining structures C and D using C and D as intermediates. This step starts by assigning the extended weights corresponding to the equivalences in the pairwise alignment of A and B to their respective pairwise weights. Let the thickness of black lines in the alignment be a visual representation of their weights. The pairwise weights of A and B are first extended using C. We have A_2 equivalent to C_1 in the pairwise alignment of A and C, and C_1 equivalent to B_2 in the pairwise alignment of B and C. Then the transitive correspondence between A_2 and B_2 is established. In this case the transitive correspondence reinforces the original pairwise equivalence between A_2 and B_2 in the pairwise alignment of A and B and hence the extended weight of this correspondence is incremented by one (Eq. 10) as reflected in the increase in the thickness of the corresponding black line in the figure. The weight between A_3 and B_3 is incremented similarly through C_2.

In contrast, the transitive correspondence between A_4 and B_4 through C_3 is inconsistent with the pairwise equivalence between A_4 and *gap* in the pairwise alignment of A and B. Hence, a correspondence between A_4 and B_4 is established with a weight equal to the weight of the original pairwise equivalence between B_4 and A_5 (Eq. 8). The inconsistent transitive correspondence between A_5 and B_5 (through the intermediate C_4) is assigned the maximum of the weights of the pairwise equivalences, $A_5 \equiv B_4$ and $B_5 \equiv A_6$ (Eq. 6).

Having extended the pairwise weights between A and B through the intermediate C, we now extend the same weights using D as the intermediate structure. The transitive correspondences $A_1 \equiv B_1$ (through D_1), $A_2 \equiv B_2$ (through D_2), and $A_3 \equiv B_3$ (through D_3) are all consistent with the original pairwise alignments; hence the corresponding weights are incremented by one as before. In this case again, the transitive correspondences $A_4 \equiv B_4$ (through D_4), and $A_5 \equiv B_5$ (through D_5) are inconsistent with the equivalences from the pairwise alignment. However, we had already encountered these (transitive) correspondences during the previous extension through C, hence the corresponding extended weights are merely incremented by one in these cases. By the end of all the extensions, we now have a new scoring matrix for A and B with weights that reflect the information of equivalences from other structures (C and D) in the ensemble. Such extensions are undertaken for all pairs before using these new extended weights in the final progressive alignment stage.

The computation in this phase grows as $O(n^3 l^2)$ [$l = \max(L_1, \ldots, L_n)$].

Phase 4: Progressive alignment phase

The progressive alignment phase involves finding a guide tree along which the final multiple alignment is constructed.[8,62] The problem of building the phylogenetic tree of proteins based on structural information is still an open problem.[37] But at this step we merely aim to generate a quick and practical guide tree in order to align the structures.

In order to calculate a distance–divergence matrix K, the normalized alignment score of each pairwise (structural) alignment, $\zeta_{ii'} (\forall 1 \le i \le i' \le n)$, is first calculated using the equivalences in the pairwise alignment $\mathcal{P}^{ii'}$ as,

$$\zeta_{ii'} = \sum_{(j,j') \in P^{ii'}} \left(\overline{\mathrm{w}}^{ii'}_{jj'} \times \frac{\max_{cond1}(\overline{\mathrm{w}}^{kk'}_{ll'})}{\max_{cond2}(\overline{\mathrm{w}}^{ii'}_{ll'})} \right) \tag{11}$$

where $cond1 \equiv [(\forall 1 \le k < k' \le n) \wedge (\forall 1 \le l \le L_k) \wedge (\forall 1 \le l' \le L_{k'})]$, and $cond2 \equiv ((\forall 1 \le l \le L_i) \wedge (\forall 1 \le l' \le L_{i'}))$.

The similarity score $\zeta_{ii'}$ is then transformed into a distance divergence score $K_{ii'}$ as,

$$K_{ii'} = -\ln\left(\frac{\zeta_{ii'}}{\max_{1 \le j < j' < n} \zeta_{jj'}} \right) \tag{12}$$

The binary guide tree is constructed using the neighbor-joining method[63] applied to the distance divergence matrix, K. In the final alignment stage, alignments are forced between two clusters of subalignments. In MUSTANG we use a profile function calculated as the sum of scores of all pairwise correspondences (SP) between any two columns in the two subalignments:

$$SP^{xy} = \sum_{i,j \in x} \sum_{i',j' \in y} (\overline{\mathrm{w}}^{ii'}_{jj'}) \tag{13}$$

where x and y are two columns of alignments — one from each cluster — that are to be joined together, $i(i')$, $j(j')$ are the indices of the structure and the position of the residue in the chain, respectively.

The result is a multiple alignment. From the correspondences in the multiple alignment, all the structures are optimally superposed on each other using the method described in Diamond,[55] applied to the subset of positions occupied in all sequences by genuine residues (i.e., the set of positions for which no sequence contains a gap).

Implementation of MUSTANG

The MUSTANG algorithm has been implemented using a conservative subset of C++. The experiments on various data sets were conducted on a Fedora Core 3 Linux platform with 1.2 GHz CPU and 256 MB primary memory. The speed of the program depends on the following factors:

1. The number of input structures in the ensemble.
2. The sizes of input structures.
3. The sizes of the maximal fragment pairs list.

For example, the 11 Set 3 globin structures (see Table I) took 133 s to align.

The command line program (including its source) is available from http://www.cs.mu.oz.au/~arun/mustang. The program accepts, as input, coordinates in PDB format. MUSTANG reports the alignment of sequences in an HTML format (appearing as in Fig. 1b) and optionally in many other formats. It also generates as output the structures brought to a common coordinate frame in which they are optimally superposed based on the multiple alignment.

RESULTS AND DISCUSSION

Tests of MUSTANG Alignments and Comparison with Other Programs

The performance of MUSTANG was tested on numerous data sets. Several published alignments were used in the evaluation. Of particular interest are tests on difficult cases, involving very highly diverged proteins or struc-

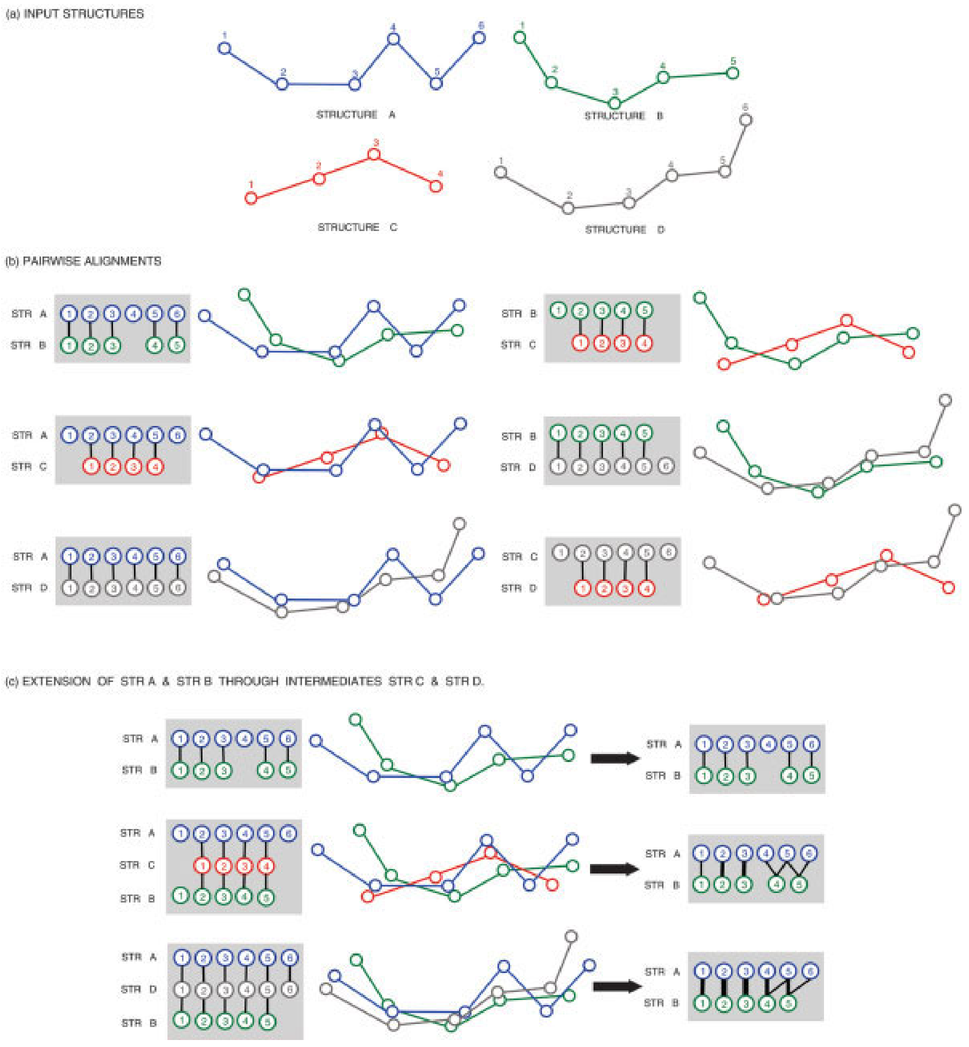


Fig. 3. An idealized example of the extension phase where weights are recalculated (extended) for each pair, in the context of the multiple alignment. (**a**) Input (hypothetical) structures A (blue), B (green), C (red), D (gray). (**b**) Various pairwise alignments of the input structures; (**c**) illustration of extension of pairwise correspondences between structure A and structure B using structures C, and D as intermediates.

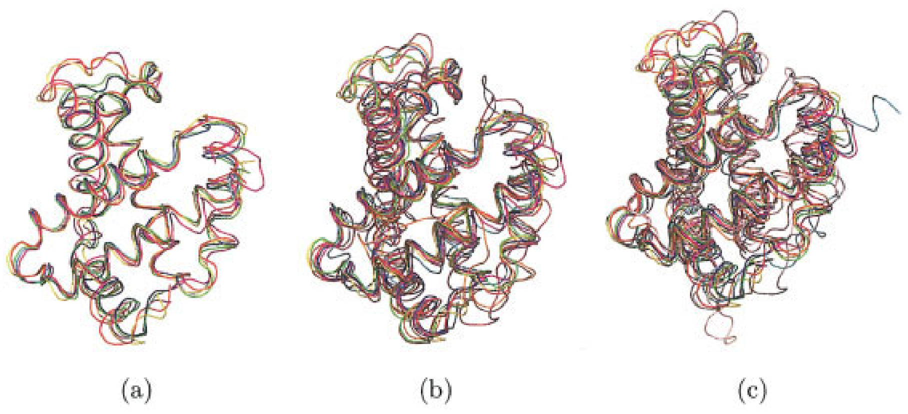


Fig. 4. Visualization of superpositions of the Globin structures in Table I (**a**) Set 1, (**b**) Set 2, and (**c**) Set 3 based on the MUSTANG's alignment.

TABLE I. Globin and Serine Proteinase Data Sets

Data Set	Number of Structures	PDB Codes, followed in some cases by chain identifiers a or b
Globins[a]		
Set 1	5	1hhoa, 2dhba, 1hhob, 2dhbb, 1mbd
Set 2	9	1hhoa, 2dhba, 1hhob, 2dhbb, 1mbd, 2lhb, 1hbg, 1eco, 2lh7
Set 3	11	1hhoa, 2dhba, 1hhob, 2dhbb, 1mbd, 1eco, 2lh7, 4vhb, 1dlw, 1dly, 1idr
Serine Proteinases		
Set 1	7	3est, 2pka, 1ton, 3rp2, 4ptp, 5cha, 1ppb
Set 2	13	3est, 2pka, 1ton, 3rp2, 4ptp, 5cha, 1ppb, 1sgt, 1arb, 2sga 3sgb, 2alp, 2snv

[a]There are some minor differences between these contemporary data sets and those used in the work by Lesk and Chothia[21] carried out 25 years ago.

tures which show conformational flexibilities. MUSTANG's performance was also compared with POSA,[37] CE-MC,[35] MultiProt,[42] and MALECON[64] using our selected data sets as well as those described in the relevant papers. All the HOMSTRAD alignment families were aligned using MUSTANG and comparisons were made with the database benchmarks. Finally, because the reliability of MUSTANG is based on its pairwise structural alignments, and because DALI is widely considered to represent the state of the art in pairwise structural alignment, we compared the results of MUSTANG and DALI over 633 HOMSTRAD families containing only two structures per family.

Comparisons with curated alignments

We ran MUSTANG on several data sets for which manually generated alignments are available.

Globins. The globins are one of the most extensively studied protein families. Lesk and Chothia[21] reported a carefully curated alignment of globin structures then available. These proteins were realigned using MUSTANG.

For these tests, the structures used were grouped into three sets, corresponding to different degrees of divergence. Set 1 contains mammalian globins only: the α and β subunits of human and horse hemoglobin, and sperm whale myoglobin. Set 2 contains all the structures treated in the 1980 work, extending Set 1 by the inclusion of invertebrate and plant globins. Since 1980, the globin family has grown considerably. In particular, a set of *truncated globins* from microorganisms has been discovered, and some of the structures solved and their sequences aligned with the full-length globins.[65,66] Table I contains the PDB codes of these three sets of structures.

MUSTANG's alignments of these Globin data sets can be found at http://www.cs.mu.oz.au/~arun/mustang/globins.html.

The alignment of all the three sets agrees with the published alignments described in Lesk and Chothia[21] and Lesk,[65] except for a few minor disagreements that are only in regions of dissimilar conformation where it is difficult to align with confidence. The inclusion of additional structures in Set 2 over Set 1 did not affect the alignment of common structures. However, the alignment of the Set 2 structures in the Set 2 alignment was slightly better, in some regions, than the alignment of the Set 2 structures within the Set 3 (= Set 2 + three truncated globins) alignment, suggesting in this case, some deterioration. Figure 4 shows the superpositions of the structures based on the alignments of the above sets. Comparison with Figure 1a shows that the conserved core of the full-length globins comprises an unusually high portion of the structure.

Serine proteinases. MUSTANG was run on the Serine proteinase data sets shown in Table I and compared with the published alignments in Lesk and Fordham.[48] Again, Set 1 contains a relatively closely related group of mammalian serine proteinases, and Set 2 contains a more widely diverged set of mammalian and bacterial serine proteinases. (Because of horizontal gene transfer, *Streptomyces griseus* trypsin is effectively a mammalian protein.) MUSTANG's alignments of the above data sets can be found at http://www.cs.mu.oz.au/~arun/mustang/ serine_prot.html. For Set 1, we observe that all the positions in the Lesk and Fordham[48] alignment that are occupied by a residue (not a gap) in every sequence are identically aligned by MUSTANG. The published alignment has 174 columns/positions which are ungapped whereas MUSTANG's alignment has 205 such positions. Note the large difference between the programs in choosing when to align and when not to. In almost all the regions of the alignments where there are disagreements, MUSTANG's alignment is preferable.

In Set 2, the published alignment has 63 ungapped positions whereas MUSTANG's alignment produced 119 such positions in its alignment, consistent with the suggestion that the published alignment is more conservative than MUSTANG in assigning equivalences. However, with a few exceptions, most of the ungapped columns in the published alignment are the same in MUSTANG's alignment. Figure 5a and b show the superpositions of these two sets according to the MUSTANG alignments.

As in the case of the globins, we also compared the effect of inclusion of additional structures in Set 2 on the alignment of the structures common to Set 1 and Set 2. In doing so we not only evaluate the performance of MUSTANG on these examples but also ask how stable MUSTANG is with respect to the addition of more structures to the ensemble. At almost all positions, MUSTANG produced alignments in which the structures common to both sets are equivalently aligned. Almost all of the differences we observe are very minor shifts in noisy regions. In this

TABLE II. Results of Comparisons on the Globin Data Sets (see Table I) across the Multiple Structural Alignment Programs MUSTANG, POSA, and CE-MC

	MUSTANG		POSA		CE-MC	
Globin Data Set	NCORE	RMSD	NCORE	RMSD	NCORE	RMSD
Set 1	139	1.41 Å	139	1.47 Å	138	1.40 Å
Set 2	117	1.94 Å	121	2.08 Å	117	2.10 Å
Set 3	94	2.24 Å	90	2.32 Å	92	2.47 Å

NCORE, The number of positions in the common core; RMSD, average root-mean-square-deviation of the common core.

case the additional structures in Set 2 neither improve the alignment of the Set 1 structures nor degrade the quality of their separate alignment.

Comparisons with other multiple structural alignment programs

MUSTANG was compared with other multiple structural alignment programs such as POSA, CE-MC, MultiProt, and MALECON. Table II shows the results of comparison between MUSTANG, POSA, and CE-MC on the globin data sets shown in Table I, based on the number of positions in the common core (NCORE), and its average RMSD. A proper comparison with MultiProt was not possible as it reported alignments of subsets of structures. (MultiProt reports all local similarities of varied lengths in subsets of structures.) For Set 1 MultiProt reported a largest alignment with NCORE = 132 and RMSD = 1.20 Å in a subset containing 4 (of 5) structures. For Set 2, and 3 the corresponding values are (137, 4.40 Å) in a subset of 8 (of 9) structures, and (116, 5.20 Å) in a subset of 8 (of 11) structures, respectively.

In addition to these three globin data sets, comparisons were also made using another group containing 15 data sets as described in Ochagavia and Wodak.[64] Table III compares the values of NCORE and RMSD across various tools. Note that we use the values reported in Ye and Godzik[37] for this data set.

Table IV compares the results on the serine proteinase data sets shown in Table I. Note that we were unable to verify the performance of CE-MC on these data sets as the server does not process the uploaded coordinate files.

In summary, the comparisons between MUSTANG and other programs on the above data sets based on the common core suggest that MUSTANG shows comparable performance for sets of relatively closely related proteins, and performs better in difficult cases containing very distantly related proteins (such as Set 3 of Globins and Set 2 of Serine proteinases).

MUSTANG was compared with POSA on structures that contain structural flexibilities in which POSA performs largely better than other available programs. We treated the two data sets on which POSA's performance has been documented: calmodulin-like proteins and tRNA synthetases.

On the calmodulin-like data set containing 3 structures (PDB codes: 1jfj, chain a; 1ncx; 2sas) MUSTANG was able to detect a common core of length 129 positions. The alignment results can be found at http://www.cs.mu.oz.au/~arun/mustang/Calmodulin.html. POSA on the other hand detected a common core of 132 positions. Although the alignments were comparable, there is a difference in the superposition facilities of MUSTANG and POSA. POSA incorporates structural flexibility into its superposition; MUSTANG, in its current form does not. (Such a feature will be added to the program shortly.) Therefore Figure 6a, showing the overall superposition MUSTANG generated, is unsatisfactory. Figure 6b shows the individual superpositions of each of the two domains generated using MUSTANG's alignment of the entire proteins.

For the four tRNA synthetase structures used in POSA (PDB codes: 1adj, 1hc7, 1qf6, and 1ati), MUSTANG found 291 aligned positions as the common core of its alignment compared with 278 positions in POSA. Figure 7a shows the global superposition of the tRNA synthetases MUSTANG generated. Figure 7b show the individual superpositions of the region around the common core in domain 1 and domain 2.

The results of the calmodulin-like, and tRNA synthetase data sets suggests that MUSTANG is robust and compa-

TABLE III. Results of Comparisons on the Globin Data Set Described in Ochagavia and Wodak (2004) across Various Programs

Ochagavia and Wodak[64] Globin Data Set	NCORE	RMSD
MUSTANG	89	2.25 Å
POSA	71	2.29 Å
MALECON	59	1.73 Å
MALECON$^+$	55	1.30 Å
CE-MC	91	2.48 Å

NCORE, The number of positions in the common core; RMSD, average root-mean-square-deviation of the common core.

TABLE IV. Results of Comparisons on the Serine Proteinase Data Sets (Table I) across the Multiple Structural Alignment Programs MUSTANG, POSA, and CE-MC

Serine Proteinase	MUSTANG		POSA		CE-MC	
Data Set	NCORE	RMSD	NCORE	RMSD	NCORE	RMSD
Set 1	205	1.56 Å	199	1.43 Å	—	—
Set 2	119	2.42 Å	86	2.00 Å	—	—

NCORE, The number of positions in the common core; RMSD, average root-mean-square-deviation of the common core.

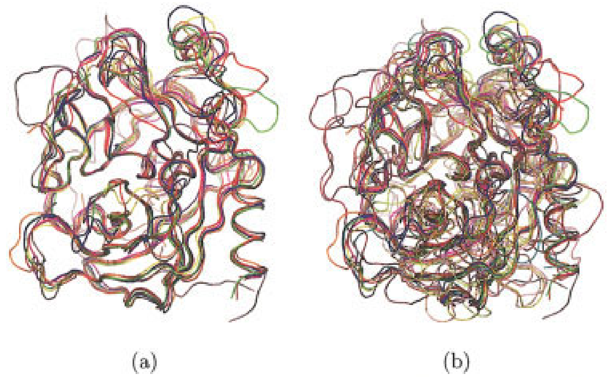


Fig. 5. Visualization of superpositions of the Serine proteinase structures in Table I (**a**) Set 1, and (**b**) Set 2 based on the MUSTANG's alignment.

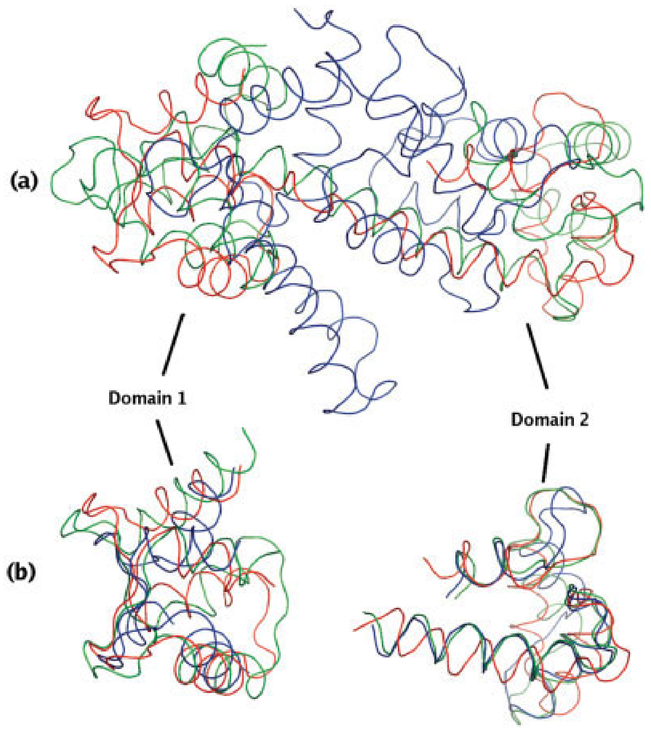


Fig. 6. Visualization of (**a**) global superposition based on MUSTANG's alignment, (**b**) individual superpositions of the common core in first and second domains, respectively, based on the correspondences in MUSTANG's alignment, for the three calmodulin-like proteins used in POSA. (1jfja, red; 1ncx, green; 2sas, blue.)

rable to POSA in the alignments it produces in cases where the data sets have structural flexibility.

MUSTANG's performance on HOMSTRAD families

HOMSTRAD is a database of protein structural alignments for homologous families.[67] Its alignments were generated using structural alignment programs such as MNYFIT, STAMP, and COMPARER followed by a manual scrutiny of individual cases. The performance of MUSTANG was compared with the database of 1033 HOMSTRAD families. The alignment results along with the

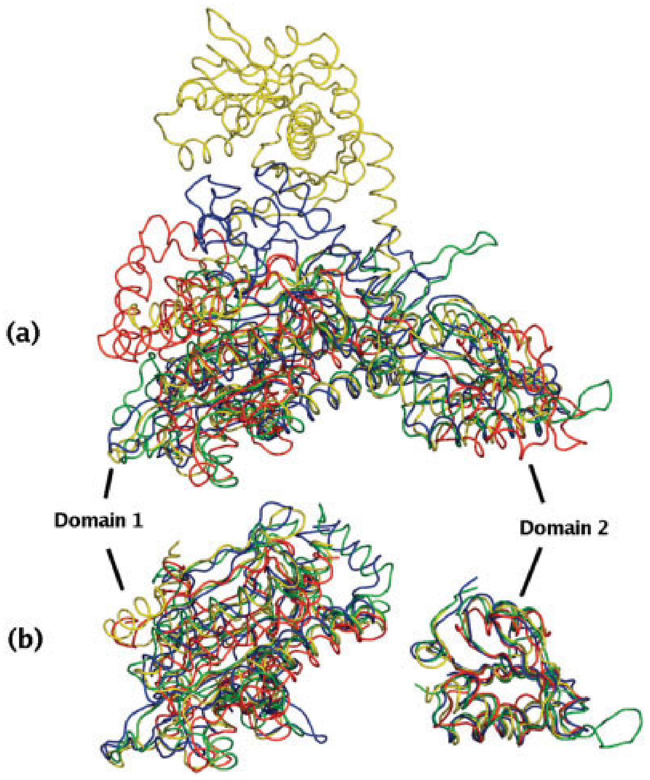


Fig. 7. Visualization of (**a**) global superposition based on the MUSTANG's alignment, (**b**) individual superpositions of the common core in domains 1 and 2 based on the correspondences in the MUSTANG's alignment, for the four tRNA synthetase structures. (1adj, red; 1ati, green; 1hc7, blue; 1qf6, yellow.)

superposed coordinates can be found at http://www.cs.mu.oz.au/~arun/ mustang/homstrad.html.

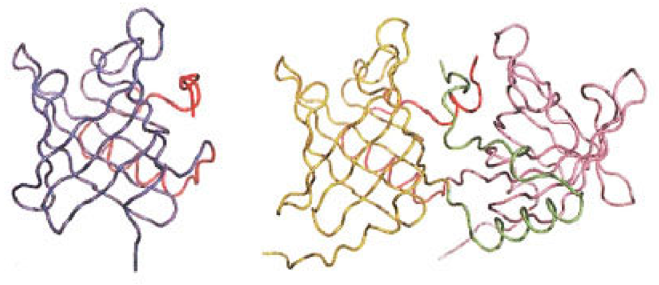
The *alignment accuracy* (*ACC*) metric was used for comparison with HOMSTRAD. *ACC* is calculated by comparing every possible imposed pairwise alignment in the query (MUSTANG) multiple alignment against the corresponding imposed pairwise alignment in the reference (HOMSTRAD) alignment. All correctly aligned residue pairs in comparison with the reference are considered as hits and those that disagree as errors. The percentage of correctly aligned residues in every pairwise alignment is then calculated and the mean of these pairwise accuracies is used as the accuracy, *ACC,* of the query multiple alignment with respect to the reference.

The mean *ACC* over all 1033 data sets is 93.4%. In general MUSTANG alignments gave smaller common cores when compared to the database alignments, and, concomitantly, slightly lower RMSDs.

Comparisons with DALI at a Pairwise Level

MUSTANG, at a pairwise level, was compared to DALI, which is widely considered to be the best of the available pairwise structural alignment tools. For this comparison we used the 633 HOMSTRAD alignment families containing two structures per family.

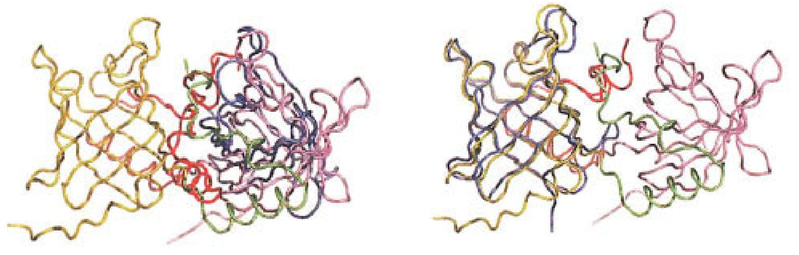
The metric *ACC* defined in the previous section was used to compare MUSTANG with HOMSTRAD, DALI with



(a) (b)

```
Sequence alignment
1a3y          ------------------------------------------------------------------------------------------
1obp      1 QEEEAEQNLSELSGPWRTVYIGSTNPEKIQENGPFRTYFRELVFDDEKGTVDFYFSVKRDGKWKNVHVKATKQDDGTYVADYEGQNVFKI  90

1a3y          ------------------------------------------------------------------P---------FELSGKWITSYIGS  15
1obp     91 VSLSRTHLVAHNINVDKHGQTTELTGLFVKLNVEDEDLEKFWKLTEDKGIDKKNVVNFLENEDHPHPEEEEAEQNLSELSGPWRTVYIGS 180

1a3y     16 SDLEKIGENAPFQVFMRSIEFDDKESKVYLNFFSKENGICEEFSLIGTKQEGNTYDVNYAGNNKFVVSYASETALIISNINVDEEGDKTI 105
1obp    181 TNPEKIQENGPFRTYFRELVFDDEKGTVDFYFSVKRDGKWKNVHVKATKQDDGTYVADYEGQNVFKIVSLSRTHLVAHNINVDKHGQTTE 270

1a3y     76 MTGLLGKGTDIEDQDLEKFKEVTRENGIPEENIVNIIERDDCPA 149
1obp    271 LTGLFVK-LNVEDEDLEKFWKLTEDKGIDKKNVVNFLENEDHPH 313

Structural alignment
1a3y      1 PF----------ELSGKWITSYIGSSDLEKIGENAPFQVFMRSIEFDDKESKVYLNFFSKENGICEEFSLIGTKQEGNTYDVNYAGNNKF  80
1obp      1 --QEEEAEQNLSELSGPWRTVYIGSTNPEKIQENGPFRTYFRELVFDDEKGTVDFYFSVKRDGKWKNVHVKATKQDDGTYVADYEGQNVF  88

1a3y     51 VVSYASETALIISNINVDEEGDKTIMTGLLGKG--------------------------------------------------------- 113
1obp     59 KIVSLSRTHLVAHNINVDKHGQTTELTGLFVKLNVEDEDLEKFWKLTEDKGIDKKNVVNFLENEDHPHPEEEEAEQNLSELSGPWRTVYI 178

1a3y          ------------------------------------------------------------------------------------------
1obp    179 GSTNPEKIQENGPFRTYFRELVFDDEKGTVDFYFSVKRDGKWKNVHVKATKQDDGTYVADYEGQNVFKIVSLSRTHLVAHNINVDKHGQT 268

1a3y    114 ---------TDIEDQDLEKFKEVTRENGIPEENIVNIIERDD---CPA 149
1obp    269 TELTGLFVKLNVEDEDLEKFWKLTEDKGIDKKNVVNFLENEDHPH--- 313
```

(c)



(d) (e)

Fig. 8. Structures of (**a**) pig odorant-binding protein (1a3y), and (**b**) cow odorant-binding protein (1obp); (**c**) sequence alignment using ClustalW, and structural alignment using MUSTANG, of the two odorant-binding proteins; (**d**) superposition based on sequence alignment, and (**e**) superposition based on structural alignment.

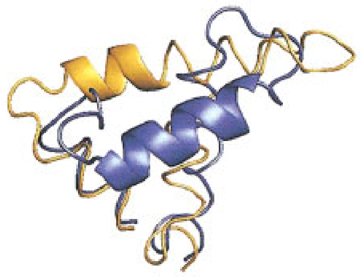
HOMSTRAD (both using the HOMSTRAD alignment as reference), and MUSTANG with DALI (using the DALI alignment as reference).

The average alignment accuracies of the above three comparisons averaged over 633 HOMSTRAD alignment families are 93.9%, 92.6%, and 87.8%, respectively. The results show that MUSTANG and DALI have comparable performances when comparing their alignments to HOMSTRAD. MUSTANG agrees with HOMSTRAD slightly better than DALI does. To compare the performance of MUSTANG and DALI specialized to a specific type of secondary structure we extracted all-α and all-β families from HOMSTRAD. MUSTANG and DALI shared 88.4% and 86.6% of aligned residues in common, averaged over all families in all-α and all-β sets, respectively. Notice that these values do not differ significantly from the average over all 633 HOMSTRAD families. This indicates that the two methods perform with comparable accuracy on regions containing these two types of secondary structures.

We also compared the alignments produced by MUSTANG and DALI for many pairs of homologous proteins, over a range of similarity. In a vast majority of the cases the alignments agree, either completely, or to within a few

```
HOMSTRAD's database alignment
1cvl    121 DFVQDVLKTDPTGLSSTVIAAFVNVFGTLVSSSHNTDQDALAALRTLTTAQTATYNRNFP 180
4lipd   120 DFVQGVLAYDPTGLSSTVIAAFVNVFGILTSSSNNTNQDALAALKTLTTAQAATYNQNYP 179
MUSTANG's alignment
1cvl    121 DFVQDVLKTDPTGLS-STVIAAFVNVFGTLVSSS----HNTDQDALAALRTLTTAQTATY 175
4lipd   120 DFVQGVLAYDPTGLSSTVIAAFVNVFGILTSSS-NNTNQ-DA---LAALKTLTTAQAATY 174
```

(a)



(b)

Fig. 9. (**a**) The region in the bacterial lipase family where the HOMSTRAD and MUSTANG alignments disagree; (**b**) a visualisation of the region in the alignment showing the variability of the structures even though the sequences in this region are almost identical. (1cvl, yellow; 4lipd, blue.)

minor differences involving small shifts in few residues. Often it is a question of precisely where to insert a necessary gap.

In a number of cases where the two methods gave significantly different alignments, we examined the alternative superpositions of the relevant substructures in detail. Many of the differences appear in ill-fitting, noisy regions where it is not objectively possible to decide which of the two alignments is better. In some cases we think that DALI's alignment of certain regions is correct and MUSTANG's is wrong; in a comparable number of other cases we think that MUSTANG's alignment is correct and DALI's is wrong; in a few cases we feel that neither program gets the alignment quite right. We did not check by hand the very large number of regions in which MUSTANG and DALI agree, although it is possible that we might not accept their common answer in all cases.

As MUSTANG and DALI share in common 87.8% of their respective alignments (on a residue-position basis averaged over 633 HOMSTRAD families), MUSTANG and DALI agree far more than they disagree.

MUSTANG is more conservative than DALI in deciding whether to align residues or to declare them unalignable. In other words, MUSTANG tends to introduce more gaps into its alignments than DALI does. Averaged over 633 HOMSTRAD families, MUSTANG inserts 4.44% more gaps than DALI. Recall that in structural alignment the notion of what can or cannot be aligned is not well defined (Irving et al.[14]). Any program can be set to align more or fewer residues by adjusting parameters that control its degree of tolerance to conformational differences. Therefore, we do not regard the overall difference in the percentage gaps introduced in the alignments as any measure of the quality of the alignments produced by MUSTANG and DALI.

In conclusion, we do not suggest that MUSTANG should replace DALI for pairwise alignment. But neither would we advise *against* using MUSTANG for pairwise alignment. If we wished to determine a structural alignment of two proteins, we would run *both* DALI and MUSTANG, compare the results, and inspect any differences by drawing pictures.

Comparisons of Sequence-Based and Structural Alignments in Cases in which They are in Conflict

The examples in this section reflect and emphasize the differences between alignments based purely on sequences, and structural alignments. They include examples of closely related sequences of proteins that contain structurally dissimilar regions.

Domain Swapping: Odorant-Binding Proteins

Pig odorant-binding protein (PDB code: 1a3y) is a monomer containing in its C-terminal segment a helix and a strand of sheet (shown in red in Fig. 8a). Cow odorant-binding protein (PDB code: 1obp) is a dimer (shown in olive and pink in Fig. 8b) in which the C-terminal helix and strand of each monomer flip over to interact with the partner (shown in green and red in Fig. 8b).[68]

Figure 8c shows the sequence alignment (generated using ClustalW[8]) and structural alignment (generated using MUSTANG) of the two odorant-binding proteins. The sequences of the two proteins are very similar. In the alignment based solely on the sequences, the monomer (1a3y) is aligned with one of the monomers of 1obp. However, the aligned C-terminal segments of the two proteins do not occupy the same positions in space (see Fig. 8d). A structural alignment that matches residues occupying equivalent positions in space generates an alignment that differs in the region of the domain swap (Fig. 8e).

Bacterial lipases

The HOMSTRAD family Bacterial Lipase (*bac_lipase*) contains two proteins with very closely related sequences:

1. Lipase (triacylglycerol lipase) from *Chromobacterium viscosum* (PDB code: 1cvl).
2. Lipase precursor (triacylglycerol lipase) from *Burkholderia cepacia* (PDB code: 4lip, chain d).

Figure 9 shows the alignments of these proteins by HOMSTRAD and MUSTANG, in the region in which they disagree, and the superposition of the structures in this region. Outside this region the HOMSTRAD and MUSTANG alignments are in total agreement with each other.

From a perspective of the evolutionary relationship of these *sequences,* HOMSTRAD has correctly aligned the residues in this region, which are almost identical. Despite the similarity of sequences, however, the structures of the region are different (see Fig. 9b). Any structural alignment method will insert some gaps in this regions, as MUSTANG does. This is not a case of such widely diverged proteins that sequence alignment will be untrustworthy, but rather a case where there is a choice between an alignment based purely on sequences, and a structural alignment. The two alignments are complementary: by comparing them, one's attention is called to the structural difference.

The implication is that, in choosing sequence or structural alignment, the user must pay attention to the purpose of the calculation and the intended use of the result. It is a case of "Be careful what you ask for, you might get it."

CONCLUSIONS

We have designed, written, and tested an algorithm for multiple structural alignment of proteins. The program is robust, fully automatic, efficient, and easy to use. The performance of MUSTANG compares favorably with published multiple structural alignment programs, and indeed is more reliable than others on hard data sets.

REFERENCES

1. Read RJ. Pushing the boundaries of molecular replacement with maximum likelihood. Acta Crystallogr 2001;D57:1373–1382.
2. Storoni LC, McCoy AJ, Read RJ. Likelihood-enhanced fast rotation functions. Acta Crystallogr 2004;D60:432–438.
3. Schwarzenbacher R, Godzik A, Grzechnik SK, Jaroszewski L. The importance of alignment accuracy for molecular replacement. Acta Crystallogr 2004;D60:1229–1236.
4. Altschul SF, Madden TL, Schaeffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 1997;25: 3389–3402.
5. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in amino acid sequence of two proteins. J Mol Biol 1970;48:443–453.
6. Smith TF, Waterman MS. Identification of common molecular subsequences. J Mol Biol 1981;147:195–197.
7. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol 1990;215:403–410.
8. Thompson JD, Higgins DG, Gibson TJ. CLUSTAL W: improving the sensitivity of progressive multiple alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res 1994;22:4673–4680.
9. Notredame C, Higgins D, Heringa J. T-Coffee: a novel method for multiple sequence alignments. J Mol Biol 2000;302:205–217.
10. Edgar RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res 2004;32:1792–1797.
11. Notredame C. Recent progress in multiple sequence alignments: a survey. Pharmacogenomics 2002;3:1–14.
12. Holm L, Sander C. Protein structure comparison by alignment of distance matrices. J Mol Biol 1993;233:123–138.
13. Chothia C, Lesk AM. The relationship between the divergence of sequence and structure in proteins. EMBO J 1986;5:823–826.
14. Irving JA, Whisstock JC, Lesk AM. Protein structural alignments and functional genomics. Proteins 2001;42:378–382.
15. Šali A, Blundell TL. Definition of general topological equivalence in protein structures. A procedure involving comparison of properties and relationships through simulated annealing and dynamic programming. J Mol Biol 1990;212:403–428.
16. Pearce M, Bottomley S, Pike R, Lesk A. Serpin conformations. In: Lomas D, Silverman G, editors. Molecular and cellular aspects of the serpinopathies and disorders in serpin activity. Singapore; World Scientific:2006.
17. Levine M, Stuart D, Williams J. A method for systematic comparison of the three-dimensional structures of proteins and some results. Acta Crystallogr 1984;A40:600–610.
18. Karpen ME, de Haseth PL, Neet KE. Comparing short protein substructures by a method based on backbone torsion angles. Proteins 1989;6:155–167.
19. Lesk AM. Computational molecular biology. In: Kent A, Williams JG, editors. Encyclopedia of computer science and technology, vol. 31. New York: Marcel Dekker; 1994. p 101–165.
20. Caprara A, Carr R, Istrail S, Lancia G, Walenz B. 1001 optimal pdb structure alignments: integer programming methods for finding the maximum contact map overlap. J Comput Biol 2004;11: 27–52.
21. Lesk AM, Chothia C. How different amino acid sequences determine similar protein structures: I. The structure and evolutionary dynamics of the globins. J Mol Biol 1980;136:225–270.
22. Lesk AM. 11th Lipari international summer school in computational biology. 1999.
23. McLachlan AD. A mathematical procedure for superimposing atomic coordinates of proteins. Acta Crystallogr 1972;A28:656.
24. Kabsch W. A solution for the best rotation to relate two sets of vectors. Acta Crystallogr 1976;A32:922–923.
25. Kearsley SK. On the orthogonal transformation used for structural comparisons. Acta Crystallogr 1989;A45:208–210.
26. Rustici M, Lesk AM. Three-dimensional searching for recurrent structural motifs in databases of protein structures. J Comput Biol 1994;1:121–132.
27. Orengo CA, Taylor WR. A rapid method for protein structure alignment. J Theor Biol 1990;147:517–551.
28. Gerstein M, Levitt M. Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of protein structures. In: Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology. Menlo Park, CA; AAAI Press; 1996. p 59–67.
29. Nussinov R, Wolfson HJ. Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. Proc Natl Acad Sci 1991;88:10495–10499.
30. Koehl P. Protein structure similarities. Curr Opin Struct Biol 2001;11:348–353.
31. Just W. Computational complexity of multiple sequence alignment with SP-score. J Comput Biol 2001;8:615–623.
32. Wang L, Jiang T. On the complexity of multiple sequence alignment. J Comput Biol 1994;1:337–348.
33. Goldman D, Papadimitriou CH, Istrail S. Algorithmic aspects of protein structure similarity. In: FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science. Washington, DC: IEEE Computer Society; US 1999. p 512.
34. Guda C, Scheeff ED, Bourne PE, Shindyalov IN. A new algorithm for the alignment of multiple protein structures using Monte Carlo optimization. In: Pacific Symposium on Biocomputing. Singapore; World Scientific: 2001. p 275–286.
35. Guda C, Lu S, Scheeff ED, Bourne PE, Shindyalov IN. CE-MC: a multiple protein structure alignment server. Nucleic Acids Res 2004;32:100–103.
36. Shindyalov IN, Bourne PE. Protein structure alignment by incre-

mental combinatorial extension (CE) of the optimal path. Protein Eng 1998;11:739–747.
37. Ye Y, Godzik A. Multiple flexible structure alignment using partial order graphs. Bioinformatics 2005;21:2362–2369.
38. Lee C, Grasso C, Sharlow MF. Multiple sequence alignment using partial order graphs. Bioinformatics 2002;18:452–464.
39. Ye Y, Godzik A. Flexible structure alignments by chaining aligned fragment pairs allowing twists. Bioinformatics 2003;19:II246–II255.
40. Ye J, Janardan R. Approximate multiple protein structure alignment using the sum-of-pairs distance. J Comput Biol 2004;11:986–1000.
41. Gusfield D. Efficient methods for multiple sequence alignment with guaranteed error bounds. Bull Math Biol 1993;55:141–154.
42. Shatsky M, Nussinov R, Wolfson H. MULTIPROT— a multiple protein structural alignment algorithm. In: Guigo R, Gusfield D, editors. Workshop on algorithms in bioinformatics, Vol. 2452 Berlin: Springer-Verlag; 2002. p 235–250.
43. Dror O, Benyamini H, Nussinov R, Wolfson HJ. Multiple structural alignment by secondary structures: algorithm and applications. Protein Sci 2003;12:2492–2507.
44. Leibowitz N, Nussinov R, Wolfson HJ. MUSTA — a general, efficient, automated method for multiple structure alignment and detection of common motifs: application to proteins. J Comput Biol 2001;8:93–121.
45. Russell R, Barton G. Multiple protein sequence alignment from tertiary structure comparison: assignment of global and residue confidence levels. Proteins 1992;14:309–323.
46. Sutcliffe MJ, Haneef I, Carney D, Blundell TL. Knowledge based modelling of homologous proteins. Part I: three-dimensional frameworks derived from the simultaneous superposition of multiple structures. Protein Eng. 1987;1:377–384.
47. Lesk AM. Three-dimensional pattern matching in protein structure analysis. In: Galil Z, Ukkonen E, editors. Combinatorial pattern matching, Vol. 937 Berlin: Springer-Verlag; 1995. p 248–260.
48. Lesk AM, Fordham WD. Conservation and variability in the structures of serine proteinases of the chymotrypsin family. J Mol Biol 1996;258:501–537.
49. Taylor WR, Flores TP, Orengo CA. Multiple protein structure alignment. Protein Sci 1994;3:1858–1870.
50. Taylor WR, Orengo CA. Protein structure alignment. J Mol Biol 1989;208:1–22.
51. Ding DF, Qian J, Feng ZK. A differential geometric treatment of protein structure comparison. Bull Math Biol 1994;56:923–943.
52. May ACW, Johnson MS. Improved genetic algorithm-based protein structure comparisons: pairwise and multiple superpositions. Protein Eng 1995;8:873–882.
53. Eidhammer I, Jonassen I, Taylor WR. Protein bioinformatics: an algorithmic approach to sequence and structure analysis. Chichester, U.K.; Wiley: 2004.
54. Shapiro A, Botha JD, Pastore A, Lesk AM. A method for multiple superposition of structures. Acta Crystallogr 1992;A48:11–14.
55. Diamond R. On the multiple simultaneous superposition of molecular structures by rigid body transformations. Protein Sci 1992;1:1279–1287.
56. Maiorov VN, Crippen GM. Contact potential recognizes the correct folding of globular proteins. J Mol Biol 1992;227:876–888.
57. Crippen GM. Recognizing protein folds by cluster distance geometry. Proteins 2005;60:82–89.
58. Havel TF, Kuntz ID, Crippen GM. The theory and practice of distance geometry. Bull Math Biol 1983;45:665–720.
59. Richards FM, Kundrot CE. Identification of structural motifs from protein coordinate data: secondary structure and first-level supersecondary structure. Proteins 1988;3:71–84.
60. Morgenstern B. Dialign2: improvement of the segment-to-segment approach to multiple sequence alignment. Bioinformatics 1999;15:211–218.
61. Neuwald AF, Liu JS, Lipman DJ, Lawrence CB. Extracting protein alignment models from the sequence database. Nucleic Acids Res 1997;25:1665–1677.
62. Feng DF, Doolittle RF. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J Mol Evol 1987;25:351–360.
63. Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 1987;4:406–425.
64. Ochagavia ME, Wodak S. Progressive combinatorial algorithm for multiple structural alignments: applications to distantly related proteins. Proteins 2004;55:436–454.
65. Lesk AM. The evolution of the globins: we thought we understood it. In: Bastolla U, Porto M, Roman HE, Vendruscolo M, editors. Structural approaches to sequence evolution. Berlin: Springer-Verlag; 2005.
66. Vuletich DA, Lecomte JT, Lesk AM. Structural divergence and distant relationships in proteins: evolution of the globins. Curr Opin Struct Biol 2005;15:290–301.
67. Mizuguchi K, Deane CM, Blundell TL, Overington JP. HOMSTRAD: a database of protein structure alignments for homologous families. Protein Sci 1998;7:2469–2471.
68. Ramoni R, Vincent F, Ashcroft AE, Accornero P, Grolli S, Valencia C, Tegoni M, Cambillau C. Control of domain swapping in bovine odorant-binding protein. Biochem J 2002;365:739–748.