BIOINFORMATICS ORIGINAL PAPER
Vol. 28 no. 24 2012, pages 3274–3281
doi:10.1093/bioinformatics/bts618

Structural bioinformatics
Advance Access publication October 23, 2012

Fast protein structure alignment using Gaussian overlap scoring of backbone peptide fragment similarity

David W. Ritchie[1,*], Anisah W. Ghoorah[1,2], Lazaros Mavridis[3] and Vishwesh Venkatraman[4]
[1]Inria Nancy, 615 Rue du Jardin Botanique, 54600 Villers-lès-Nancy, [2]Université de Lorraine, LORIA, 54506 Nancy, France, [3]University of St. Andrews, St. Andrews KY16 9AJ, Scotland, UK and [4]Norwegian University of Science and Technology, Høgskoleringen 5, Trondheim, Norway

Associate Editor: Anna Tramontano

ABSTRACT
Motivation: Aligning and comparing protein structures is important for understanding their evolutionary and functional relationships. With the rapid growth of protein structure databases in recent years, the need to align, superpose and compare protein structures rapidly and accurately has never been greater. Many structural alignment algorithms have been described in the past 20 years. However, achieving an algorithm that is both accurate and fast remains a considerable challenge.

Results: We have developed a novel protein structure alignment algorithm called 'Kpax', which exploits the highly predictable covalent geometry of C_α atoms to define multiple local coordinate frames in which backbone peptide fragments may be oriented and compared using sensitive Gaussian overlap scoring functions. A global alignment and hence a structural superposition may then be found rapidly using dynamic programming with secondary structure-specific gap penalties. When superposing pairs of structures, Kpax tends to give tighter secondary structure overlays than several popular structure alignment algorithms. When searching the CATH database, Kpax is faster and more accurate than the very efficient Yakusa algorithm, and it gives almost the same high level of fold recognition as TM-Align while being more than 100 times faster.

Availability and implementation: http://kpax.loria.fr/.

Contact: Dave.Ritchie@inria.fr.

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on May 19, 2012; revised on October 4, 2012; accepted on October 14, 2012

1 INTRODUCTION

Aligning and comparing protein structures is important for understanding their evolutionary and functional relationships (Hasegawa and Holm, 2009; Sierk and Kleywegt, 2004). By quoting Lewis Carrol's Red Queen character (*it takes all the running you can do, to keep in the same place*), Holm *et al.* (2008) recently alluded to the computational challenge of searching increasingly large protein structure databases. Today, 4 years later, with some 80 000 protein structures in the Protein Databank and with the number of new structures being solved growing exponentially (Berman, 2008), the need to compare the 3D structures of protein molecules rapidly and reliably has never been greater. Efficient pattern matching algorithms such as FASTA (Lipman and Pearson, 1985) and BLAST (Altschul *et al.*, 1990) are now standard tools for searching nucleotide and amino acid sequence databases. Dynamic programming (DP) algorithms provide a rapid way to find the optimal global (Needleman and Wunsch, 1970) or local (Smith and Waterman, 1981) alignments of pairs sequences. However, there is still no generally accepted standard for how to align and compare two similar protein structures (Sippl and Wiederstein, 2008).

This discrepancy arises because the additional complexity of working in 3D space makes structure alignment considerably more difficult than sequence alignment. For example, two common and closely related measures of structural similarity are the number of residue equivalences and the root mean squared deviation (RMSD) between the corresponding C_α atoms. However, while it is often possible to improve one at the expense of the other, it is difficult to optimize these two quantities simultaneously (Zemla, 2003). Furthermore, because RMSD values tend to be dominated by long-distance pairs, it can be useful to restrict RMSD-based similarity measures to selected 'core' residues (Sierk and Kleywegt, 2004). But this then raises the question of precisely which residues should be considered as core residues. Consequently, different similarity scoring schemes can assign different sets of 'optimal' residue equivalences (Hasegawa and Holm, 2009).

Despite such difficulties, many practical structural alignment algorithms have been described (Taylor *et al.*, 2001), and the number of recent publications is growing rapidly (Hasegawa and Holm, 2009). These algorithms can often be classified according to how they use the internal geometries of proteins to calculate interatomic distances (Holm and Sander, 1993; Taylor and Orengo, 1989; Zhou and Tang, 2005; Zhu and Weng, 2005) or cliques of distances (Malod-Dognin *et al.*, 2010), interatomic C_α vectors (Chew *et al.*, 1999; Ortiz *et al.*, 2002), or triplets of C_α atoms (Venkateswaran *et al.*, 2011), torsion angles (Charpentier *et al.*, 2005; Jung *et al.*, 2011; Täubig *et al.*, 2006; Tung *et al.*, 2007), or combinations of distances and angles (Shen *et al.*, 2010; Ye *et al.*, 2004). Geometric-hashing (Kifer *et al.*, 2011) and Voronoi tessellation techniques have also been used to calculate pose-invariant measures of structural similarity (Ilyin *et al.*, 2004; Sacan *et al.*, 2008).

Different algorithms may also be distinguished according to whether and how they use secondary structure elements (SSEs) in the similarity scoring function or whether they match SSEs as

*To whom correspondence should be addressed structural units (Gibrat *et al*., 1996; Kawabata and Nishikawa, 2000; Kolbeck *et al*., 2006; Krissinel and Henrick, 2004; Lu, 2000; Shindyalov and Bourne, 1998; Szustakowski and Weng, 2000; Zhang *et al*., 2010). For example, Sheba finds an initial superposition using a DP scoring matrix that combines sequence homology, secondary structure similarity, solvent accessibility and polarity to obtain an initial superposition which is then optimized using DP to maximize the number of close C_α–C_α distances. The method of Jung and Lee (2000), ProSup (Lackner *et al*., 2000) and TM-Align (Zhang and Skolnick, 2005) identify short seed fragments to give initial superpositions which are then optimized in similar ways. Yet other approaches define and match higher-order structural alphabets (Konagurthu *et al*., 2008; Lo *et al*., 2007; Razmara *et al*., 2012; Stivala *et al*., 2009; Tyagi *et al*., 2007; Yang and Tung, 2006) or fragments that might subsequently be re-assembled (Budowski-Tal *et al*., 2010; Pandit and Skolnick, 2008). Most algorithms treat proteins as rigid-body objects, but a few can take into account structural flexibility (Salem *et al*., 2010; Ye and Godzik, 2003), permutations of structural motifs (Chen *et al*., 2006; Ilyin *et al*., 2004; Sabarinathan *et al*., 2010; Sippl and Wiederstein, 2012; Szustakowski and Weng, 2000) and even composite alignments involving multiple chains (Sippl and Wiederstein, 2012). However, almost all of these approaches are prohibitively expensive if the aim is to search large protein structure databases.

As well as the diverse geometric and symbolic representations listed above, algorithms based on sophisticated algebraic approaches have also been described, including Lagrangian contact map optimization (Andonov *et al*., 2008), eigenvector analysis (Shibberu and Holder, 2011) and Fourier correlation techniques (Mavridis *et al*., 2012). Recent improvements to the contact map optimization approach using integer linear programming (Andonov *et al*., 2011; Wohlers *et al*., 2010) can now find provably optimal structural alignments in favourable cases. However, as before, such techniques are currently too expensive to search large structural databases.

Here, our aim is to meet the need for a general purpose structural alignment algorithm that is sufficiently fast to support *ad hoc* queries against large structural databases while also being sufficiently sensitive to provide high quality (but not necessarily optimal) pair-wise alignments. Our overall approach is motivated by the basic observation that the four covalent bonds of each C_α atom have a highly predictable tetrahedral geometry, which could be used to superpose arbitrary pairs of C_α atoms and their covalent partner atoms using least-squares fitting. Thus, armed with the corresponding 3D fitting matrix, the relative positions of neighbouring up-stream and down-stream residues could easily be compared and scored.

However, rather than using explicit least-squares fitting, which is a computationally expensive operation and which requires the coordinates of two sets of atoms, we note that a similar (but not strictly identical) transformation can be achieved by individually translating each C_α atom to the coordinate origin and by aligning two of its covalent bonds with the coordinate axes. In other words, applying such a transformation will allow the backbone peptide fragment associated with each C_α position to be placed in a standard, or canonical, orientation. In the context of database search, this is clearly advantageous because it allows each structure to be pre-processed separately in linear time.

From this starting point, we calculate a similarity score for putative pairs of backbone fragments using a sum of products of Gaussian functions centred on the C_α positions within each fragment and on a small number of further 'virtual' atom positions that encode the apparent centre of mass (COM) of each protein, as described below. We then use these Gaussian overlap scores to construct a DP scoring matrix with SSE-dependent gap penalties which allows an optimal set (according to our scoring function) of equivalent residues to be found and superposed efficiently. A particular feature of a scoring function based on sums of Gaussians is that it implicitly favours close contacts without necessarily needing to exclude long-range pairs. Although we implemented SSE-dependent gap penalties initially to improve database retrieval performance, we find that using such penalties together with our Gaussian scoring function tends to produce alignments with smaller numbers of aligned residues but with somewhat tighter 3D overlays of SSEs than the other alignment algorithms studied here.

It should be noted that Taylor and Orengo (1989) first demonstrated the utility of using local coordinate systems. They constructed a local coordinate frame for each residue using N-C and C_β-H bond vectors, but crucially they used it only to compare patterns of *intra*-molecular vectors rather than *inter*-molecular distances. Although later approaches that use internal torsion angles such as Yakusa (Charpentier *et al*., 2005) and SABIC (Shen *et al*., 2010) share a similar insight to our approach, the inter-molecular scoring functions in these algorithms can only compare one angle or one distance for each pair of residues. In contrast, for each position along a backbone, our Gaussian scoring function can be used to score the similarity of two local backbone fragments by comparing the positions of up to three residues in each direction along the chain. Hence, we also use Gaussian overlap scores to define the secondary structure of each residue according to its similarity to a model α-helix or β-sheet. On a contemporary workstation, our multi-threaded algorithm can calculate thousands of structural alignments per second. Hence we named it 'Kpax' (being short for 'thousands of protein alignments by canonical C_α coordinate centres per second').

2 METHODS

The Kpax similarity score for a pair of residues i and j has the form:

$$K_{ij} = w_l K_{ij}^{\text{local}} + w_s K_{ij}^{\text{spatial}} + w_b K_{ij}^{\text{blosum}}, \quad (1)$$

where each K is a normalized score with a value between zero and one, and each w is a weight factor in the same range and normalized such that $w_l + w_s + w_b = 1.0$. The first term gives a measure of the local similarity of a pair of residues when calculated in a common coordinate frame. The second term measures their spatial similarity with respect to the relative position of the COM of a protein (also calculated in the same frame). These two terms are described in more detail below. The final term is the Blosum62 amino acid similarity score, in which each pair-wise score has been scaled onto the above range. Here, we set $w_b = 0$ in order to consider only structure-based alignments.

2.1 Constructing local C_α coordinate frames

Here, we set up a local 3D coordinate system for each amino acid residue by using the coordinates of its C_α, C and N backbone atoms to construct

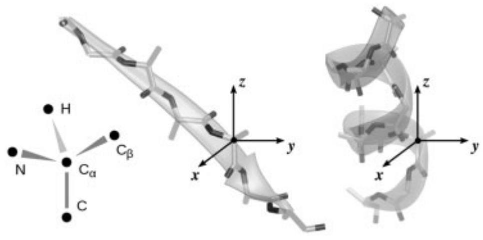

Fig. 1. Constructing a local coordinate frame about a tetrahedral C_α coordinate centre. The C_α atom is placed at the origin, the C atom on the negative z axis, and the N atom in the positive x region of the xz plane

a 3D transformation matrix, $\underline{K}$, defined as a sequence of transformations consisting of (i) a translation that locates the C_α atom at the origin, (ii) a rotation that places the C atom on the negative z axis, and (iii) a further rotation about the z axis that places the N atom in the xz plane with a positive x coordinate. For the 19 common non-glycine L-amino acid residues, applying $\underline{K}$ will place the C_β and H_α atoms on the positive and negative y sides of the xz plane, respectively. For glycine, applying $\underline{K}$ will locate the H_α and H_β symmetrically on either side of the xz plane. However, none of the above backbone atoms play any subsequent role in our scoring function because their local-frame coordinates are almost invariant.

Figure 1 shows how fragments of a theoretical α-helix and β-strand from the CCP4 fragment library (Cowtan, 2008) may be located in a canonical orientation at the origin using the $\underline{K}$-transform. Using a different residue to calculate the $\underline{K}$-transform would shift each fragment by the corresponding number of peptide units along its principal secondary structure axis. For an infinitely repeating polyalanine α-helix or β-strand, applying the $\underline{K}$-transform for each residue would give an indistinguishable result. On the other hand, given a suitable scoring function, it is natural to suppose that near the coordinate origin, similar but non-identical SSEs could be brought into close register by applying a suitable local frame shift of one structure relative to the other.

2.2 The local frame fragment similarity score

To calculate the similarity of two backbone fragments using a smooth scoring function, we represent the fragment centred on residue i as a product of normalized Gaussian density functions located on n down-stream and n up-stream C_α atoms:

$$\psi_i = \varphi_i^{-1}(\underline{x}_{i-1})\varphi_i^{+1}(\underline{x}_{i+1}) \ \ldots \ \varphi_i^{-n}(\underline{x}_{i-n})\varphi_i^{+n}(\underline{x}_{i+n}). \qquad (2)$$

Each individual Gaussian function has the form

$$\varphi_i^k(\underline{x}_{i+k}) = N_k e^{-\beta_k r_k^2/2\sigma_k^2}, \qquad (3)$$

where r_k is the distance from the kth C_α atom, β_k is a scale factor and σ_k is the Gaussian width of the k'th density function. N_k is a normalization factor such that

$$\int \varphi_i^k(\underline{x}_{i+k})^2 \mathrm{d}\underline{x}_k = 1. \qquad (4)$$

We then transform each of the 2 n C_α atom coordinates of a pair of fragments, i and j, into a common local coordinate frame using $\underline{K}_i$ and $\underline{K}_j$, and we calculate the local similarity score between two fragments as a multi-term overlap integral

$$K_{ij}^{\mathrm{local}} = \int \psi_i\psi_j \mathrm{d}\underline{x}_{-1}\mathrm{d}\underline{x}_{+1} \ \ldots \ \mathrm{d}\underline{x}_{-n}\mathrm{d}\underline{x}_{+n}. \qquad (5)$$

The overall scheme is illustrated in Figure 2. Assuming there is no overlap between density functions belonging to different C_α atoms and that equivalent C_α atom positions may be assigned the same Gaussian

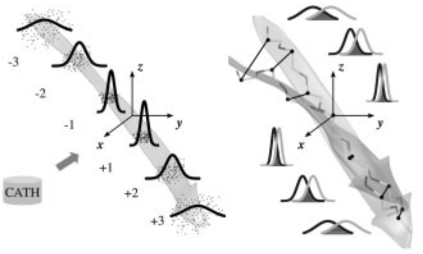

Fig. 2. Calculating local pair-wise similarity scores. The local-frame distribution of all C_α coordinates from the CATH database (left) is used to calculate the Gaussian width, σ_k, for each up-stream and down-stream residue position. For a given pair of residues, i and j, the 'K-score' is calculated as a product of Gaussian overlap integrals (Equation 6) using the local frame pair-wise C_α distances (right)

parameters, it can be shown (Boys, 1950) that the above integral may be simplified to give

$$K_{ij}^{\mathrm{local}} = e^{-\sum\limits_{k=-n}^{n} \beta_k R_{i+k,j+k}^2/4\sigma_k^2}, \qquad (6)$$

where $R_{i+k,j+k}$ is the distance between the C_α atoms at positions $i+k$ and $j+k$ on chains A and B, respectively, and β_k is a scale factor which we currently set to unity. The summation excludes $k=0$ because $R_{i+0,j+0}=0$ by construction. In other words, the local similarity between residues i and j is calculated as a Gaussian sum of the squared distances between pairs of up-stream and down-stream C_α atoms.

To obtain suitable values for the parameters σ_k, we treat each σ as the standard deviation (SD) of a normal Gaussian distribution, and by considering each residue in turn of each domain in the CATH database (Cuff *et al.*, 2009), we calculated the mean and SDs of all residues at relative positions ±1 to ±3 with respect to the residue under consideration to obtain the values: $\sigma_{+1}=1.46$, $\sigma_{-1}=1.03$, $\sigma_{+2}=3.72$, $\sigma_{-2}=3.54$, $\sigma_{+3}=5.52$ and $\sigma_{-3}=5.74$. To apply the above scoring function to residues near the N and C termini of one or both chains, we use a simple wrapping scheme in order to maintain the total number of terms in Equation (6). For example, when $i=1$, there are no residues with negative offsets, and so the contribution from φ_i^{+1} to φ_i^{+n} is doubled in Equation (6). When $i=2$, the contributions from φ_i^{+1} and φ_i^{-1} are calculated using Equation (6), but the contribution from φ_i^{+2} to φ_i^{+n} is doubled, and so on. A similar scheme is used for the C-terminal residues.

2.3 The spatial similarity score

If two protein domains which share a similar fold are superposed, their COMs will often be close together. Thus, if a protein's COM is transformed into the local frame of each residue, it follows that residues which are well aligned will 'see' the COM in similar positions in space (although consecutive residues will see the COM in quite different positions). In particular, the COM will appear in approximately the same location only for every fourth residue of an α-helix, and every second residue of a β-sheet. To exploit this property in a similar way to the local frame fragment scoring function, we transform the COM into the local frame of each residue, and we place a virtual atom (VA) at a distance of 2 Å from the origin on the direction vector of the transformed COM. This is illustrated in Figure 3. Thus, each VA represents and encodes the local frame spatial direction of a protein's COM. Then, in analogy to Equation (6), the spatial similarity score is calculated as a product of Gaussian overlap integrals

$$K_{ij}^{\mathrm{spatial}} = e^{-\sum\limits_{k=-n}^{n} \beta_k R_{i+k,j+k}^2/4\tau_k^2}, \qquad (7)$$

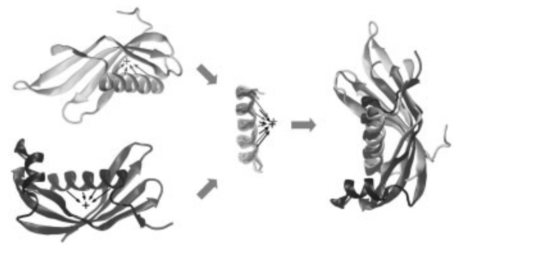

Fig. 3. An illustration of the principle behind structural alignment using local coordinate frame COM vectors. If two similar proteins are superposed, the COM vectors from equivalent aligned pairs of residues will often be co-linear, as shown here for PDB codes 1cew and 1 mol. Conversely, a pair of similar proteins may be superposed by aligning their COM vectors

where $R_{i+k,j+k}$ is the distance between the VAs at positions $i+k$ and $j+k$, and the summation excludes $k=0$ as before. Each τ_k is considered as the SD of a Gaussian distribution of the VA coordinates. Hence, VAs were placed on all of the residues of the CATH database and the SDs were calculated from the resulting distributions to give $\tau_{+1}=2.43$, $\tau_{-1}=2.17$, $\tau_{+2}=4.13$, $\tau_{-2}=3.93$, $\tau_{+3}=5.74$ and $\tau_{-3}=5.58$.

2.4 Assigning secondary structure elements

By placing the centre residue of a theoretical model of an α-helix and a β-strand at the local coordinate origin, Equation (6) provides a straight-forward way to detect the SSE type of a given residue by calculating its local-frame similarity to each model fragment. Here, we use the five-residue template files *theor-helix-5.pdb* and *theor-strand-5.pdb* from the CCP4 fragment library (Cowtan, 2008), and for each residue in a given structure we assign type α to residue i if $K^{\alpha}_{i,3} > K^{\beta}_{i,3} > 0.1$, we assign β if $K^{\beta}_{i,3} > K^{\alpha}_{i,3} > 0.1$, and we assign γ (i.e. loop or coil) otherwise. Any singleton α or β residues within a γ region are re-assigned to γ.

2.5 Dynamic programming structural alignments

The optimal 'pose-invariant' alignment for two chains, A and B, of length N_A and N_B is calculated by first initializing a DP matrix of dimension $(N_A+1)\times(N_B+1)$ using $D_{0,j}=D_{i,0}=0.0$, and by filling the remaining elements using

$$D_{ij} = \max(D_{i-1,j-1}+K_{ij}, D_{i,j-1}-P^A_i, D_{i-1,j}-P^B_j), \quad (8)$$

where K_{ij} is the similarity score for residues i and j (Equation 1) and P_i is the penalty for introducing a gap between residues i and $i-1$. Since a typical C_α–C_α distance is about 3.8 Å, we set $\sigma=(\sigma_{+1}+\sigma_{-1})/2=1.245$ Å and we calculate the gap penalty unit, ρ, using $\rho=e^{-3.8^2/4\times1.245^2}\simeq 0.1$. In other words, the gap penalty is derived directly from the physical length scale of one peptide unit.

To penalize alignment gaps within regular secondary structures and to encourage gaps in loop regions, we set $P_i=2\rho$ if positions i and $i-1$ were both called as α, $P_i=\rho$ for both β, $P_i=\rho/2$ for both γ, and $P_i=\rho$ otherwise. No gap extension penalty is used here, nor is any penalty applied for overhangs at the start or end of a chain. However, if a chain contains any physical breaks (we assume a chain break exists if the distance between consecutive C_α atoms exceeds 1.5×3.8 Å), we set the gap opening penalty to zero for the two residues that border each break. Because the local Gaussian scoring function automatically generates low similarity scores for any residues near a physical break, this is sufficient to ensure that chains with missing segments are handled gracefully.

By tracing through the DP matrix in the usual way, we obtain a global alignment in which each diagonal step corresponds to a matched pair of residues. An overall 'K-score' is then calculated as

$$K_{AB} = \sum_{i=1}^{N_A}\sum_{j=1}^{N_B}\mu_{ij}K_{ij}, \quad (9)$$

where $\mu_{ij}=1$ if residues i and j are matched, and $\mu_{ij}=0$ otherwise. Despite being a global structural similarity score, it is worth noting that this penalty-free score is 'pose-invariant' in that it does not depend on the orientations of the given proteins, and that for two perfectly matching backbones it will be numerically equal to the number of aligned residues. It is also worth noting that this score does not depend on the order in which the chains are given, and it does not involve any least-squares fitting calculations. Indeed, our technique of placing backbone peptide fragments in a canonical orientation at the coordinate origin costs only $O(N)$ operations per protein. Populating the DP scoring matrix still requires $O(N^2)$ operations, although the form of Equations (6) and (7) allows this cost to be reduced to essentially just two exponential function calls per matrix element.

For typical protein domains, we find that K^{local}_{ij} and K^{spatial}_{ij} are almost equally effective scoring functions. Therefore, by default, we use a combination of both scores with weights $w_l=0.5$, $w_s=0.5$. When superposing similar domains, the best path through the DP matrix is often near the main diagonal. Hence, in principle, many pair-wise similarity scores never need to be calculated. However, because the overall algorithm is so fast, we find that the main rate limiting step comes from reading the coordinate data into computer memory. Therefore, we currently do not apply any banding or lazy evaluation techniques to accelerate the DP calculation.

2.6 Calculating and optimizing 3D superpositions

Given an alignment from the DP matrix, an initial 3D structural superposition is calculated by least-squares fitting (Kabsch, 1976) in which the pair-wise K-scores are used as fitting weights. This is then refined by one further cycle of fitting with uniform weights in which the weight for any pair of residues is set to zero if the distance between their C_α atoms exceeds 8 Å. This often produces a visually acceptable superposition. We then optimize this initial superposition by applying further rounds of DP and fitting using a pose-dependent Gaussian score based on pairs of C_α distances:

$$G^{\text{pose}}_{ij} = e^{-R^2_{ij}/4\sigma^2_{\text{pose}}}, \quad (10)$$

where a Cartesian grid is used to find residue pairs within the above distance threshold, and where no gap penalty is used in the DP matrix. This procedure allows some additional residue pairs (e.g. in loop regions) to be found and added to the alignment. In analogy to Equation (9), we then define a pose-dependent Gaussian superposition score as

$$G_{AB} = \sum_{i=1}^{N_A}\sum_{j=1}^{N_B}\mu_{ij}G^{\text{pose}}_{ij}. \quad (11)$$

When using $\sigma_{\text{pose}}=1.4$ Å, a gap penalty of $\rho=0.1$, and uniform fitting weights, we find that applying just two rounds of Gaussian optimization is normally sufficient to give a satisfactory superposition with a low C_α RMSD. Sippl and Wiederstein (2012) recently described a similar Gaussian sum expression to rank alignments generated by their TopMatch algorithm.

2.7 Searching structural domain databases

To allow efficient queries against structural databases such as CATH or SCOP (Murzin *et al.*, 1995), we first pre-calculate and store the up-stream and down-stream fragment coordinates for every database residue (i.e. 6 C_α and 6 VA coordinates per residue). This allows a database to be searched rapidly using a single round of DP to calculate the K-score similarity between the query and each member of the database.

However, in order to rank alignments and superpositions of chains of different lengths, Kpax uses normalized scores defined by

$$\bar{K}_{AB} = K_{AB}/(N_A N_B)^{1/2}, \quad \bar{G}_{AB} = G_{AB}/(N_A N_B)^{1/2}. \tag{12}$$

In particular, $\bar{K}_{AB} = 1$ and $\bar{G}_{AB} = 1$ represent the perfect alignment and superposition of two identical backbones, respectively. Except for queries involving highly populated domain families such as the immunoglobulins, often only a relatively small number of database structures will superpose well onto the query. Hence, Kpax calculates 3D superpositions and G-scores only for the top 300 structures with the best K-scores.

2.8 Implementation details and availability

Kpax has been implemented in the C programming language using thread-safe programming techniques. Hence all database searches are multi-threaded by default. The program and some command scripts for building CATH and SCOP databases on Linux systems are available at *http://kpax.loria.fr/*. Databases of user-defined collections of PDB files may be built in a similar way. However, Kpax currently assumes that each PDB file contains just one chain, and it only considers the first model of any multi-model structures. PDB files of the structures used in the following sections are also available from the above address.

3 RESULTS AND DISCUSSION

3.1 Comparing SSE assignments with Stride and DSSP

Figure 4 shows an example of the SSE assignments obtained using the Kpax template-matching procedure (Section 2.4) in comparison with the SSEs calculated by DSSP (Kabsch and Sander, 1983) and Stride (Frishman and Argos, 1995). Supplementary Figure S1 shows a colour version of this figure along with a further nine examples. These figures show that our algorithm often assigns quite similar α and β SSEs to Stride and DSSP, although there are often some small differences around the start and end positions of each SSE. Also, as expected, Kpax does not distinguish specific types of turn from random coil regions. Nonetheless, because the main aim here is to provide SSE-dependent alignment gap penalties, the above procedure allows the secondary structure environment of each residue to be estimated rather well and very rapidly without requiring external software.

3.2 Comparison with CE, Sheba and TM-Align

As a first test of the Kpax alignment algorithm, we compared its performance with CE, Sheba and TM-Align using the 10 ‘difficult’ pairs of structures from Fischer *et al.* (1996) that were first used as a reference benchmark by Shindyalov and Bourne (1998) and later by several other groups (Lackner *et al.*, 2000; Novotny *et al.*, 2004; Shen *et al.*, 2010; Shibberu and Holder, 2011). Table 1 indicates that all of these algorithms can calculate good alignments for these structures, although the variation in some of the numbers of aligned residues and RMSDs suggests that these examples still appear to be difficult to align consistently. On the other hand, Supplementary Figure S2 confirms graphically that each algorithm gives rather similar 3D superpositions for each pair of folds. This demonstrates the difficulty of trying to compare different structure alignment algorithms directly using such raw numerical measures. Nonetheless, it is worth

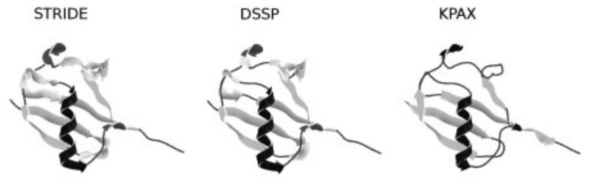

Fig. 4. Cartoon representations of the SSE assignments calculated by Stride, DSSP and Kpax for ubiquitin (PDB code: 1ubq). Supplementary Figure S1 shows nine further examples

noting that the Kpax superpositions have RMSDs that are lower than CE and TM-Align in all cases, and which are generally comparable with, but slightly worse than, those of Sheba. Furthermore, the final column of Table 1 (column ‘Kpax-K’) shows that using just one round of DP with the pose-invariant K-scores provides a fast way to calculate a good initial superposition. This shows that the K-score is identifying many equivalent residues for each pair of structures without requiring any iteration or least-squares fitting.

To understand better the differences between the selected alignment methods, Supplementary Table S1 lists for each pair of methods the individual and average C_α RMSD differences between the computed positions of each superposed structure when calculated with respect to a common reference structure. This table shows that, on average, the 3D superpositions produced by Kpax resemble most closely those of TM-Align (with an average C_α RMSD of 1.31 Å), and indeed that the Kpax and TM-Align superpositions are more similar than the superpositions calculated by all other pairs of algorithms (CE/Sheba: 2.77; CE/TM-Align: 2.05; CE/Kpax: 2.21; Sheba/TM-Align: 2.04; Sheba/Kpax 2.70 Å RMSD). The largest individual difference between Kpax and TM-Align is seen in the first pair (PDB codes: 1fxi/1ubq), in which the superposed positions of the ubiquitin domain (1ubq) differ by 3.83 Å RMSD. Figure 5 shows the TM-Align and Kpax overlays for this case. A large colour version of this image is shown in Supplementary Figure S3. Visual inspection of these figures shows that Kpax produces a tighter overlay of the main α-helix and the three large β-strands in this pair than TM-Align.

3.3 Comparing the local and spatial scoring functions

To investigate the strengths and weaknesses of the Kpax scoring functions, we compared the performance of Kpax’s local, spatial and local-plus-spatial scores with TM-Align using six low sequence identity pairs of domains identified previously by Sippl and Wiederstein (2008) and six further pairs from Gerstein and Levitt (1998). Table IV of Mavridis *et al.* (2012) gives some results from CE, SSM, Dali and 3D-Blast for these examples.

Although we consider these pairs to be more challenging than those of Fischer *et al.*, Table 2 shows that both TM-Align and the combined Kpax scoring function find full-length superpositions in all twelve cases (see Supplementary Figs S4 and S5 for images of the 3D superpositions). As in Table 1, Kpax produces alignments with lower RMSDs but also with lower numbers of aligned residues than TM-Align for the majority of cases. In terms of trading between RMSD and number of aligned residues, these results reaffirm the tendency for Kpax to produce

Table 1. Superposition performance comparison for the 10 'difficult' pairs of structures from Fischer *et al.* (1996)[a]

PDB codes	CE[b]	Sheba[c]	TM-Align[d]	Kpax[e]	Kpax-K[f]
1fxiA/1ubqA	64 (2.8)	49 (2.1)	63 (2.6)	45 (2.0)	31 (2.8)
1tenA/3hhrB	87 (1.9)	82 (1.4)	87 (1.8)	84 (1.7)	48 (3.1)
3hlaA/2rheA	85 (3.5)	62 (2.2)	80 (3.1)	66 (2.7)	41 (3.9)
2azaA/1pazA	85 (2.9)	74 (2.0)	86 (2.8)	69 (2.3)	64 (3.5)
1cewI/1molA	81 (2.3)	74 (1.8)	82 (2.3)	69 (1.9)	63 (4.0)
1cidA/2rheA	98 (3.0)	83 (1.8)	100 (2.9)	76 (2.2)	56 (4.2)
1crlA/1edeA	220 (3.9)	139 (2.1)	235 (4.4)	156 (2.5)	93 (4.3)
2simA/1nsbB	297 (3.3)	235 (2.1)	312 (3.8)	255 (2.7)	201 (3.7)
1bgeB/2gmfA	109 (4.6)	81 (2.1)	111 (4.0)	77 (2.7)	68 (3.8)
1tieA/4fgfA	117 (3.0)	93 (1.7)	117 (2.8)	100 (2.4)	84 (3.8)
Average time (s)	1.88	0.48	0.13	0.07	<0.001

[a]Listed are the number of residues aligned by each method, along with the corresponding RMSD in parentheses. Calculation times were measured on a 2.8-GHz quad-core Intel Xeon workstation.
[b]Calculated using jCE (Prlić *et al.*, 2010).
[c]Sheba version 4.0.1 (Jung and Lee, 2000).
[d]TM-Align version 20120126 (Zhang and Skolnick, 2005).
[e]Grid-optimized superposition using the G-score (Equation 11).
[f]Unoptimized superposition obtained directly from the pose-invariant K-score alignment (Equation 9).

Table 2. Structural alignment results for 12 low sequence identity pairs[a]

SCOP domains	Kpax-L[b]	Kpax-S[c]	Kpax[d]	TM-Align	Δ-RMSD[e]
d1euda1/d1ccwa_	79 (2.2)	78 (2.1)	79 (2.2)	103 (3.3)	1.1
d1euda2/d1ccwa_	79 (2.9)	93 (2.4)	94 (2.3)	101 (3.1)	1.3
d1euda1/d1euda2	70 (2.9)	72 (3.1)	72 (3.1)	99 (3.4)	2.4
d1gt8a4/d1mo9a1	132 (1.8)	130 (1.8)	141 (2.2)	165 (3.0)	0.5
1te2B02/1zolA02[f]	60 (2.2)	61 (2.2)	62 (2.1)	67 (2.9)	0.3
d1lt3a_/d1efya2	69 (2.6)	34 (3.4)[g]	63 (2.7)	103 (3.7)	3.2
d1amfa_/d3mbpa_	151 (2.2)	159 (2.9)	161 (2.7)	217 (3.5)	1.5
d1vdca2/d2tmda2	99 (1.5)	103 (1.9)	103 (1.9)	115 (2.3)	0.6
d1sqca1/d1cema_	200 (2.5)	210 (2.5)	210 (2.5)	288 (3.6)	1.3
d1nal1_/d1qbaa3	119 (3.7)	125 (3.6)	132 (3.7)	229 (4.9)	5.0
d1mtyb_/d1ryta1	26 (2.3)[g]	38 (2.0)[g]	100 (2.6)	142 (2.4)	2.9
d4aaha_/d1gofa3	207 (3.3)	197 (3.6)	178 (3.9)	336 (5.4)	11.6

[a]Listed are the number of residues aligned by each method with the corresponding C_α RMSD in parentheses.
[b]Local scoring (weights: $w_l = 1.0$, $w_s = 0.0$) plus grid search.
[c]Spatial scoring (weights: $w_l = 0.0$, $w_s = 1.0$) plus grid search.
[d]Local-plus-spatial scoring ($w_l = 0.5$, $w_s = 0.5$) plus grid search.
[e]The C_α difference in the coordinates of the moving domain between the Kpax and TM-Align superpositions (calculated using ProFit: http://www.bioinf.org.uk/software/profit/).
[f]These are CATH domains.
[g]These are cases in which the expected full length alignment was not found.

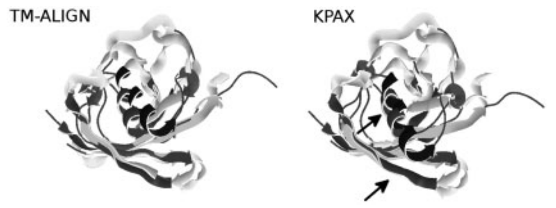

Fig. 5. The structural alignments calculated by TM-Align and Kpax for the pair 1fxiA/1ubqA. Arrows highlight particularly tight regions of the Kpax alignment compared with the TM-Align alignment. SSEs are drawn using Stride assignments. See Supplementary Figure S3 for a large colour version

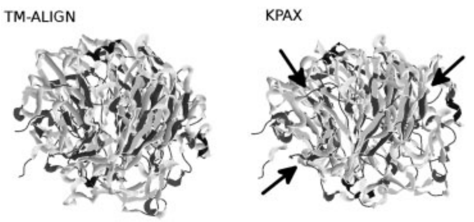

Fig. 6. The structural alignments calculated by TM-Align and Kpax for the pair d4aaha_/d1gofa3. Arrows mark especially tight regions of the Kpax alignment. See Supplementary Figure S6 for a large colour version

superpositions towards the low RMSD, or 'tight', end of the number/RMSD performance metric.

On the other hand, Table 2 also shows that the individual local and spatial functions (tabulated 'Kpax-L' and 'Kpax-S', respectively) both fail to find the full alignment for the pair d1mtyb_/d1ryta1, and the spatial function fails for one further pair (d1lt3a_/d1efya2). The first of these cases involves correctly matching the 4-helix bundle of rubrerythrin (d1ryta1) within the larger 9-helix bundle of the methyl monooxygenase hydroxylase (d1mtyb_). This is clearly difficult for both scoring functions because locally all helices will look the same, and because spatially the COMs of these different-sized domains will not closely coincide in a good overlay. Similarly, for the second pair, the different positions of their COMs with respect to their correctly aligned β-barrels probably explains why the spatial scoring functions fails in this case. Nonetheless (and somewhat surprisingly for the helix bundle case, d1mtyb_/d1ryta1), these examples show that using the combined Kpax score appears to overcome the weaknesses in the individual scoring functions.

The last column of Table 2 shows that, as before, Kpax generally gives quite similar overlays to TM-Align. However, the final example involving a pair of β-propellers (SCOP codes: d4aaha_/d1gofa3) has a large RMSD difference of 11.6 Å between the Kpax and TM-Align superpositions. Figure 6 shows these superpositions graphically, and Supplementary Figure S6 provides a large colour version of the same figure. From close visual inspection of these figures, we believe the tighter Kpax alignment to be superior despite the smaller calculated number of aligned residues.

3.4 CATH database search comparison

As a final and more demanding test, we compared the ability of Kpax, TM-Align and Yakusa to retrieve structural homologues from CATH using a diverse set of structural queries. We chose TM-Align because in our opinion (based on a preliminary comparison of several structural aligners) it is one of the best structure aligners available, and we chose Yakusa because it was specifically designed for rapid database searching. For this test,

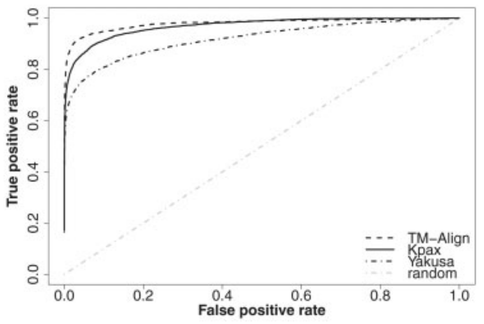

Fig. 7. Aggregate ROC comparison of Kpax, TM-Align and Yakusa using 213 structural queries against the CATH database (version 3.4, comprising 11 330 domains at the 35% sequence identity level)

we selected 213 CATH families for which each family has at least 10 members, and we used CATH's representative structure for each family as the query. These families have an average of 30.7 members, and the query structures have an average of 160 residues. Here, we treat any structure having the same four-digit CATH code as the query as a 'positive' instance, and all other structures as 'negative' instances. We then measured the ability of each algorithm to retrieve structures having the same CATH code as the query by plotting the rate of true positives against the rate of false positives as the ranked list of matching structures is traversed. The area under the curve (AUC) of such receiver-operator-characteristic (ROC) plots may be used as a single objective measure of performance (Fawcett, 2006).

Figure 7 shows the aggregate ROC curves obtained for each algorithm by vertically averaging each set of 213 ROC curves. This figure shows that TM-Align gives the best overall performance, with an average AUC of 0.976, closely followed by Kpax with an AUC of 0.966. Yakusa gives an average AUC of 0.915. Closer examination of the individual curves (not shown here) indicated that the slightly better performance of TM-Align in this test comes from its ability to superpose more distantly related structures. For example, TM-Align achieves AUC >0.99 for 140 of the queries, compared with 102 for Kpax, and just 56 for Yakusa. On the other hand, Kpax is extremely efficient compared with existing structural alignment algorithms. For example, on a 2.8-GHz quad-core workstation, the above calculations took 46 h using TM-Align compared with 2.2 h for Yakusa, and just 22.5 min for Kpax. Furthermore, using only the pose-invariant K-scores without calculating superpositions takes just 13.5 min and gives an almost indistinguishable ROC curve to the superposition search. This corresponds to an average rate of 2980 structural alignments per second. In contrast, from the timing results of a previous study (see Table IV of Mavridis *et al.*, 2012), we estimate that performing the above database searches using CE, SSM, Dali and our own 3D-Blast algorithm would require approximately 286, 454, 1322 and 2012 CPU-hours, respectively. Thus, Kpax offers a useful way to keep ahead of the Red Queen.

Although Kpax allows large domain structure databases such as CATH and SCOP to be searched rapidly, only a single global alignment and superposition is reported for each pair of compared structures. In the future, we would like to use Kpax to search the entire PDB directly. However, if the aim is to detect sub-structure matches e.g. when comparing a single small domain with large multi-domain structures, we expect that the spatial scoring function will be less useful than the local similarity score. Additionally, it will require further work to be able to handle arbitrary PDB files, which might contain multiple structures and conformations, and which would therefore require additional processing to collect and report multiple possible structural alignments.

4 CONCLUSION

We have presented Kpax, a novel protein structure alignment and superposition algorithm that uses multiple C_α coordinate systems and a Gaussian peptide fragment scoring scheme to provide a sensitive structural similarity score. For the pairs of structures studied here, Kpax gives similar alignment statistics to Sheba, and it generally calls fewer aligned residues with lower RMSDs than CE and TM-Align. However, this does not imply low alignment quality. We have shown that the superpositions produced by Kpax resemble more closely those produced by TM-Align than those of CE and Sheba, and we have demonstrated that Kpax produces tighter superpositions of SSEs than TM-Align in several cases.

We have also shown that Kpax may be used to perform fast and sensitive structural database searches. In our comparison with Yakusa and TM-Align using the CATH database, we showed that Kpax is faster and more accurate than the very efficient Yakusa algorithm, and it gives almost the same high level of fold recognition as TM-Align while being more than 100 times faster. Our timing estimates for CE, SSM and Dali predict even greater speed-ups with respect to these algorithms. These results demonstrate that Kpax is both fast and respectably accurate in comparison with the current state of the art. However, it still has some caveats. For example, it produces only a single rigid global alignment for each pair of compared structures, it cannot handle permutations or multi-structure PDB files, and its spatial scoring function is not well suited for comparing protein domains that differ significantly in size. Nonetheless, with the number of solved protein structures growing ever more rapidly, we believe the publicly available Kpax program will provide a useful tool for high throughput comparisons of 3D protein structures.

ACKNOWLEDGEMENTS

We thank the Associate Editor and three anonymous referees for several suggestions for improvements to the presentation of this article.

Funding: Part of this work was funded by the Agence Nationale de la Recherche, grant number ANR-11-MONU-006-02.

Conflict of Interest: none declared.

REFERENCES

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Andonov,R. *et al.* (2008) An efficient Lagrangian relaxation for the contact map overlap problem. *LNCS*, **5251**, 162–173.

Andonov,R. *et al.* (2011) Maximum contact map overlap revisited. *J. Comp. Biol.*, **18**, 27–41.

Berman,H.M. (2008) The protein data bank: a historical perspective. *Acta Crystallogr.*, **A38**, 88–95.

Boys,S.F. (1950) Electronic wave functions I A general method of calculation for the stationary states of any molecular system. *Proc. Roy. Soc.*, **A200**, 542–554.

Budowski-Tal,I. *et al.* (2010) FragBag, an accurate representation of protein structure, retrieves structural neighbours from the entire PDB quickly and accurately. *Proc. Natl Acad. Sci.*, **107**, 3481–3486.

Charpentier,M. *et al.* (2005) YAKUSA: a fast structural database scanning method. *Proteins: Struct. Func. Bioinf.*, **61**, 137–151.

Chen,L. *et al.* (2006) Revealing divergent evolution, identifying circular permutations and detecting active-sites by protein structure comparison. *BMC Struct. Biol.*, **6**, 18.

Chew,L.P. *et al.* (1999) Fast detection of common geometric substructures in proteins. *J. Comp. Biol.*, **6**, 313–325.

Cowtan,K. (2008) Modified phased translation functions and their application to molecular-fragment location. *Acta Crystallogr.*, **D54**, 750–756.

Cuff,A.L. *et al.* (2009) The CATH classification revisited—architectures reviewed and new ways to characterize structural divergence in superfamilies. *Nucleic Acids Res.*, **37**, D310–D314.

Fawcett,T. (2006) An introduction to ROC analysis. *Pat. Recog. Lett.*, **7**, 861–874.

Fischer,D. *et al.* (1996) Assessing the performance of fold recognition methods by means of a comprehensive benchmark. In *Proceedings of the 1st Pacific Symposium on Biocomputing*. World Scientific Publishing Co, Singapore, pp. 300–318.

Frishman,D. and Argos,P. (1995) Knowledge-based protein secondary structure assignment. *Proteins*, **23**, 566–579.

Gerstein,M. and Levitt,M. (1998) Comprehensive assessment of automatic structural alignment against a manual standard, the scop classification of proteins. *Prot. Sci.*, **7**, 445–456.

Gibrat,J.F. *et al.* (1996) Surprising similarities in structure comparison. *Curr. Opin. Struct. Biol.*, **6**, 377–385.

Hasegawa,H. and Holm,L. (2009) Advances and pitfalls of protein structure alignment. *Curr. Opin. Struct. Biol.*, **19**, 341–348.

Holm,L. and Sander,C. (1993) Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, **233**, 123–138.

Holm,L. *et al.* (2008) Seaching protein structure databases with DaliLite v.3. *Bioinformatics*, **24**, 2780–2781.

Ilyin,V.A. *et al.* (2004) Structural alignment of proteins by a novel TOPOFIT method, as a superimposition of common volumes at a topomax point. *Prot. Sci.*, **13**, 1865–1874.

Jung,J. and Lee,B. (2000) Protein structure alignment using environmental profiles. *Protein Eng.*, **13**, 535–543.

Jung,S. *et al.* (2011) Validity of protein structure alignment method based on backbone torsion angles. *J. Proteomics Bioinform.*, **4**, 218–226.

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

Kabsch,W. and Sander,C. (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, **22**, 2577–2637.

Kawabata,T. and Nishikawa,K. (2000) Protein structure comparison using the Markov transition model of evolution. *Proteins*, **41**, 108–122.

Kifer,I. *et al.* (2011) GOSSIP: a method for fast and accurate global alignment of protein structures. *Bioinformatics*, **27**, 925–932.

Kolbeck,B. *et al.* (2006) Connectivity independent protein-structure alignment: a hierarchical approach. *BMC Bioinformatics*, **7**, 510.

Konagurthu,A.S. *et al.* (2008) Structural search and retrieval using a tableau representation of protein folding patterns. *Bioinformatics*, **24**, 645–651.

Krissinel,E. and Henrick,K. (2004) Secondary structure matching (SSM), a new tool for fast protein structure alignment in three dimensions. *Acta Crystallogr.*, **D60**, 2256–2268.

Lackner,P. *et al.* (2000) ProSup: a refined toolf for protein structure alignment. *Protein Eng.*, **13**, 745–752.

Lipman,D.J. and Pearson,W.R. (1985) Rapid and sensitive protein similarity searches. *Science*, **227**, 1435–1441.

Lo,W.C. *et al.* (2007) Protein structural similarity search by Ramachandran codes. *BMC Bioinformatics*, **8**, 307.

Lu,G. (2000) TOP: a new method for protein structure comparisons and similarity searches. *J. Appl. Crystallogr.*, **33**, 176–183.

Malod-Dognin,N. *et al.* (2010) Maximum cliques in protein structure comparison. *Experimental Algorithms LNCS*, **6049**, 106–117.

Mavridis,L. *et al.* (2012) Representing and comparing protein folds and fold families using 3D shape-density representations. *Proteins*, **80**, 530–545.

Murzin,A.G. *et al.* (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, **247**, 536–540.

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

Novotny,M. *et al.* (2004) Evaluation of protein fold comparison servers. *Proteins*, **54**, 260–270.

Ortiz,A.R. *et al.* (2002) MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison. *Prot. Sci.*, **11**, 2606–2621.

Pandit,S.B. and Skolnick,J. (2008) Fr-TM-align: a new protein structural alignment method based on fragment alignments and the TM-score. *BMC Bioinformatics*, **9**, 531.

Prlić,A. *et al.* (2010) Pre-calculated protein structure alignments at the RCSB PDB website. *Bioinformatics*, **26**, 2983–2985.

Razmara,J. *et al.* (2012) TS-AMIR: a topology string alignment method for intensive rapid protein structure comparison. *Algorithms Mol. Biol.*, **7**, 4.

Sabarinathan,R. *et al.* (2010) ProSTRIP: a method to find similar structural repeats in three-dimensional protein structures. *Comput. Biol. Chem.*, **34**, 126–130.

Sacan,A. *et al.* (2008) Integrated search and alignment of protein structures. *Bioinformatics*, **24**, 2872–2879.

Salem,S. *et al.* (2010) FlexSnap: flexible non-sequential protein structurea alignment. *Algorithms Mol. Biol.*, **5**, 12.

Shen,Y.F. *et al.* (2010) Protein structure alignment based on internal coordinates. *Interdiscip. Sci.*, **2**, 308–319.

Shibberu,Y. and Holder,A. (2011) A spectral approach to protein structure alignment. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **8**, 867–875.

Shindyalov,I. and Bourne,P. (1998) Protein structure alignment by incremental combinatiorial extension (CE) of the optimal path. *Protein Eng.*, **11**, 739–747.

Sierk,M.L. and Kleywegt,G.J. (2004) Déjà vu all overa again: finding and analyzing protein structure similarities. *Structure*, **12**, 2103–2111.

Sippl,M. and Wiederstein,M. (2008) A note on difficult structure alignment problems. *Bioinformatics*, **24**, 426–427.

Sippl,M. and Wiederstein,M. (2012) Detection of spatial correlations in protein structures and molecular complexes. *Structure*, **20**, 718–728.

Smith,T.F. and Waterman,M.S. (1981) Identification of molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Stivala,A. *et al.* (2009) Tableau-based protein substructure search using quadratic programming. *BMC Bioinformatics*, **10**, 153.

Szustakowski,J.D. and Weng,Z. (2000) Protein structure alignment using a genetic algorithm. *Proteins*, **38**, 428–440.

Täubig,H. *et al.* (2006) PAST: fast structure-based searching in the PDB. *Nucleic Acids Res.*, **34**, W20–W23.

Taylor,W.R. and Orengo,C.A. (1989) Protein structure alignment. *J. Mol. Biol.*, **208**, 1–22.

Taylor,W.R. *et al.* (2001) Protein structure: geometry, toplogy, and classification. *Rep. Prog. Phys.*, **64**, 517–590.

Tung,C.H. *et al.* (2007) Kappa-alpha plot derived structural alphabet and BLOSUM-like substitution matrix for rapid search of protein structure database. *Genome Biol.*, **8**, R31.

Tyagi,M. *et al.* (2007) Protein structure mining using a structural alphabet. *Proteins*, **11**, 920–937.

Venkateswaran,J.G. *et al.* (2011) Protein structural similarity search by Ramachandran codes. *IEEE Trans. Comput. Biol. Bioinform.*, **8**, 819–831.

Wohlers,I. *et al.* (2010) Towards optimal alignment of protein structure distance matrices. *Bioinformatics*, **26**, 2273–2280.

Yang,J.M. and Tung,C.H. (2006) Protein structure database search and evolutionary classification. *Nucleic Acids Res.*, **34**, 3646–3659.

Ye,J. *et al.* (2004) Pairwise protein structure alignment based on an orientation-independent backbone representation. *J. Bioinform. Comput. Biol.*, **2**, 699–717.

Ye,T. and Godzik,A. (2003) Flexible structure alignment by chained aligned fragment pairs allowing twists. *Bioinformatics*, **19** (**Suppl. 2**), ii246–ii255.

Zemla,A. (2003) LGA a method for finding 3D similarities in protein structures. *Nucleic Acids Res.*, **31**, 3370–3374.

Zhang,Y. and Skolnick,J. (2005) TM-align: a protein structure alignment algorithm based on TM-score. *Nucleic Acids Res.*, **33**, 2302–2309.

Zhang,Z.H. *et al.* (2010) deconSTRUCT: general purpose protein database search on the substructural level. *Nucleic Acids Res.*, **38**, W590–W594.

Zhou,L.C.T. and Tang,Y. (2005) Protein structure alignment by deterministic annealing. *Bioinformatics*, **21**, 51–62.

Zhu,J. and Weng,Z. (2005) FAST: a novel protein structure alignment algorithm. *Proteins*, **58**, 618–627.