PROTEINS: Structure, Function, and Genetics 43:235–245 (2001)

Automated Multiple Structure Alignment and Detection of a Common Substructural Motif

Nathaniel Leibowitz,[1] Zipora Y. Fligelman,[1] Ruth Nussinov,[2,3]* and Haim J. Wolfson[1]

[1]*School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel*

[2]*Sackler Institute of Molecular Medicine, Department of Human Genetics and Molecular Medicine, Sackler Faculty of Medicine, Tel Aviv University, Tel Aviv, Israel*

[3]*Intramural Research Support Program, SAIC, Laboratory of Experimental and Computational Biology, National Cancer Institute, FCRDC, Frederick, Maryland*

ABSTRACT **While a number of approaches have been geared toward multiple sequence alignments, to date there have been very few approaches to multiple structure alignment and detection of a recurring substructural motif. Among these, none performs both multiple structure comparison and motif detection simultaneously. Further, none considers all structures at the same time, rather than initiating from pairwise molecular comparisons. We present such a multiple structural alignment algorithm. Given an ensemble of protein structures, the algorithm automatically finds the largest common substructure (core) of C_α atoms that appears in all the molecules in the ensemble. The detection of the core and the structural alignment are done simultaneously. Additional structural alignments also are obtained and are ranked by the sizes of the substructural motifs, which are present in the entire ensemble. The method is based on the geometric hashing paradigm. As in our previous structural comparison algorithms, it compares the structures in an amino acid sequence order-independent way, and hence the resulting alignment is unaffected by insertions, deletions and protein chain directionality. As such, it can be applied to protein surfaces, protein–protein interfaces and protein cores to find the optimally, and suboptimally spatially recurring substructural motifs. There is no predefinition of the motif. We describe the algorithm, demonstrating its efficiency. In particular, we present a range of results for several protein ensembles, with different folds and belonging to the same, or to different, families. Since the algorithm treats molecules as collections of points in three-dimensional space, it can also be applied to other molecules, such as RNA, or drugs. Proteins 2001;43:235–245.**

Key words: multiple structural alignment; geometric hashing; invariants; structural core; multiple structural comparison

INTRODUCTION

Clearly, possessing a tool that enables carrying out an automated multiple structure comparison of an ensemble of structures, and doing it at high speed, is highly advantageous. This is particularly the case if the technique permits detection of spatially conserved substructural motifs, independent of the order of the residues on the chains. Under such circumstances, in principle, the motifs that are detected may contain particularly useful information. First, residues that are conserved across different species, illustrating which are the more essential residues for stability, catalysis, or binding. Second, conserved motifs may constitute good candidates for threading and design. Third, the motifs may be particularly useful in homology modeling. Further, because the method considers residues as a collection of points in three-dimensional (3-D) space, using only the coordinates of the C_α atoms without taking account of the order in which they are arranged on the protein backbone, it may substitute the coordinates of the C_α atoms by those of surface atoms, and seek motifs on protein surfaces. Fourth, the atomic coordinates may be substituted by coordinates of points describing protein surfaces, such as critical points (e.g., Lin et al.[1,2]). Fifth, since the points can be those of any atoms, the method can also be used in searches for pharmacophores in drugs, or in comparisons of RNA structures. Here, however, we confine ourselves to multiple structural comparisons of proteins, using their C_α atoms.

While the advantages of having tools for multiple structure alignment and detection of recurring substructural motifs are obvious, to date most efforts have focused on multiple sequence comparisons. In principle, multiple structure comparisons with the only input being the atomic coordinates of the entire structures, without a predefinition of the motif, and disregarding the order of the residues on the chain, is a very difficult problem. To carry out such an endeavor, one needs to multiply align

Grant sponsor: Magnet grant; Grant sponsor: Ministry of Science grant; Grant sponsor: Center of Excellence in Geometric Computing and Its Applications/Israel Science Foundation/Israel Academy of Sciences; Grant sponsor: Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University; Grant sponsor: National Cancer Institute/National Institutes of Health; Grant number: NO1-CO-56000.

*Correspondence to: Ruth Nussinov, Intramural Research Support Program, SAIC, Laboratory of Experimental and Computational Biology, NCI, FCRDC, Bldg 469, Room 151, Frederick, MD 21702. E-mail: ruthn@ncifcrf.gov

Received 19 June 2000; Accepted 5 January 2001

every substructure in each of the molecules in the ensemble, in all rotations and in all translations. We illustrate how, using the geometric hashing algorithm,[3] such a task can be carried out.

To date, only few methods address the problems of multiple structural alignment and the detection of recurring substructural motifs. These methods can be roughly classified into two categories: The first group includes algorithms, which already initiate from a multiple sequence alignment. This initial alignment may be either of the sequences or of the secondary structures. The output of such methods is a detected preserved core, or a refinement of a previously detected core. The second group includes algorithms, which address the structural alignment problem itself. Algorithms that belong to this group usually work by performing an iterative series of pairwise structural alignments to finally obtain the multiple alignment. A representative example of the first group is the structurally invariant method for detection of a substructural core by Gerstein and Altmann.[4] This algorithm accepts an aligned set of structures. It selects the spatially recurring position found in all the compared structures. This position serves as an initial structural core. Through an iterative procedure it computes an average structure, removing the structurally most variable positions from the core. This procedure is applied iteratively, until a certain cutoff is reached. The average structure of an ensemble is similarly obtained through an iterative procedure. In each iteration $O(N^2)$ (where N is the number of structures) pairwise best root-mean-square deviation (RMSD) fits[5] are performed between the structures in the ensemble. The algorithm of Gelfand et al.[6] also initiates from a set of aligned structures. Next, for each pair of positions in the aligned ensemble, the average distance between these positions and the dispersion of this distance across the structures is computed. A recurring subset of these positions is sought, with the stipulated requirement that in each position the average dispersion relative to the other recurring core positions be low. Since this first group of algorithms initiates from a multiple alignment (e.g., of sequences), they circumvent the need for a priori solving the difficult multiple structural alignment problem. Instead, they focus on the refinement of an optimally recurring core for a given alignment.

Very few algorithms belong to the second group, addressing the multiple structural alignment problem itself. Among these is the SSAPm method of Orengo and Taylor.[7] This method works by performing all pairwise alignments of the structures using double dynamic programming. The best fitting pair serves as a seed for the multiple alignment. At each stage the average consensus structure and the variance at each position are computed, with this information being used in subsequent stages. The best-fitting structure is joined to the consensus in an iterative manner, with the consensus structure and positional variations being recalculated until all the structures are aligned. The positional variance is employed to extract weights, both for the comparisons and for the assessment of the positional conservation. Gerstein and Levitt[8] also perform all pairwise structural alignments using their iterative dynamic programming structural alignment method. They select a structure, which is most similar to all other structures in the least squares sense. All other structures are aligned to this most similar structure. However, as these investigators note,[8] such a procedure does not automatically ensure geometric consistency at any given position across all the structures. Hence, the authors suggest double-checking such positions with the automatically generated pairwise alignments.

The algorithm presented is fully automated. It solves the structural alignment and the detection of the substructural (core) motif simultaneously. A relatively small number of sequence-order-independent positions (in this implementation, we use five) appearing in all structures under consideration serve as initial seed. A collection of such seeds is detected by geometric hashing of their invariants. The seeds induce pairwise transformations between the corresponding protein structures. Initial seeds are subsequently merged into larger substructures with the condition that all the induced pairwise rigid transformations are simultaneously satisfied. In the following discussion, we describe this algorithm and present a range of results, illustrating its performance. A preliminary version of the algorithm was given at the Intelligent Systems for Molecular Biology (ISMB) conference.[15] A full version of the algorithm is given in ref. 16.

THE MULTIPLE STRUCTURE ALIGNMENT ALGORITHM

An overview of the multiple structure alignment (MUSTA) algorithm is presented in the flowchart in Figure 1. The input to this algorithm consists of an ensemble of N molecules. Each of the molecules is represented by the 3-D coordinates of its C_α atoms. Our goal is to detect the largest common geometric configuration of these C_α atoms, which appears in all the molecules belonging to the ensemble. This recurring geometric configuration constitutes the geometric core of the ensemble. By definition, all these geometric configurations occurring in the molecules constituting the ensemble are congruent up to a small error factor. Thus, for each pair of molecules M_I, M_J the C_α atoms of the two cores are aligned, and a rigid transformation (3-D rotation and translation) T_{IJ}, which superimposes these atoms with a small RMSD. In particular, in our algorithm, the core and the induced alignments are solved simultaneously. The structural alignment of the geometric cores automatically induces structural alignments of the full molecules. A convenient way to represent this simultaneous N-structure alignment is as follows: Let us pick one of the structures (e.g., the first). This is our reference structure. We compute all N-1 rigid transformations between this first structure and the remaining structures. The resulting N-1 dimensional vector of rigid transformations uniquely defines the multiple alignment, which superimposes the respective congruent cores. Below, we call these other N-1 structures source structures and the resulting N-1 dimensional transformation vector a multidimensional transformation.

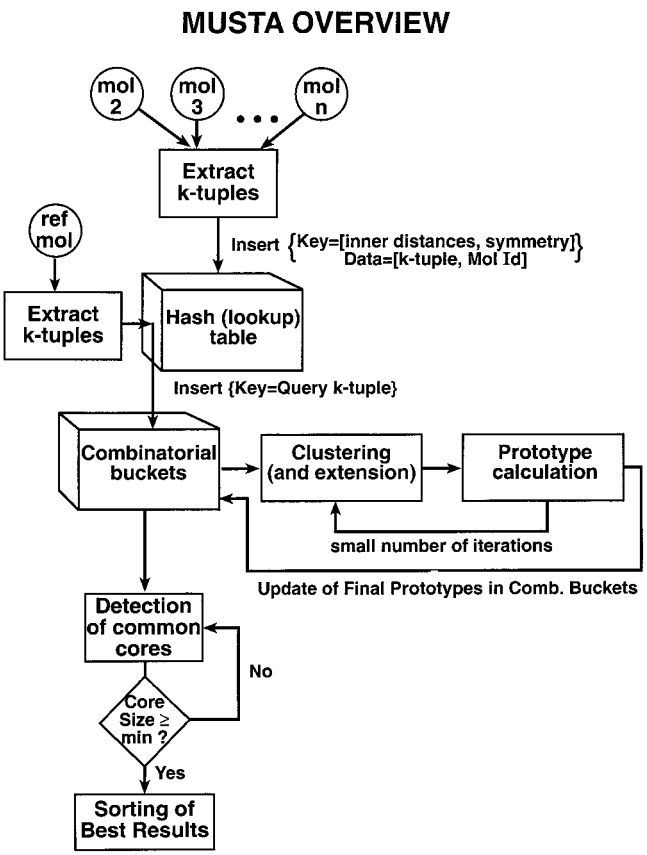

Fig. 1. Overview of MUSTA, the automated multiple structure alignment algorithm. The algorithm initiates by extracting *k*-tuples from the source molecules and inserting them into a hash table using their inner distances and their symmetries as a key to the table. The source molecules (Mol 2, Mol 3,..., Mol *n*) are depicted at the top of the schematic chart. Mol 1 will be our reference molecule. In the table, we store the *k*-tuple, and the source molecule it is associated with. Then we extract, in a similar manner, the *k*-tuples from the reference molecule (noted as ref. mo). Here, too, we use the inner-distances and symmetry as key, now to query the hash table. The *k*-tuples, which match this query, are inserted into combinatorial buckets. The address to the specific combinatorial bucket table is the query *k*-tuple. The combinatorial buckets containing *k*-tuples from all the source molecules, i.e., the fully preserved buckets, are passed on to the clustering stage. Clustering is performed in a small number of iterations, where at the end of each clustering step a representative prototype for the cluster is calculated, and considered as input to the next clustering iteration. The clustering stage is performed in a pairwise manner, between the reference molecule and each source molecule. At each iteration we extend the match list the cluster is based upon. At the end of the clustering process, the final prototypes are updated in the combinatorial buckets replacing the *k*-tuples from which they arose. Next, the combinatorial buckets are searched. We seek to detect in these buckets common cores shared by all molecules. If a core is large enough (more than a predefined threshold) we report it. We sort the best results according to the core match size.

In this work, our goal is to obtain the largest geometric core. Nevertheless, a biologically interesting result might not necessarily be the largest recurring substructural core motif. The method we present allows detection of smaller substructures as well, as long as they are larger than some predefined threshold size. The algorithm consists of three major stages:

1. Detection of seed matches and of candidate multidimensional transformations
2. Clustering of the transformations in each of the multidimensional transformation components and extension of the seed matches corresponding to the cluster prototypes
3. Computation of the highest scoring multidimensional transformations. These induce the largest cores. Lower-ranked transformations can be obtained as well, which lead to the smaller substructural motifs

Each of these three stages is described in the following discussion.

First Stage: Detection of Seed Matches and of Candidate Multidimensional Transformations

Our goal is to detect *k*-tuples of atoms (i.e., points) whose geometric configuration appears in all N molecules. The *k*-tuple should be large enough ($k \geq 3$) to determine a rigid transformation between a pair of molecules. The size of *k* is a tradeoff between the complexity of this first stage of the algorithm and the discriminatory power of a *k*-tuple structure. The larger the *k*-tuple, the higher the complexity. However, when *k* is larger, the number of spurious matches is smaller. We use $k = 5$, which appears to be a reasonable compromise between the two considerations. For a recurring substructural motif of size larger than *k*, congruent copies of its *k*-tuples appear in all the molecules. Hence, we can pick at least one set of N congruent *k*-tuples from all the molecules. Below, we call such a set a fully preserved *k*-tuple. Clearly, large conserved substructures imply many fully preserved *k*-tuples. Each fully preserved *k*-tuple defines a multidimensional transformation. At this stage of the algorithm, we detect the *k*-tuples recurring in all the molecules in the ensemble, and compute the multidimensional transformations superimposing them. For each such multidimensional transformation, we store the (multidimensional) core consisting of all the *k*-tuples, which induce this transformation.

In practice, in order to reduce the run-time complexity, and to a priori filter out obvious (uninteresting) results, such as fragments of α-helices, only *k*-tuples that satisfy predefined constraints are kept. Specifically, for each C_α-atom, we extract k-1 nonconsecutive C_α atoms within a spherical shell centered at that atom (the two radii defining the shell are user-dependent parameters). The C_α atoms constituting the *k*-tuple are ordered internally in both increasing and decreasing order of the sequence. In the event that we wish to ignore this local sequence information, or to compare points describing molecular surfaces, we may order the *k*-tuple atoms according to geometric criteria, e.g., the lengths of the distances between the atoms.

The fully preserved *k*-tuples are computed efficiently, using the geometric hashing algorithm.[9] Each *k*-tuple is represented by a parameter vector. This vector is invariant to 3-D rotation and translation and permits reconstruction of the *k*-tuple. Congruent *k*-tuples are then represented by identical parameter vectors. The *k*-tuples are inserted into a hash table addressed by these invariants. This automatically implies that congruent *k*-tuples reside at the same address.

Let us consider an ordered 5-tuple as a closed polygon in space. Clearly, the shape of this polygon is uniquely determined by nine parameters consisting of the ordered set of the lengths of its four edges, the three angles between them, and the two torsion angles between consecutive triangles. Thus, these nine parameters can serve as the required invariants. However, in order to enhance numerical stability, our set of invariant parameters is based only on inter-point distances. Namely, the invariant parameter vector is defined by 9 (out of the 10) inner distances. Yet, these distances do not define uniquely the shape of the 5-tuple. Consider the plane defined by the first three points. The fourth and fifth points can be above or below this plane without violating the distance constraints. This defines four possible "symmetries" for a given 5-tuple with the same 9 inter-point distances. Thus, we construct four 9-dimensional hash tables, one for each symmetry, and insert the source molecule information as follows. Each source molecule is processed separately. For each k-tuple in a spherical shell within each of these molecules, we compute the 9-inner distances invariant, and find the symmetry of this tuple. Using the 9-distance invariant vector as an address, the k-tuple is inserted into the appropriate hash table. This step can be considered as a preprocessing step if the same data set is used in several multiple structure comparison runs. In the next step, we scan the reference molecule. For each k-tuple (within a spherical shell) in this reference molecule, we compute the nine inner distances invariant, find the tuple's symmetry, and use the 9-inner-distances-invariant as an address to the appropriate hash table. If an entry similar up to a predefined threshold is found, we store the results of the query in a bucket associated with this reference molecule k-tuple.

At the end of this process, each of the buckets we have formed is associated with a k-tuple from the reference molecule. The bucket contains k-tuples from source molecules, which are congruent with this reference molecule k-tuple. If the k-tuple is fully preserved in the ensemble, its bucket will contain representatives from all the source molecules. In the implementation presented in this discussion, all buckets in which at least one source molecule is not represented, are ignored. This substantially reduces the practical complexity of the problem, as it eliminates the k-tuples that are not fully preserved.

Each of the remaining buckets, which contains k-tuples from all the source molecules, and from the reference molecule, determines at least one multidimensional transformation. One would like to compute efficiently those multidimensional transformations that induce large congruent cores. Let m_i, $i = 2\ldots, N$ be the number of k-tuples belonging to the ith source molecule in a bucket under consideration. Theoretically,

$$\prod_{i=2}^{N} m_i$$

different multidimensional transformations can be generated for the reference k-tuple, which defines this bucket, which we nickname a combinatorial bucket (Fig. 2). Our goal is to explore this combinatorial space without actually enumerating over all of the cases. In particular, we shall do it avoiding the cases that produce short alignments and by clustering similar enough transformations.

Clearly, each multidimensional transformation has a seed (geometrical) core, associated with it. This core consists of the aligned k-tuples, which induce the multidimensional transformation. We next cluster these initial transformations, and use the prototype transformations of each cluster to extend the associated seed cores. This is done in the following stages of the algorithm.

Second Stage: Clustering of the Multidimensional Transformations and Extension of the Seed Matches

In this stage, we extend the seed cores by clustering the multidimensional transformations. Each component of a multidimensional transformation is a rigid transformation between the reference molecule and a source molecule. The clustering procedure is applied separately to each such component; hence, at each stage, we cluster pairwise rigid transformations between the reference and one of the source molecules. Since each rigid transformation is uniquely defined by its three rotational and three translational parameters, a straightforward clustering approach is to cluster these six parameter vectors. A natural distance between such vectors is a weighted 6-D Euclidean distance with different weights assigned to the translational and rotational parameters. Such an approach proves problematic for several reasons. First, it is not obvious which relative weights should be assigned to the rotation versus the translation parameters. Second, the numerical stability might present problems. For example, a small change in the rotation angles would result in larger effects on the transformation of 3-D points, which are farther from the origin. And third, the differences in the rotation parameters can be somewhat compensated by counter changes in the translation parameters. As a result, we adopt a different distance definition between a pair of transformations. Since we are not interested in the transformation parameters per se but, in the transformation's action on given sets of 3-D points, our notion of distance between a pair of transformations is represented by the number of reference molecule points, which are mapped to different locations. A precise definition of this distance is given in the following paragraph. In principle, this should ensure that transformations belonging to the same cluster map relevant 3-D points to (almost) identical locations.

We consider separately each component in all multidimensional transformations accumulated so far. Each component defines a rigid transformation between the reference molecule and the source molecule corresponding to that specific component. We cluster this set of transformations, and compute a prototype transformation for each cluster. First, a distance is defined between a pair of transformations. Recall that both transformations map the reference molecule to the same source molecule. Each such transformation can be represented by its match list

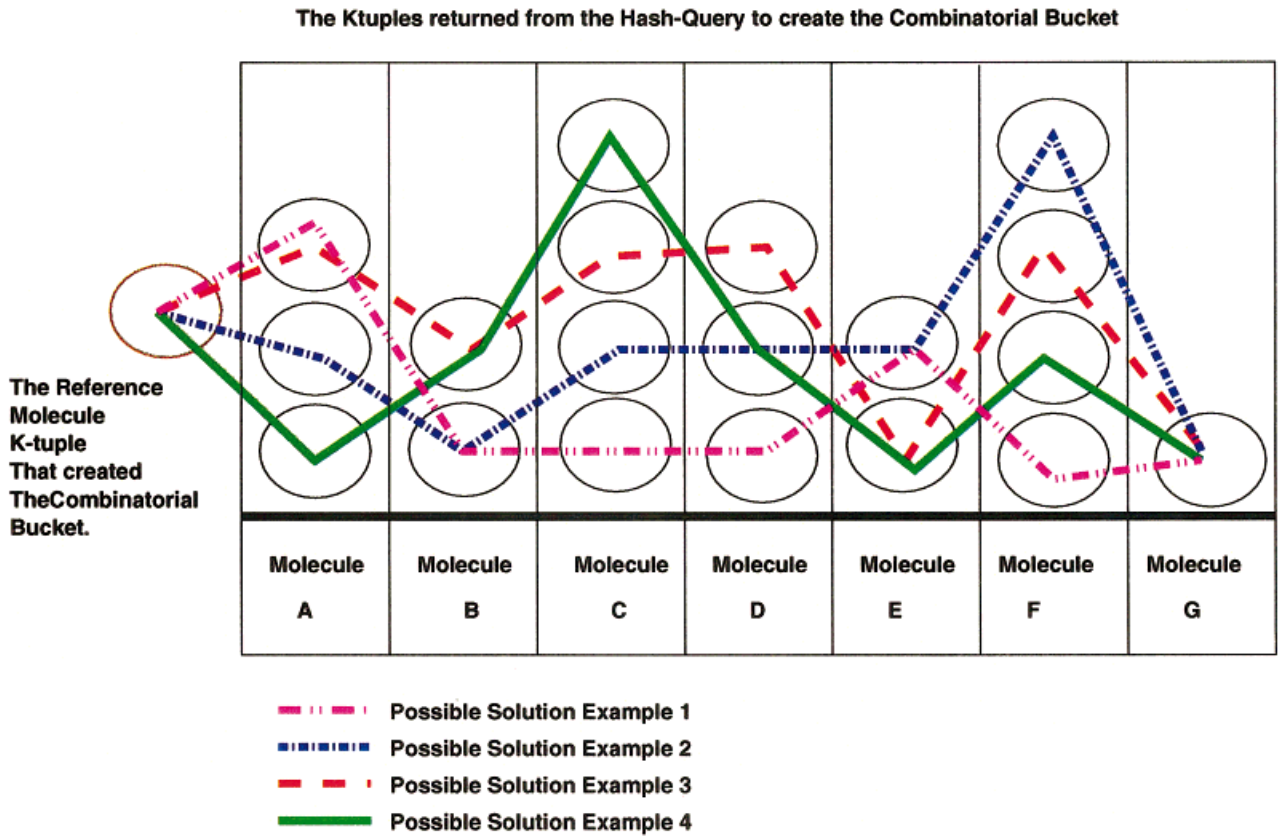

Fig. 2. Illustration of how a combinatorial bucket defines a possible solution, which is a multidimensional transformation. The reference *k*-tuple that generates the bucket (left-hand side) is shown as a brown circle. The *k*-tuples retrieved from the Hash Table, sharing the same invariants (up to a predefined error), are drawn as black circles. Each circle denotes a *k*-tuple with a different identity; i.e., it originates from a different combination of C_α atoms. The *k*-tuples are ordered in columns. Each column represents a different source molecule, from which the *k*-tuple was derived. These are denoted by the symbolic name of the molecules given below the thick line. The paths appearing in green, red, purple, and blue denote several possible multidimensional solutions between all molecules. Note that although molecule *G* has only one representative *k*-tuple sharing these invariants, this is not always the case. We only require that each molecule has at least one representative. All possible paths through the circles representing the *k*-tuple are possible solutions induced by such a *k*-tuple. The flowchart in Fig. 1 illustrates how the combinatorial buckets are searched, and used to generate the final, optimal (largest) match.

corresponding to the congruent *k*-tuples, i.e., the list of atom pairs, with one atom from the reference, and its pairing one from the source molecule. In the first iteration, each transformation has a match list of size *k*. At every iteration, we create a list, which is the union of the match list pairs of all the participating transformations. Then, each transformation is applied (in turn) to the C_α atoms of the reference molecule in this entire union match list. For each pair from the union match list we check whether the reference molecule C_α atom is mapped in the vicinity of its corresponding source molecule atom. If the answer is positive, the pair is marked as "consistent" with the examined transformation. The list of pairs consistent with the transformation is defined as the "transformation mask." The distance between two transformations is then defined as the number of consistent pairs that appears in one of the lists, but not in the other. This distance definition is obviously symmetric. The smaller this number, the closer the distance, and the more similar the transformations. The distance definition complies with an intuitive notion, namely, that similar transformations should have similar match lists. To cluster, we use a technique that iteratively clusters proximate transformations and replaces each cluster by its representative prototype. Such prototype transformation is defined by a match list, which is the union of the match lists of all the transformations belonging to the represented cluster. It is calculated as the rigid transformation, which yields the best superposition of this union match list in the least-squares (i.e., RMSD) sense.[5] Since prototypes emerge only from seed match lists that are fully preserved in all the molecules in the ensemble, this method of generating prototypes is suitable for the multiple aspect of the problem. Clustering of the transformations is carried out iteratively, namely, the input for a new clustering iteration are the prototype transformations with their associated match lists generated by the previous iteration. The iterations are continued until no further merging of match lists consistent with a given transformation can be obtained.

To further clarify the procedure, there are three additional points to note. First, each combinatorial bucket contains *k*-tuples, appearing in each of the source mol-

TABLE I. The Structural Classification by SCOP of the Calcium Binding Structural Comparison

PDB code	No. of C_α	Family	Protein	Species
4cpv	108	Parvalbumin	Parvalbumin	Carp
2scpA	174	Calmodulin-like	Sarcoplasmic calcium-binding protein	Sandworm
2sas	185	Calmodulin-like	Sarcoplasmic calcium-binding protein	Amphioxus
1top	162	Calmodulin-like	Troponin C	Chicken
1scmB	138	Calmodulin-like	Myosin essential chain	Bay scallop
3icb	75	Calbindin D9K	Calbindin D9K	Bovine

ecules (Fig. 2), congruent to a specific k-tuple of the reference molecule. Hence, the match lists, which are candidates for merging in the clustering procedure, contain different reference (molecule) k-tuples, originating from different combinatorial buckets. Thus, obviously, the creation of the union match lists is cross-combinatorial buckets, i.e., with more than one combinatorial bucket contributing to the union. Second, in the clustering procedure only one source molecule is considered at a time. Third, on the technical side, to make the clustering procedure more efficient, a rough preclustering procedure is applied, which clusters the transformations according to their translation distance.

Third Stage: Computation of the Highest Scoring Multidimensional Transformations

The prototype between the molecule and each of the source molecules serves as input to the third stage of the algorithm. In principle, the combinatorial complexity of finding the optimal multidimensional transformation superimposing the largest number of C_α-atom pairs from all structures in the ensemble, is of the order of

$$\prod_{i=2}^{N} P_i$$

where P_i is the size of the prototype set (i.e., the number of prototypes created by the clustering stage) for the ith source molecule. However, not all the prototype combinations will produce multidimensional transformations. Hence, we limit the search to those prototypes, which appear simultaneously in at least one of the combinatorial buckets. These necessarily share at least one fully preserved k-tuple, i.e., the one defined by the reference molecule for that combinatorial bucket. This reduces the complexity to that of the combinatorial buckets, which is a fraction of the one cited above. In addition, we remove those prototypes whose match list is below the required minimal core size threshold (i.e., they are not large enough), which again contributes to the reduction in the number of fully preserved combinatorial buckets. The complexity of the final solution space explored is therefore bounded from above by

$$\sum_{j \in \text{RemainingFullComb.Buckets}} \prod_{i=2}^{N} m_i^{*j}$$

where m_i^{*j} is the number of prototype transformations from the reference molecule to the ith source molecule, which have remained in the jth combinatorial bucket.

The MUSTA algorithm concludes by inspecting all the combinations which are present in the combinatorial buckets. For each combination, the intersection of the N-1 match lists is computed. If the intersection is larger than a predetermined minimal number of atoms stipulated for a geometric core, this combination is stored as a potential multiple alignment solution. Finally, the solutions are ranked by their core size.

RESULTS AND DISCUSSION

We have experimented extensively with the algorithm, producing a range of results. Some of these are presented below. The list of parameters is presented in the Appendix. This list is used for the results presented. In particular, we note that we have alternated the identity of the reference molecule, obtaining identical results.

We have carried out numerous multiple structure comparisons, using the SCOP[10] database. This database presents a useful hierarchical classification of the Protein Data Bank (PDB).[11] Classes are defined at the upper level. Each class has several folds. A fold contains a number of superfamilies. Each superfamily consists of several families, which in turn are composed of proteins that are further classified according to the species from which they are derived. As one descends down the SCOP tree, the structural similarity among the proteins increases. We have performed multiple structural alignments of protein ensembles belonging to various levels of the SCOP tree. The sets of proteins used in this work are given in Tables I–IV.

The first set of multiple structure comparisons has been carried out on molecules belonging to the serpin fold. The serpin fold consists of a single superfamily with a single family, which, in turn contains proteins from different species. We have considered α-antitrypsin, elastase inhibitor, ovalbumin, antichymotrypsin α-I, antitrypsin, and antithrombin from human, horse, bovine and hen. The PDB codes for the 13 molecules we have compared are 7apiA, 8apiA, 1hleA, 1ovaA, 2achA, 9apiA, 1psi, 1atu, 1ktc, 1athA, 1attA, 1antI, and 2antI. The number of C_α atoms in these molecules ranges from 337 to 420. Table V presents the results of four different runs, increasing the number of molecules, from 6, to 9, to 11, and to 13 molecules. Table V refers to these, respectively, as serp6, serp9, serp11, and serp13. As expected, the results show high similarity among the molecules, and a decrease in the

TABLE II. The TIM-Barrels' Structural Classification by SCOP

PDB code	No. of C_α	Superfamily	Family	Protein
4enl[a]	436	Enolase C-terminal domain-like	Enolase	Enolase
2mnr[a]	357	Enolase C-terminal domain-like	D-Glucarate dehydratase-like	Mandelate racemase
1chrA[a]	370	Enolase C-terminal domain-like	D-Glucarate dehydratase-like	Chlormuconate cycloisomerase
7timA	247	Triosephosphate isomerase (TIM)	Triosephosphate isomerase (TIM)	Triosephosphate isomerase
1tml	286	Cellulases	Cellulases	Cellulase E2
1btc	491	Glycosyltransferases	β-Amylase	β-Amylase
1pii	457	Ribulose-phosphate-binding barrel	Tryptophan biosynthesis enzymes	N-(5′phosphoribosyl)antranilate isomerase
6xia	387	Xylose isomerase	Xylose isomerase	D-Xylose isomerase
5rubA	436	RuBisCO, C-terminal domain	RuBisCO, large subunit C-terminal domain	Ribulose 1,5-bisphosphate carboxylase-oxygenase
2taa	478	Glycosyltransferases	α-Amylases, N-terminal domain	Fungal α-amylases

[a]The first three examples come from the same superfamily and the same family, but they are different proteins (and from different species). Protein names: 4enl, enolase; 2mnr, muconate lactonizing enzyme-like; 1chrA, mandelate racemase.

TABLE III. The Structural Classification of the Globins Structural Comparisons

PDB code	Fold	Superfamily	Family	Species	Protein
1mbc	Globin-like	Globin-like	Globins	Sperm whale	Myoglobin
1hlb	Globin-like	Globin-like	Globins	Sea cucumber	Hemoglobin
2lh3	Globin-like	Globin-like	Globins	Yellow lupin	Leghemoglobin
1ecd	Globin-like	Globin-like	Globins	Midge fraction III	Erythrocruorin
2lhb	Globin-like	Globin-like	Globins	Sea lamprey	Lamprey globin
3sdhA	Globin-like	Globin-like	Globins	Ark clam	Hemoglobin
1thbA	Globin-like	Globin-like	Globins	Human	Hemoglobin α-chain
1mba	Globin-like	Globin-like	Globins	Sea hare	Myoglobin
1ith	Globin-like	Globin-like	Globins	Innkeeper worm	Hemoglobin
1cpc	Globin-like	Globin-like	Phyocyanins	Cyanobacterium	c-Phycocyanin
1colA	Toxins, membrane translocation	Colicin	Colicin	*Escherichia coli*	Colicin A

TABLE IV. The Proteins of the Helix-Bundle Comparison and Their Structural Classification

PDB code	Fold	Superfamily	Family	Species
1flx	Theoretical fold	Protein designs	Protein designs	De novo-designed gene
1aep	Apolipophorin III	Apolipophorin III	Apolipophorin II	African locust
1bbhA	4-helical up and down bundle	Cytochromes	Cytochrome c′	*Chromatium vinosum*
1bgeB	4-helical cytokines	4-helical cytokines	Long-chain cytokines	Canine
1le2	4-helical up and down bundle	Apoliporotein	Apolopoprotein	Human
1r	4-helical cytokines	4-helical cytokines	Short-chain cytokines	Human
256bA	4-helical up and down bundle	Cytochromes	Cytochrome b562	*Escherichia coli*
2ccyA	4-helical up and down bundle	Cytochromes	Cytochrome c′	*Rhodospirillum mollshianum*
2hmzA	4-helical up and down bundle	Hemerythrin	Hemerythrin	Sipunculid worm
3inkC	4-helical cytokines	4-helical cytokines	Short-chain cytokines	Human

recurring substructural core size as the number of molecules which are aligned increases.

The next entry in Table V presents the results obtained in the comparisons of the calcium binding proteins, clustered in the dataset of Fischer et al.[12] Although these proteins belong to the same fold and the same EF hand-like superfamily, some of the molecules are from different SCOP families. The proteins are 4cpv from the parvalbumin family; 2scpA, 2sas, 1top, and 1scmB from the calmodulin-like family; and 3icb from the Cal-binding D9K family. Their sizes range from 75 to 185 (Table I).

The next set of comparisons is from the α-globin fold; it includes the files 1mbc, 1hlb, 2lh3, 1ecd, 2lhb, 3sdhA, 1thbA, 1mba, 1ith, 1cpcL, and 1colA. Their sizes range from 136 for 1ecd to 197 residues for 1colA. The first nine molecules are globins, while the last two are a phycocyanin and a colicin, respectively. The latter belongs both to a different fold (toxins membrane translocation), and a different superfamily (colicin). We have carried out two types of multiple structure alignment: the first aligned only globins, increasing the number of molecules compared (globin A and globin B in Table V), and the second

TABLE V. Summary of the Results

Name of experiment	Molecules participating in experiment	No. of molecules	Avg no. of C_α	No. of *k-tuples* inserted	No. of C_α in top solutions
serp 6	7apiA, 8apiA, 1hleA, 1ovaA, 2achA, 9apiA	6	347	57,183	233 (69.1%)
serp 9	7apiA, 8apiA, 1hleA, 1ovaA, 2achA, 9apiA, 1psi, 1atu, 1kct	9	356	94,599	180 (53.4%)
serp 11	7apiA, 8apiA, 1hleA, 1ovaA, 2achA, 9apiA, 1psi, 1atu, 1kct, 1athA, 1attA	11	365	125,862	176 (52.2%)
serp 13	7apiA, 8apiA, 1hleA, 1ovaA, 2achA, 9apiA, 1psi, 1atu, 1kct, 1athA, 1attA, 1antl, 2antl	13	372	154,982	163 (48.4%)
serine prot	1cseE, 1sbnE, 1pekE, 3prk, 3tecE	5	277	118,023	220 (80.3%)
cal-bind	4cpv, 2scpA, 2sas, 1top, 1scmB, 3icb	6	140	12,435	31 (41.3%)
globin A	1mbc, 1hlb, 2lh3, 1ecd, 2lhb, 2sdhA, 1thbA	7	148	23,331	84 (59.6%)
globin B	1mbc, 1hlb, 2lh3, 1ecd, 2lhb, 2sdhA, 1thbA, 1mba, 1ithA	9	147	29,198	73 (51.8%)
globin cross	1mbc, 1thbA, 1cpc, 1colA	4	166	78,693	46 (32.6%)
globin grst	2hhbA, 2hhbB, 2hlb, 1mbd, 2hbg, 1mba, 1ecd	7	146	159,976	71 (52.2%)
Tim 2	4enl, 2mnr, 1chrA	3	388	115,104	191 (53.5%)
Tim C3	7timA, 1tml, 1btc	3	341	64,699	98 (40.0%)
Tim C5	7timA, 1tml, 1btc, 4enl, 1pii	5	383	135,179	66 (27.0%)
Tim C7	7timA, 1tml, 1btc, 4enl, 1pii, 6xia, 5rubA	7	391	55,802	40 (16.2%)
tpi	1amk, 5timA, 1htiA, 1timA, 1ypiA, 1treA, 1ydvA, 1aw2A	8	249	31,791	216 (87.8%)
hbundle A	1flx, 1aep, 1bbhA, 1bgeB, 1le2, 1rcb, 256bA, 2ccyA	8	129	77,498	31 (39.2%)
hbundle B	1flx, 1aep, 1bbhA, 1bgeB, 1le2, 1rcb, 256bA, 2ccyA, 2hmzA	9	127	82,118	31 (39.2%)
hbundle C	1flx, 1aep, 1bbhA, 1bgeB, 1le2, 1rcb, 256bA, 2ccyA, 2hmzA, 3inkC	10	140	46,039	27 (34.2%)

The data appearing in the columns are as follows: 1) name of the experiment; 2) PDB codes of molecules participating in the structural comparison experiment. The first molecule listed is the reference molecule; 3) number of aligned molecules; 4) number of C_α atoms per molecule; 5) number of collected *k-tuples;* 6) the number of C_α atoms in the geometric core. The percentage in brackets is computed in comparison to the smallest participating molecule, since the core cannot exceed the size of this molecule.

aligned molecules from different families (globin cross, Table V). As expected, the first type of comparisons yielded more extensive similarities than the latter. We have also carried out these automated multiple structural comparisons on a set of seven globin molecules (see globin grst in Table V). These have been previously analyzed by Gerstein and Levitt, using dynamic programming, iteratively carrying out pairwise alignments to obtain the median structure. The median structure subsequently served as the seed for the multiple alignment.[8] Our results are similar to those of these authors. They are presented in Table V (noted as globin grst).

Table V also presents the results we have obtained for the TIM barrel fold. We have carried out multiple structural comparisons for molecules from the same superfamily (e.g., Tim 2 in Table V), and from different superfamilies (e.g., Tim C3–Tim C7 in Table V). The structural classification of these molecules is presented in Table II. As noted in the Appendix, the automated multiple structure comparisons of the TIM barrels were the most runtime intensive, clearly owing to the size of these molecules. Additional examples of the TIM barrel fold include the triose phosphate isomerase family (tpi in Table V). This example was picked from the HOMSTRAD database.[13] These molecules belong to the same fold, superfamily, and family but come from different species. We did not limit our comparison to a detection of a multiple sequence core. We have included this structural alignment as an example illustrating how using this multiple structure alignment run, can serve for modeling purposes.

Table V also lists some results obtained in the structural comparisons of different folds of the same all-α class. The proteins 1flx, 1aep, 1bgeB, 1le2, 1rcb, 256bA, 2ccyA, 2hmzA, 3inkC of sizes from 79 (1flx) to 159 (1bgeB) were picked for this comparison. The folds include four-helix up and down bundle, four-helical cytokines, apolipophorin III, and a theoretical fold (of a de novo designed protein). For the 10 helix-bundle proteins, different spherical shell parameters have been used than for the other two. This is because of the recurrence of the helical conformation, and resulted in a reduction in the initial number of *k*-tuples inserted into the hash table, although the number of molecules in the comparison runs increased.

Figures 3–7 depict some of the results we have obtained and that are listed in Table V. Figures 3–7 have been created with the VMD[14] viewer. Figure 3 illustrates the superpositioning obtained for the globins. The results for all the helix bundle proteins are shown in Figure 4. Figures 5, 6, and 7 present the results obtained for TIM-barrel proteins. In Figure 5 we have taken the structurally aligned proteins from the triose phosphate isomerase HOMSTRAD family,[13] which was based on the multiple alignment obtained for this algorithm. For comparison, in Figure 6 the structures of the respective TIM

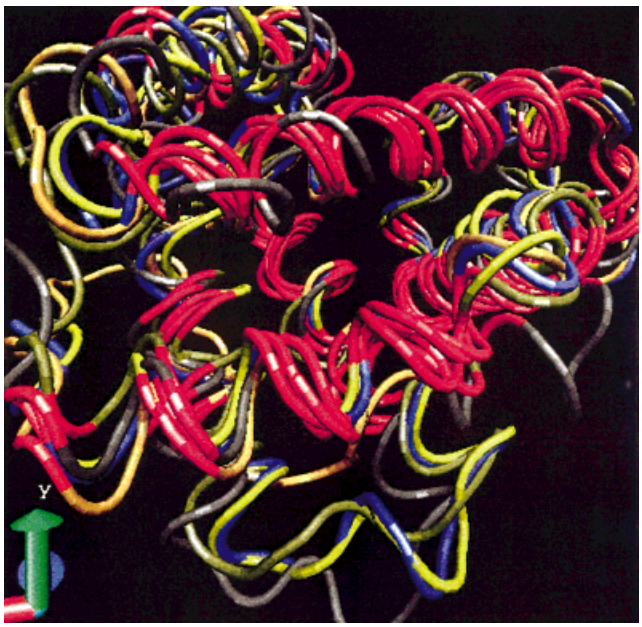

Fig. 3. Alignment of five globin molecules (PDB codes: 1mbc, 1hlb, 2hlb, 1thbA, and 1ithA). The C_α atoms participating in the core are highlighted in white. There is a core of five α-helices whose spatial formation is preserved in all molecules in the match.

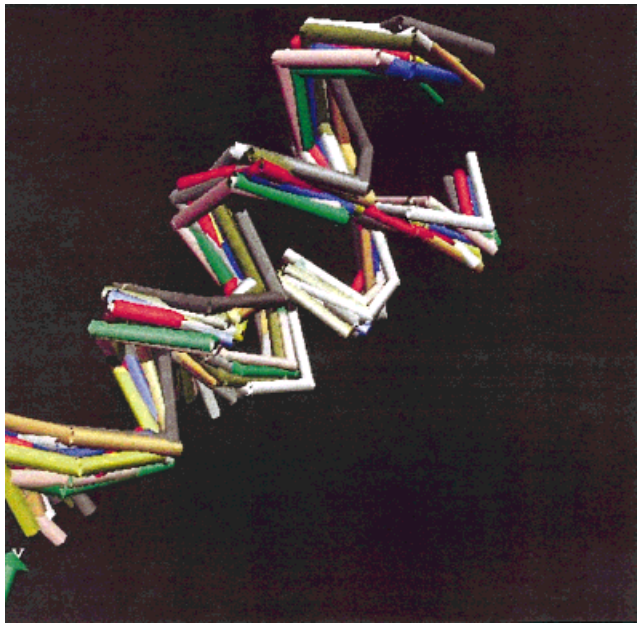

Fig. 4. Different view of the common core in all molecules in the α-helix bundle runs. The molecules are: 1flx, 1aep, 1bgeB, 1lea, 1rcb, 256bA, 1ccyA, 2hmzA and 3inkC.

barrels were taken as such, unaligned. Our algorithm carried out the multiple structure comparison in an entirely automated fashion. Figure 7 illustrates a more difficult multiple structure alignment, with the TIM barrels coming from different superfamilies in SCOP, and hence illustrating higher diversity.

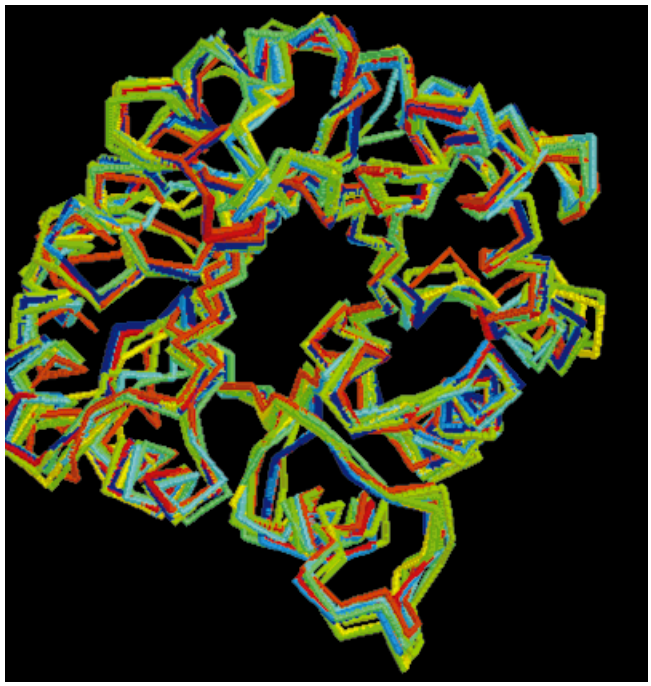

Fig. 5. The consensus structure of 10 proteins of the triose phosphate isomerase family, as manually aligned in HOMSTRAD.[13] In this run we have used their multiple sequence alignment. The molecules are: 1amk, 5timA, 1htiA, 1timA, 1ypiA, 1treA, 1yduA, 1aw2A, 2btmA, and 1tcdA.

Furthermore, this algorithm has been applied to the photosystems. Photosystems are a class of protein/chlorophyll complexes that function as antennas and reaction centers, converting solar energy into a useful electrochemical potential. Taken together, the sequence similarity between the various photosystems is insignificant (i.e., less than 20%). Using our method, multiple structural alignment, including alignments of different subgroups within the examined proteins was performed on bacterial photosynthetic reaction center (2prc, 1aij, 1clt), photosystem I (2pps) and photosystem II (1dop, along with theoretical models). Hence, the study included crystallographically resolved structures as well as theoretical ones and was conducted both on the full protein and on the subunit level. The results point to a structural core that is common to all photosystems, providing an insight to the conserved regions and their evolutionary path (I. Samish and A. Scherz, personal communication).

Currently, the algorithm is being applied to protein–protein interfaces, with the goal of detection of a recurring pattern of hotspots.[17] This type of residue-order-independent multiple structure alignment algorithm is ideal for this purpose. We begin by finding recurring patterns in families of interfaces[18] and proceed to see how general these are. A further application of such an algorithm is to find patterns on protein surfaces. Such work is currently in progress. Since patterns on protein surfaces, or their interfaces, are likely to be order-independent, and since we seek patterns that are general in binding, such a tool is particularly useful.

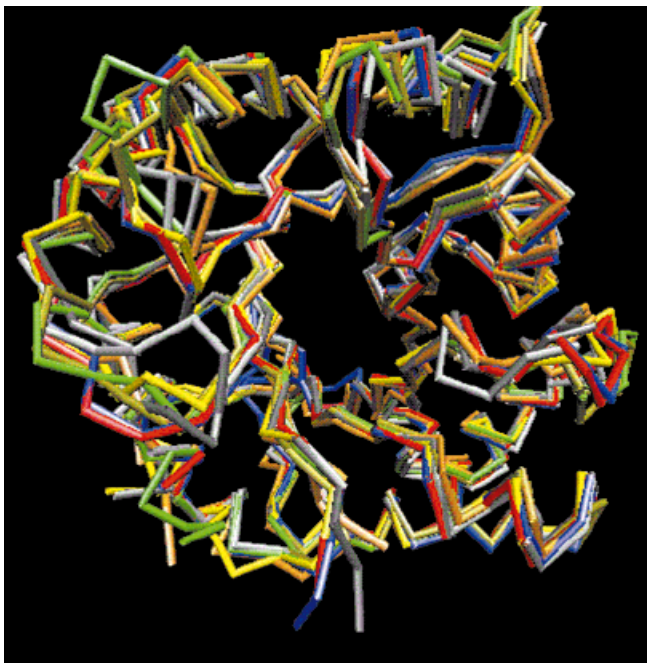

Fig. 6. The consensus structures of eight proteins of the triose phosphate isomerase family as obtained automatically by the MUSTA algorithm, with no prior knowledge of their correspondence. The molecules are 1amk, 5timA, 1htiA, 1timA, 1ypiA, 1treA, 1yduA, and 1aw2A.

Fig. 7. The purple molecule is the reference molecule in the cross TIM-barrel experiment, where the molecules derive from different superfamilies in the SCOP tree. The consensus of 8 such molecules is given in different colors (one color per molecule), which are, as expected, in the core of the TIM-barrel molecules. The molecules are 7timA, 1tml, 1btc, 4enl, 1pii, 6xia, 5rubA, and 2taa. this molecule.

CONCLUSIONS

We have presented a completely automated multiple structure comparison algorithm. Given an ensemble of structures, the algorithm aligns the structures and simultaneously detects a substructural motif (core) that recurs in all structures in the ensemble. The only input is the set of the atomic coordinates. No sequence information is used, and no initial seed is provided. The algorithm is amino acid sequence order-independent, and hence applicable to comparisons of active sites, protein–protein interfaces, protein surfaces, RNA, and drugs. We have implemented it and shown the results for the best alignment, producing the largest substructural motif. However, in practice, the algorithm returns a sorted list of highest scoring solutions, so suboptimal solutions are received as well.

The algorithm presented has performed well on a large set of protein examples, which include structures with different extents of similarity. Further, the examples tested derive from different types of structural classes. In the examples presented, we have used the coordinates of the C_α atoms. The fact that we start by detecting local structures (k-tuples), which appear simultaneously in all the molecules ensures performance, which is superior to pairwise structural alignments. Alignments initiating from pairwise superpositioning may suffer from spurious matches, which might be induced by the density of the structures. The run times of the algorithm are attractive, in the range of minutes on a PC on most examples, and up to a few hours for the case of the large TIM barrel proteins.

We address the problem of multiple structure alignment, with the requirement that all structures in the ensemble contribute to the alignment, and to the substructural motif. We do not address the related problem, i.e., an alignment algorithm that finds the geometric core shared only by a subset of the ensemble, which gives a large multiple structural alignment. Clearly, the smaller the subset, the easier it is to find larger aligned substructures. Yet, some balance has to be achieved between the number of aligned structures and the size of the multiple matching substructure. An unattractive way to carry out such a chore is through an application of this MUSTA algorithm to subsets of combinations of the molecules in the ensemble. Currently, we are working on an automated algorithm that addresses this problem.

ACKNOWLEDGMENTS

The authors thank Meir Fuchs for contributing software to this project and Dr. Jacob V. Maizel for discussions and encouragement. The research of H. J. Wolfson and R. Nussinov in Israel has been supported in part by the Magnet grant, by the Ministry of Science grant, and by the Center of Excellence in Geometric Computing and its Applications, funded by the Israel Science Foundation (administered by the Israel Academy of Sciences). The research of H.J.W. is partially supported by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University. This project has been funded in whole or in part with federal funds from the National Cancer Institute, National Institutes of Health, under contract number NO1-CO-56000. The content of this publication does not necessarily reflect the view or policies of the Department of Health and Human Services, nor does mention of trade

REFERENCES

1. Lin SL, Nussinov R, Fischer D, Wolfson HJ. Molecular surface representation by sparse critical points. Proteins 1994;18:94–101.
2. Lin SL, Nussinov R. Molecular recognition via face center representation of a molecular surface. J Mol Graph 1996;14:78–90.
3. Nussinov R, Wolfson HJ. Efficient detection of three-dimensional motifs in biological macromolecules by computer vision techniques. Proc Natl Acad Sci USA 1991;88:10495–10499.
4. Gerstein MB, Altmann RB. A structurally invariant core for the globins. Comput Appl Biosci (CABIOS) 1995;11:633–644.
5. Arun KS, Huang JS, Blostein SD. Least squares fitting of two 3-D point set. IEEE Trans Pattern Anal Machine Intell 1987;9:698–700.
6. Gelfand I, Kister A, Kulikowski C, Stoyanov O. Geometric invariant core for the V_L and V_H domains of immunoglobulin molecules. Protein Eng 1998;11:1015–1025.
7. Orengo CA, Taylor WR. SSAP: Sequential structure alignment program for protein structure comparison. Methods Enzymol 1996;266:617–635.
8. Gerstein MB, Levitt M. Using iterative dynamic programming to obtain accurate pairwise and multiple alignments if protein structures. In: Proceedings of the International Conference on Intelligent Systems for Molecular Biology, AAAI Press; 1996. p 59–67.
9. Lamdan Y, Wolfson HJ. Geometric Hashing: a general and efficient model-based recognition scheme. In: Proceedings of the IEEE International Conference on Computer Vision, Tampa, FL; 1988. p 238–249.
10. Murzin AG, Brenner SE, Hubbard T, Chothia C. SCOP: a structural classification of proteins database for the investigation of sequences and structures. J Mol Biol 1995;247:536–540.
11. Bernstein FC, Koetzle TF, Williams GJB, Meyer EF Jr, Brice MD, Rodgers JR, Kennard O, Shimanouchi T, Tasumi M. The protein databank: a computer-based archival file for macromolecular structures. J Mol Biol 1977;112:535–542.
12. Fischer D, Tsai CJ, Nussinov R, Wolfson HJ. A 3-D sequence-independent representation of the protein databank. Protein Eng 1995;8:981–997.
13. Mizuguchi K, Deane CM, Blundell TL, Overington JP. HOMSTRAD: a database of protein structure alignments for homologous families. Protein Sci 1998;7:2469–2471.
14. Humphrey W, Dalke A, Schulten K. VMD—visual molecular dynamics. J Mol Graph 1996;14:33–38.
15. Leibowitz N, Fligelmann ZY, Nussinov R, Wolfson HJ. Multiple structural alignment and core detection by geometric hashing. In: Proceedings of the International Conference on Intelligent Systems for Molecular Biology; 1999. p 169–1.
16. Leibowitz N. Multiple structural alignment of proteins. M.Sc. thesis, Computer Science Department, Tel Aviv University; 1999.
17. Hu Z-J, Ma B, Wolfson H, Nussinov R. Conservation of polar residues as hot spots of protein interfaces. Proteins 39:331–342, 2000.
18. Tsai C -J, Lin S-L, Wolfson H, Nussinov R. A dataset of protein–protein interfaces generated with a sequence-order-independent comparison technique. J Mol Biol 1996;260:604–620.

APPENDIX

The Parameters

The following are the main parameters used by the multiple structure algorithm:

minShell,maxShell—Defines the shell that is used when collecting *k-tuples*. For each C_α-atom, we form *k-tuples* centered on this atom, and which includes atoms appearing in the 3-D spherical shell. The inner radius is minShell and the outer radius is maxShell. Clearly, the shell size influences the number and the type of congruent *k-tuples* serving as input to the algorithm. We used two main ranges 6–10 A and 9–12 A. The first is for local motifs and the latter for global motifs.

maxRMS—This value is used in the determination of which source, and reference *k-tuple* are congruent. The more liberal the maxRMS, the larger the number of *k-tuples* that are passed to the following stages. It further affects the goodness of the obtained geometrical similarity. The runs whose results are presented here were performed with a maxRMS range of 0.8–1.5 A.

transformationClusterDist—Relates to the transformation distance, i.e. the difference between their masks (see description of the clustering procedure in the outline of the algorithm). The transformation distance specifies the allowed percentage of consistent pairs that are not shared by the two transformations. We used a value of 0.35.

minCoreSize—This value specifies the minimum size of the geometrical core we allow. Here this parameter is at least 20.

vicDist—Specifies when two C_α atoms from different molecules are considered to be in the vicinity of each other. If the distance between the atoms is below this threshold, they are defined as a matching pair. There is a trade-off in the definition of this value. A larger value will produce larger cores, albeit with larger RMSDs.

Performance

The structural comparisons were carried out on a PC with a 400 Mhz processor, 256Mbyte RAM memory, under the Linux operating system. The code is written in C++. The running times range from seconds to hours, depending on the number of molecules, their sizes, the similarity between them and the values of the values assigned to the parameters. Most of the structural comparisons terminated within minutes. Longer running times (a few hours) were required for the TIM barrels. This algorithm is memory intensive. Hence, increasing the RAM size should yield even faster running (CPU) times.