ALIGNMENT-BASED EXTENSION TO DDPIN FEATURE EXTRACTION

David Hoksza* and Jakub Galgonek*

Abstract

Finding similarity between a pair of protein structures is one of the fundamental tasks in many areas of bioinformatical research such as protein structure prediction, function mapping, etc. We propose a method for finding pairing of amino acids based on densities of the structures and we also propose a modification to the original template modeling-score (TM-Score) rotation algorithm that assess similarity score to this alignment. Proposed modification is faster than TM and comparably robust according to non-optimal parts in the alignment. We measure the qualities of the algorithm in terms of structural classification of proteins (SCOP) classification accuracy. Regarding the accuracy, our solution outperforms the contemporary solutions at two out of three tested levels of the SCOP hierarchy.

Key Words

Protein structure, similarity, SCOP classification, TM-score

1. Introduction

Proteins and their interactions are crucial for every living organism. To be able to understand protein interactions and their evolution, the study of protein structures is inevitable. In face of large number of determined protein structures in the protein data bank (PDB) [1] a need for certain form of organization of these structures emerged. From all the classifications, manually curated hierarchical evolutionary classification SCOP [2] was established as the gold standard for organizing protein structures. The hierarchy contains four levels – family, superfamily, fold and class. Proteins in the same family can have high sequence similarity (>30%) or lower sequence similarity (>15%) with very similar function or structure. Proteins sharing common evolutionary origin (based on structural and functional features) but differing in sequence reside in the same superfamily. Structures having same major secondary structures in similar topological distribution are in the same fold. And finally, similar folds are grouped into classes.

In the face of growing size of PDB, there became a need to automate the work of a human expert, hence being able to assess a correct classification to a newly discovered protein automatically. This task facilitates the process of determination of protein function (which is grossly determined by the structure) since establishing a similarity class gives a hint to the function of the protein.

Algorithms solving structural similarity problem usually describe protein structures by a set of features based on 3D distances of proteins' amino acids (and sometimes other qualities such as amino acid burial, solvent exposure, hydrophobicity, secondary structure elements – SSE, etc.). Purpose of these features is to chracterize the protein structure as closely as possible and use them to describe similarity of a pair of proteins in terms of partial similarities of the amino acids (hence features by which they are represented). Since protein structures do not have a fixed coordinate frame, features (that are independent on the absolute position in space) are usually used to find matching pairs of amino acids in the respective proteins. Given the alignment, an algorithm is used to rotate and translate one of the structures to optimally fit the other. Here, the optimality is expressed in the sense of a distance function (various methods can use various distance functions) that has to be minimized or maximized. It has been shown that finding an optimal pairing of amino acids is NP-hard [3].

In this paper, we approach both problems – we propose an algorithm for pairing amino acids (Sections 2.1, 2.2) and this pairing is forwarded into an improved algorithm for computing template modelling-score (TM-Score) (Section 2.3).

Let's revisit few well-known algorithms used for comparison of protein structures. Distance alignment matrix method (DALI) [4], one of the first methods to compare protein structures, employs matrix of inter-residual distances and uses its parts to find similar substructures that are utilized for finding the alignment by Monte Carlo algorithm. Combinatorial extension (CE) [5] utilizes aligned fragment pairs (AFP) to describe pairs of consecutive residues in both structures sharing substantial structural similarity and these pairs are then connected to obtain the alignment. Another well-known example is sequential structure alignment program (SSAP) [6] which utilizes dynamic programming (DP) to assess similarity to pairs of amino acids which are represented by so-called views (vector of distances to all the other amino acids). Optimal

* Department of Software Engineering, Charles University in Prague, Faculty of Mathematics and Physics, Prague, Czech Republic; e-mail: {david.hoksza, jakub.galgonek}@mff.cuni.cz

Recommended by Dr. L. R. Welch
(10.2316/J.2010.210-1023)

paths in the individual DP matrices are used to fill in DP matrix at second level that outputs the final alignment.

Now, we will briefly describe newer algorithms to which we will compare our solution of the problem. The most well-known algorithm is probably TM-align [7], since it is the solution with which the TM-score [8] is usually presented. TM-score is considered as the standard for evaluating similarity of two structures given an alignment (see Section 1.1). TM-Align uses three initial alignments achieved by DP based on SSEs and/or distances of C_α amino acids. For these alignments TM-score rotation matrix is computed and used as the basis for scoring matrix for further iterative steps of the DP. Vorolign [9] uses global DP solution where scoring matrix is based on features obtained from Voronoi tessellation. The same group of authors presented later a solution called phenotypic plasticity measure (PPM) [10]. PPM identifies core blocks (blocks in the two structures sufficiently similar) which are then used to create a graph of core blocks. The path in the graph that minimizes the cost of mutation is chosen. Finally, the most recently presented solution is Vorometric [11]. Vorometric achieves in general the highest classification accuracy and (similarly to Vorolign) employs Voronoi contacts enriched with SSE information to obtain a metric scoring matrix enabling indexing of global DP computations, thus highly increasing classification speed.

1.1 Protein Structure Similarity Measures

Probably the most often employed measure is the root mean square deviation (RMSD). Given the alignment, the optimal superposition minimizing RMSD distance can be found in polynomial time by *Kabsch* algorithm [12]. In RMSD, distances (after translation) among paired amino acids are computed and then aggregated to get the distance:

$$\text{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i)^2} \qquad (1)$$

N stands for the length of the alignment and d_i for the distance between ith pair of aligned amino acids.

But such a distance could be easily influenced by outliers (partial distances being far from average) and is not sensitive to highly conserved local substructures. To handle these issues, TM-score [8] was proposed which puts emphasis on closer pairs by weighting them higher. The superposition is obtained by its own algorithm exploiting *Kabsch* (see Section 2.3 for details and optimizations). TM-score value (to evaluate quality of the superposition) is defined as:

$$\text{TM-score} = \frac{1}{L_T}\sum_{i=1}^{L_A}\frac{1}{1+\left(\frac{d_i}{d_0}\right)^2} \qquad (2)$$

Let's assume that TM-score is used for classification purposes where an unknown structure (target) is being classified. Then L_T stands for length of this structure, L_A is length of the alignment, d_i distance between ith pair of aligned amino acids and d_0 is a normalization parameter (see [8] for details).

2. Methods

In this section we describe our approach to protein structure alignment which consists of finding best possible alignment of amino acids based on the density of the residues. Afterwards we revisit the TM-score algorithm for finding optimal superposition based on given alignment presented in [8].

The alignment is the crucial component of the process of superposing protein structures. Given a meaningless alignment, no method is able to provide a good superposition. Since the problem is computationally very expensive, only heuristics are employed to find the alignment. Our proposed heuristic understands individual amino acids as viewpoints from which the rest of the protein is viewed. Then we pair similar viewpoints from respective protein structures with the help of local or global dynamic programming.

2.1 Protein Structure Representation

Our protein structure representation is based on distances and density of the amino acids (their C_α atoms) that we presented in [13]. A protein structure (of size n) is represented by a set of n feature vectors each of them describing the neighborhood of an individual amino acid. We present several semantics of feature vectors based on density of amino acids in nested 3D rings with the center in the amino acid from which the protein is viewed (see Fig. 1). Based on widths or perimeters of those rings, feature vectors are extracted which we call *viewpoint tags* (VPT) since they are blueprints of the protein according to given amino acids.

For the following alignment step, each amino acid in both proteins to be superposed is converted into feature vector (VPT) based on one of the semantics. The VPTs are then used for forming the alignment.

The description of the VPT semantics follows.

Let vp represent a particular viewpoint, then $vp[i]$ stands for the ith ring, $rad(vp[i])$ for the distance from the viewpoint to the further edge of the ith ring, $width(vp[i]) = rad(vp[i]) - rad(vp[i-1])$ $(width(vp[0]) = rad(vp[0]))$ and let $dens(vp[i])$ be the density (sum) of the residues in the ith ring (see Fig. 1). Finally, let $VPT[i]$ be the ith coordinate of the feature vector (viewpoint tag). Based on these terms, we propose several VPT semantics.

- ***sRad***: For $sRad$ (radius based semantics) it holds:
  - $\forall i, j$: $dens(vp[i]) = dens(vp[j]) = p$
  - $\forall i$: $VPT[i] = rad(vp[i])$

where p is a user-defined parameter representing percentage of amino acids in the protein.

- ***sDens***: For $sDens$ (density based semantics) it holds:
  - $\forall i, j$: $width(vp[i]) = width(vp[j]) = w$
  - $\forall i$: $VPT[i] = dens(vp[i])$

where w is a user-defined parameter representing width of the rings in Å.

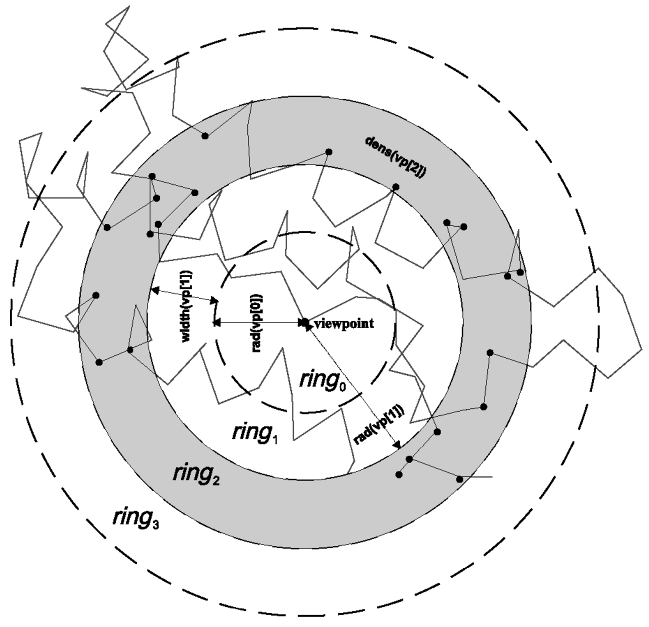


Figure 1. Visualization of a protein with PDB ID *1apc* in 2D. Vertices on the curve correspond to C_α residues of individual amino acids. $dens(vp[2])$ equals number of dots in $ring_2$.

- ***sRadSSE, sDensSSE***: $s{*}SSE$ is identical to $s*$ except for semantics of vp which slightly differs. Only residues belonging into an α helix or a β sheet are taken into account when defining viewpoints. Moreover, residues from distinct SSE types are stored separately. Hence dimension of the VPT increases twice. ith ring is represented by $VPT[2i]$ (α type residues) and $VPT[2i+1]$ (β type residues). $VPT[i]$ is defined the same as in the $sRad$ ($sDens$) VPT semantics.
- ***sDir***: For $sDir$ (direction based semantics) it holds:
  - $\forall i,j$: $width(vp[i]) = width(vp[j]) = w$
  - $\forall i$: $VPT[i] = \sum(pairs\ of\ consequent\ residues\ in\ the\ ith\ ring\ with\ the\ orientation\ from\ the\ vp)$

where w is a user-defined parameter representing width of the rings in Å. $sDir$ semantics aims to detect shape of the curve within the bounds of the density/distance approach.

We utilize weighted euclidean distance for VPTs comparison. Weighting is used for emphasizing the fact that for assessing similarity to a pair of viewpoints, their close neighborhood is more important than the more distant one. Hence, our weighting scheme favors the close neighborhood by putting more weight on the first few coordinates. Specifically, for ith coordinates of the feature vectors, we define weighting scheme w as $w(i) = n - log(i)$.

2.2 Finding an Alignment

Using VPTs and distance functions from Section 2.1, we apply DP to find optimal pairs of amino acids (similar viewpoints) following sequence order. This solution can resemble sequential structure alignment program (SSAP) [6], but despite SSAP we do not need to accomplish two levels of DP, since our viewpoints based on VPTs substitute the first level of DP. We compute distances between all pairs of VPTs in the given proteins and these are stored in a matrix that is subsequently used as scoring matrix for the DP phase. Moreover, in contrast to SSAP we use variable gap costs and modified TM-score for scoring.

2.2.1 Needleman–Wunsch (Global Alignment)

Needleman and Wunsch (NW) [14] is a DP programming algorithm that was originally invented to optimally align two protein sequences given a substitution (distance) matrix (expressing similarity of pairs of amino acids). The core of the algorithm is a recursive function deciding whether ith and jth letters should be aligned or whether a gap should be introduced into one of the sequences. The decision is based on the $S[i,j]$ and on the already computed values in the DP matrix. The value of the optimal alignment is finally stored in the lower right corner of the DP matrix and the alignment can be determined by a backtracking procedure starting at that point. To penalize gaps in the alignment, the gap price (σ) is introduced. An extension to NW algorithm allows to penalize differently start and continuation of a gap which we utilize in our algorithm. Originally, NW was proposed as a maximization problem where similar amino acids scored higher and gap penalties were usually negative. To use NW with VPTs we replace the max with min function because our substitution matrix S contains lower values for similar pairs of viewpoints, hence optimal alignment is that one having minimal overall score.

2.2.2 Smith–Waterman (Local Alignment)

To be able to concentrate more on conserved parts of the relative protein structures we also use Smith and Waterman (SW) algorithm [15]. SW was also proposed for protein sequences to find highly similar (conserved) subsequences. The DP recursion is very similar to NW – a parameter is added to the max function to recognize fields in the DP matrix where it does not make sense to continue an alignment and at which point a new one should start – this value is usually 0. Hence, if at a position the DP score drops under 0, no alignment crossing this point could possibly achieve higher score than alignment starting at that point. Moreover, score of the alignment is not found at the lower right position in the DP matrix anymore, instead it resides at the position with the highest score. That position also represents end of the local alignment (its starting point is the position with the first zero while backtracking from that point).

Using SW with our algorithm is not so straightforward as using NW. If minimization would be the only change to the algorithm then, because of adding zero (minimum value ensuring locality of the alignment) to the recursive formula, in each alignment of length at least one would score higher than zero and hence would be impossible to achieve. To be able to utilize local alignment we keep the maximization but modify the substitution matrix. At first we modify the cells so that lower values imply lower similarity by $S[i,j] = c/S[i,j]$ (c being a constant value). Moreover, median μ is computed from the modified values and $3/4 * \mu$ (empirically determined) is subtracted from each value in S. So we introduce negative values into the scoring matrix inevitable for local alignment to behave correctly.

2.2.3 Gap Costs

Both types of alignment allow using different costs for opening a gap and extending an already opened gap. In all algorithms, we are aware of, this price is constant. But since the substitution matrix is not constant in our case (as it usually is for sequence alignment algorithms), the gap costs should not be constant either. For this reason, we employ variable gap costs (varying for each single pair of structures) that was empirically determined as $OGP = \mu$ for open gap penalty and $EGP = \mu/2$ for extend gap penalty.

2.3 Scoring

As stated earlier, the quality of the final superposition is strongly dependent on the initial alignment that is passed to the superposition method. But this method itself can be robust regarding non-optimalities in the alignment. For example with the same set of alignments we are able to achieve higher classification accuracy with TM-score than with RMSD. It is not only the consequence of the formula itself but also the transformation procedure that superposes the structures (to this superposition the formula is applied). Hence, we not only use the TM-score formula but we also improve the rotation procedure (by adding iterative steps of DP) to increase the robustness of the TM-score as a whole. Moreover, we propose a modification to increase the speed of the algorithm causing only a small or no decrease of the classification accuracy.

2.3.1 Reducing the Number of Initial States of TM-Score

One of the problems of the TM-score is the absence of a fast algorithm for computing the superposition of the alignment. The algorithm presented in [8] is a relatively slow heuristics. Its main idea lies in finding such a *cut* (subset) of the input alignment whose RMSD superposition maximizes the TM-score formula.[1] The algorithm uses various initial cuts of the alignment. For each, its RMSD superposition and TM-score according to this superposition are computed and a new cut of the alignment is created. The new cut includes those pairs of the alignment that are spatially near in the superposition. The process is repeated until stabilization of the cuts (two subsequent cuts are identical) or maximum number of iterations (typically 20) is achieved. In the end, from all the superpositions that one maximizing TM-score is returned. Hence, the main factor influencing speed of the algorithm is the number of initial cuts. The algorithm examines initial cuts of lengths L, $L/2 \ldots max(L/2^5, 4)$, L being length of the input alignment. For each length, all possible continuous cuts of given length are taken.

[1] Superposition of the cut is then applied to the original alignment.

To speed up the algorithm, we reduce number of the initial cuts. In an extreme variant (called *FAST*), we use only one initial cut that includes the whole (initial) alignment (with such an approach, quality of the initial alignment and similarity of the proteins should be considerably high). In other variant (called *FAST_SSE*), we take the whole alignment as in the previous variant and its continuous cuts having pairs coming from identical SSEs. The reason for considering those cuts is based on the assumption that cuts with this property should be included in the optimal cut of the initial alignment.

In general, FAST and FAST_SSE approximations show worse results (lower value of the TM-score) than the full TM-score algorithm. However, the level of decrease of the TM-score value is dependent on similarity of the proteins (see Fig. 2). For proteins being significantly similar full TM, FAST and FAST_SSE give comparable results. The vertical line in Fig. 2 shows the average value of TM-score similarity from the query to the most similar database protein (the exact value is 0.72 with 0.11 being the standard deviation). These proteins are important for us since according to them the classification is carried out and for these proteins the difference between original TM and FAST* modifications is very small.

To classify a query protein, we find the most similar protein in the database. As stated in preceding paragraph, for structurally similar proteins full TM and FAST* variants differ only slightly. We employ this quality and use FAST* heuristics as a prestep since they are much faster than the full TM (see Sections 3.3 and 3.1.1). The database is scanned with FAST* heuristics and then one of the following methods is applied to pick up database proteins (candidate set) that are to be aligned to the query with the full TM-score method:

- *top kNN (k nearest neighbors) fitraltion* – k most similar proteins to the query are forwarded into the next stage where they are ordered according to the full TM-score
- *range filtration* – only proteins in a given distance from the most similar protein are further examined[2]

2.3.2 Iterations of TM-Score

One of the major qualities of the TM-score formula lies in concentrating on strong local structural similarities. Hence, an alignment with highly similar regions shows high TM-score. On the other hand, a superposition looking optically well (Fig. 3a[3]) can obtain lower TM-score than it should according to the look. In such case, correction of the alignment by DP should increase the TM-score value. Similarly to [17], we employ NW with scoring matrix S, defined as:

$$S[i,j] = \frac{1}{1 + \left(\frac{d_{ij}}{d_0}\right)^2} \tag{3}$$

[2] Such an apporach is more effective than the more common method where proteins in a given distance from the query are considered since it is more query-specific.

[3] Image generated by VMD [16].

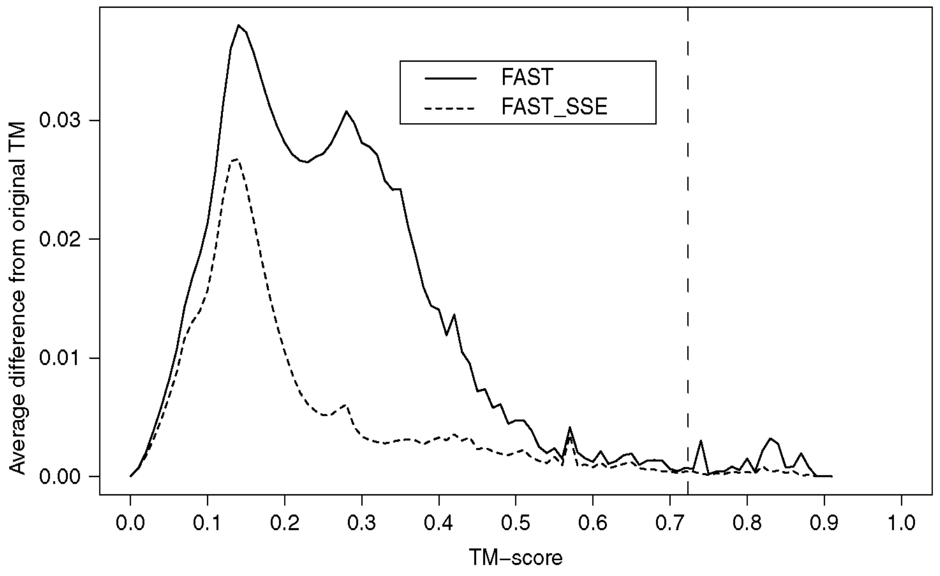


Figure 2. The average difference of FAST and FAST_SSE modifications according to the full TM-score (y-axis shows the average decrease of the score).

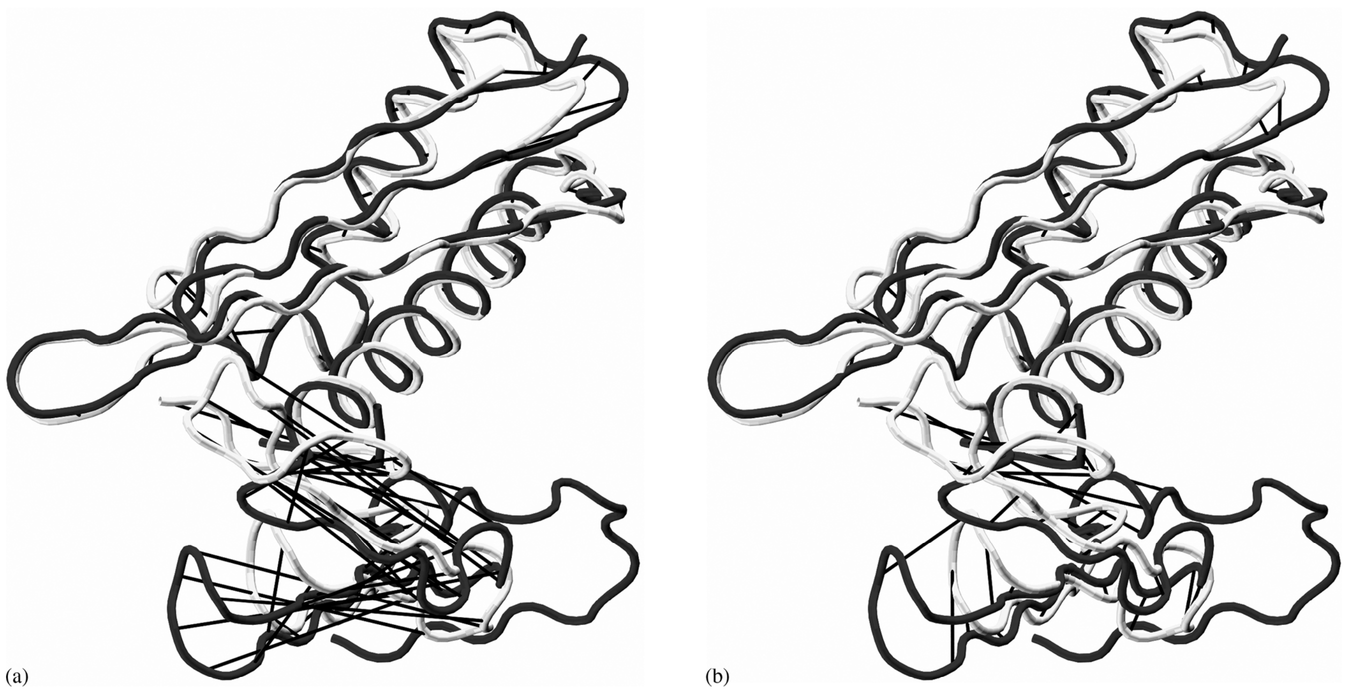


Figure 3. Superposition and alignment (aligned residues connected by black bars) of the proteins d1nyef_ and d1n2fa_ before (a) an after and (b) iterative DP (with TM-scores 0.56553 and 0.70329).

d_{ij} being the euclidean distance of the ith and jth residues of the proteins and d_0 the normalization parameter.

The optimal path through the DP matrix represents the alignment having best TM-score value according to the given superposition. We modify NW to increase speed and to avoid extensive modifications in the alignment by considering only an area (belt) in the DP matrix with a constant width going along the original alignment (we set width of the belt to 33 if not stated otherwise). Based on the newly obtained alignment, the whole process of computing TM-score superposition can be iteratively repeated (we run two iterations of DP). Superposition and alignment after two DP iterations demonstrates Fig. 3b (black bars represent the alignment).

3. Experimental Evaluation

To evaluate our method and to compare it with other algorithms, we employ simple classification accuracy (CA – ratio of proteins classified into the correct SCOP "family") and precision-recall curves in our experiments. If not otherwise stated, our algorithm uses $sDens$ semantics

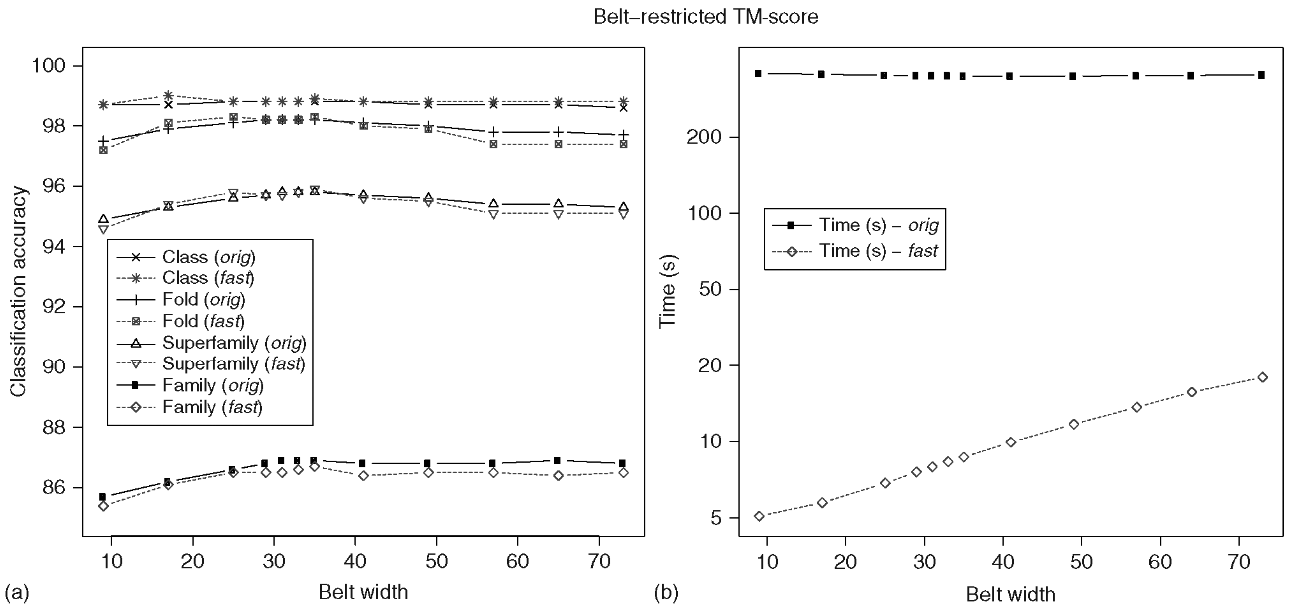


Figure 4. The influence of the belt width on the CA (a) and runtime and (b) for belt-based restriction (*orig* = original TM, *fast* = FAST modification).

(eight rings each having width 3 Å) to extract feature vectors.[4] The machine running experiments was 2.4 GHz $4 \times$ Six-Core AMD Opteron CPU, 96 GB RAM with Red Hat Linux Server 5.4.

Using dataset originally presented in [9] (also used in [10, 11]) (CA of other methods is acquired from these papers) allows us to compare our method to the best contemporary solutions. This dataset contains 979 test proteins present in version 1.67 of ASTRAL compendium [17] (based on SCOP classification) but not present in version 1.65. Test proteins were scanned against ASTRAL25 (each pair of proteins sharing no more than 25% sequence identity) version 1.65 database containing 4,357 objects, evaluating number of correctly classified proteins into family, superfamily and fold according to the SCOP classification.

3.1 Classification Accuracy (CA)

As stated earlier, a protein is classified correctly if a method classifies identically as SCOP does on a given level (family, superfamily, class). CA is an aggregation measure expressing quality of a method by summing the correct classifications and dividing them by the number of proteins to be classified (ratio of correctly classified proteins).

3.1.1 Belt-based Restriction Evaluation

In Section 2.3.2, we described how we employ belt for restriction of the space of the DP matrix in which the alignment's improvement takes place. Hence, the new alignment, for which TM-score algorithm is computed, lies within a belt defined by the previous alignment. The advantage of the belt should be twofold – higher CA (the new aligment sticks closer to the previous one and

Table 1
Classification Accuracy on SCOP Levels

	Family	Superfamily	Fold
db-iTM	86.6	**95.8**	**98.2**
db-iTM$_{orig}$	86.9	**95.8**	**98.2**
db-TM$_{orig}$	85.4	93.4	96.7
Vorometric-TM	**90.7**	94.9	97.6
PPM	88.3	94.5	97.5
Vorolign	86.4	92.4	97.7
TM-align	83.8	92.6	95.9
CE	84.6	91.9	94.1
BLAST	48.9	52.5	52.8

thus it relies more heavily on the initial alignment) and lower runtime (less time is spent in the DP). Figure 4 presents the results. Figure 4a shows how width of the belt influences the CA. We can see that the optimal belt witdh is about 33 where superfamily CA reaches 95.8% (41 wrong classifications out of 979). On family level, original TM-score outperforms the FAST heuristics but on other levels, both versions are comparable. Here we must stress out that although CA of both methods is almost equal, they do not return identical results. There are few proteins which are identified correctly with original TM and incorrectly with FAST TM and vice versa. Precision–recall experiments in Section 3.2 indicate that the overall quality of original TM is slightly better than that of FAST heuristics. On the other hand, FAST version is much faster as Fig. 4b demonstrates.[5] Original version of TM takes

[4] The values or parameters of the semantics are based on results in [13].

[5] The time values do not include 50 s for alignment computation.

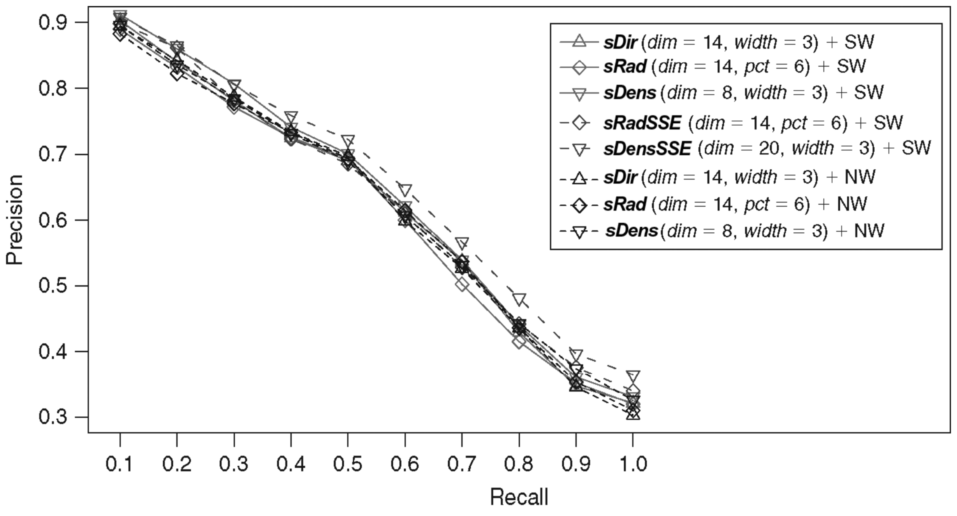


Figure 5. Precision–recall curves for various VPT semantics ($dim =$ number of rings, $width =$ width of a ring, $pct =$ percentage of amino acids in a ring).

about 365 s per classification whilst FAST modification runs only 8.6 s for belt width 33. Speed evaluation offers an interesting observation – with descreasing width of the belt the runtime of original version increases but that of FAST version decreases or stays constant. This can be explained by analysing components of the runtime. The main component of the runtime of original TM is the TM transformation procedure itself, not the DP which takes only about 7 s of the overall time. With small belt width the DP is run more often, hence more time is spent in the TM procedure which consumes most of the runtime. On the other hand, in FAST version the TM takes only 1 s and hence the main determinant is the DP, which is restricted by the belt.

3.1.2 Comparison of CA with Other Methods

Methods to be compared to our solution include BLAST [18] (used with its standard settings), CE [5], PPM [10], TM-align [7], Vorolign [9] and Vorometric [11] (its TM variant applying TM-score superposition).

In Table 1, db-iTM represents our method with fast iterative TM (TM_FAST). Original TM score rotation algorithm (exploiting *sDens* semantics) is marked as db-TM$_{orig}$ and db-iTM$_{orig}$ is its iterative version. The results demonstrate superiority of db-iTM$_{orig}$ and db-iTM in superfamily and fold CA. On family level, Vorometric, PPM as well as Vorolign outperform our solution which is probably caused by not taking sequence into account at all (as stated earlier, SCOP families are largely determined by sequence identity). We can observe poor results of BLAST (exclusively sequence-based method) which can be definitely attributed to the level of difficulty of the test set aiming at low sequence similar proteins.

We believe that the accuracy of our method lies in quality of the initial alignment and in iterative use of the TM-score (where using the belt further increases the accuracy by concentrating on polishing the found alignment instead of seeking for other possibilities). Other methods, such as Vorolign and Vorometric, use TM-score formula for the evaluation of their superpositions but these superpositions are not achieved with the help of the TM-score algorithm but with Kabsch (RMSD) algorithm. Avoiding the TM-score transformation algorithm is understandable since its runtime in the original version is very high which we investigate in further section.

3.2 Precision–Recall Experiments

More practical for real is a method returning multiple results/proteins (instead of the classification only) which can be manually revisited. Among those returned answers should of course prevail correct classifications. Information retrieval (IR) field offers utility to evaluate quality of the method according to this requirement – precision–recall curves. In our case, precision is defined as *correctly classified proteins/retrieved proteins* and recall as *correctly classified proteins/proteins in the query's "family"*.

In Fig. 5[6] precision-recall curves are shown belonging to *sRad(SSE)*, *sDens(SSE)* and *sDir* semantics using db-iTM and not using prefiltration (for non-SSE semantics, results for both local and global algorithms to obtain the initial alignments are shown). At most of the recall levels, *sDensSSE* semantics dominates but on low recall levels the dominate semantics is *sDens* which is why we use it in the CA experiments. We can also see that using local DP (SW) is more suitable than using global one (NW).

3.3 Prefiltration Evaluation

The above experiments show that original *db-iTM*$_{orig}$ and *db-iTM* (FAST) methods provide almost identical results except the fact that FAST is much faster then the original TM (*db-iTM*$_{orig}$ takes 363 s whereas *db-iTM* takes only 8.4 s). But there is still a need for original TM – the overall results of full TM might be more valuable as Fig. 6a suggests. The set of top scored proteins contains more

[6] Only our solution is presented here since we were not able to obtain source codes of the other methods.

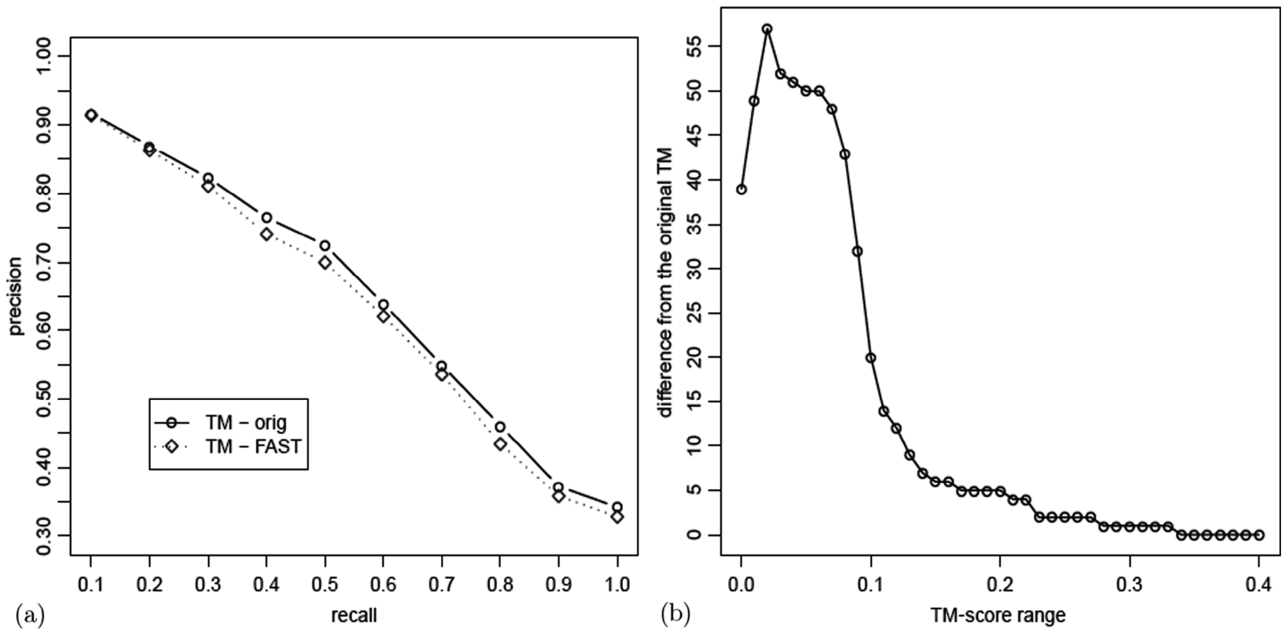


Figure 6. (a) Comparison of PR curves of orginial TM and FAST modification ($orig =$ original TM, $fast =$ FAST modification) and (b) Influence of the prefiltration range on the difference of original and prefiltred TM.

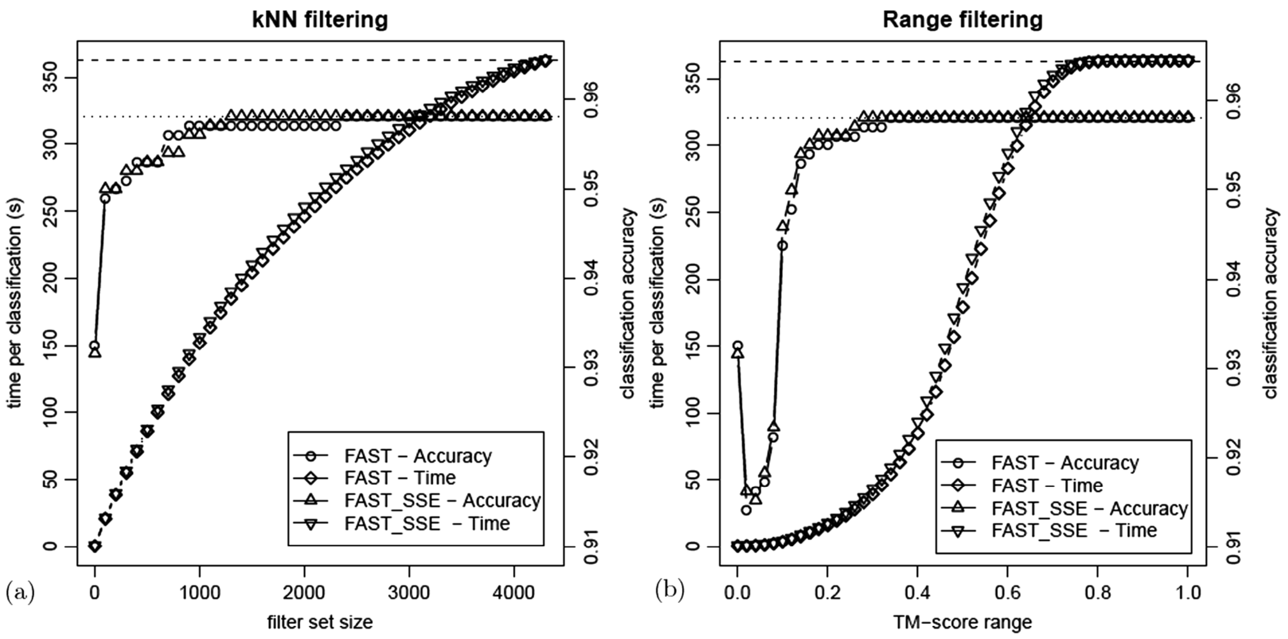


Figure 7. Impact of the prefiltration candidate set size on the runtime and CA (the upper dashed line represents runtime of the full TM version, the lower dotted line represents its CA).

proteins from the query's superfamily in case of original TM than when using the FAST variant.

To obtain original TM results in lower time, we propsed using prefiltration (see Section 2.3.1 – fullscan is carried out with non-iterative FAST TM, which takes only 0.4 s, and then the proteins in a given range from the top scored protein are rescored with original TM). Figure 6b shows what is the minimal range to be applied to the results of $db\text{-}TM$ to obtain identical results as $db\text{-}iTM_{orig}$ does (not only identical CA but also the same query proteins are classified identically in original and FAST version).

In Fig. 7, both possible type of filtering (kNN and range) are presented. Figure 7a shows that for filtration set size of about 1,300 (hence for the whole DB $db\text{-}TM$ is computed and 1,300 top proteins are verified by $db\text{-}iTM_{orig}$) the accuracy is identical to $db\text{-}iTM_{orig}$ (if prefiltred by FAST_SSE. in case of FAST the threshold is 2300). For such a set size the runtime is 179s, hence less than 50% of the runtime of the full version. Using range prefiltration, the results are even better since when CA of the prefiltration method hits the accuracy of full TM the runtime is 37 s in case of using TM_FAST_SSE for prefiltration and 46 s using TM_FAST. Hence, we are able to achieve accuracy of the full TM method in only 10% of its time. Moreover the range threshold from which prefiltration and original TM give identical results is 0.34 as Fig. 6b demonstrates. The range that is needed to hit the CA of original TM when prefiltering by *db-TM* is higher, hence we not only achieve the same CA but we also obtain identical results (structure of the result).

4. Conclusion

We proposed a novel algorithm aimed at classification of protein structures. The novelty of the method lies in using density-based representation of protein structures together with DP. Resulting alignment is forwarded to TM-score rotation procedure. Based on the result, the alignment is repaired by global dynamic programming with belt bounding the alignment. This modification increases both speed and accuracy. We also proposed modification of the TM-score rotation procedure highly increasing the speed of the algorithm causing only slight deterioration of the accuracy. Finally we enhanced the method with prefiltration step that leads to a fast method outperforming other algorithms at the superfamily and fold level.[7]

Acknowledgements

This research has been supported in part by Czech Science Foundation (GAČR) project No. 201/09/0683, by institutional research plan number MSM0021620838, and by the grant SVV-2010-261312.

References

[1] H.M. Berman, J.D. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, & P.E. Bourne, The protein data bank, *Nucleic Acids Research*, *28*(1), 2000, 235–242.

[2] A.G. Murzin, S.E. Brenner, T. Hubbard, & C. Chothia, Scop: A structural classification of proteins database for the investigation of sequences and structures, *Journal of Molecular Biology*, *247*, 1995, 536–540.

[3] R.H. Lathrop, The protein threading problem with sequence amino acid interaction preferences is np-complete. *Protein Engineering*, *7*(9), 1994, 1059–1068.

[4] L. Holm & C. Sander, Protein structure comparison by alignment of distance matrices, *Journal of Molecular Biology*, *233*(1), 1993, 123–138.

[5] I.N. Shindyalov & P.E. Bourne, Protein structure alignment by incremental combinatorial extension (CE) of the optimal path, *Protein Engineering*, *11*(9), 1998, 739–747.

[6] W.R. Taylor & C.A. Orengo, A holistic approach to protein structure alignment, *Protein Engineering*, *7*(2), 1989, 505–519.

[7] Y. Zhang & J. Skolnick, TM-align: A protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research*, *33*(7), 2005, 2302–2309.

[8] Y. Zhang & J. Skolnick, Scoring function for automated assessment of protein structure template quality, *Proteins: Structure, Function, and Bioinformatics*, *57*(4), 2004, 702–710.

[9] F. Birzele, J.E. Gewehr, G. Csaba, & R. Zimmer, Vorolign – Fast structural alignment using Voronoi contacts, *Bio in formatics*, *23*(2), 2007, 205–211.

[10] G. Csaba, F. Birzele, & R. Zimmer, Protein structure alignment considering phenotypic plasticity, *Bioinformatics*, *24*(16), 2008, i98–i104.

[7] Preliminary research on topic of further speed-up has already been discussed in [19].

[11] A. Sacan, I.H. Toroslu, & H. Ferhatosmanoglu, Integrated search and alignment of protein structures, *Bioinformatics*, *24*(24), 2008, 2872–2879.

[12] W. Kabsch, A solution for the best rotation to relate two sets of vectors, *Acta Crystallographica* Section A, *32*(5), 1976, 922–923.

[13] D. Hoksza, DDPIn – Distance and Density Based Protein Indexing, in *CIBCB*. (IEEE, 2009, 263–270).

[14] S.B. Needleman & C.D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology*, *48*(3), 1970, 443–453.

[15] T.F. Smith & M.S. Waterman, Identification of common molecular subsequences, *Journal of Molecular Biology*, *147*(1), 1981, 195–197.

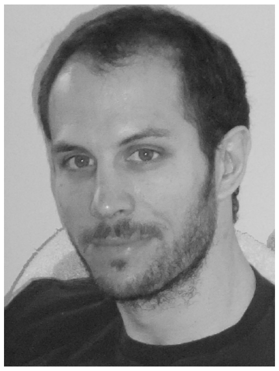
[16] W. Humphrey, A. Dalke, & K. Schulten, VMD: Visual molecular dynamics, *Journal of Molecular Graphics*, *14*(1), 1996, 33–38.

[17] J.M. Chandonia, G. Hon, N.S. Walker, L.L. Conte, P. Koehl, M. Levit, & S.E. Brenner, The ASTRAL compendium in 2004, *Nucleic Acids Research*, *32*, 2004, D189–D192.

[18] S.F. Altschul, T.L. Madden, A.A. Schffer, R.A. Schffer, J. Zhang, Z. Zhang, W. Miller, & D.J. Lipman, Gapped blast and psi-blast: A new generation of protein databas search programs, *Nucleic Acids Res*, *25*, 1997, 3389–3402.

[19] J. Galgonek & D. Hoksza, On the effectiveness of distances measuring protein structure similarity, in *SISAP*, (IEEE, 2009).

Biographies



David Hoksza obtained his M.Sc. degree in Computer Science from Charles University in Prague, Czech Republic (2006). He began his research in the area of database research focusing on effective indexing structures. Then, he switched to bio informatics where he interests in the area of automatic protein classification and other database-related problems. Currently, he is a professor at the Czech Technical University in Prague and is finishing his Ph.D. studies at Charles University in Prague.

Jakub Galgonek obtained his M.Sc. degree in Computer Science from the Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic (2008). During his study of computer science, he became interested in molecular biology which he studied at the Faculty of Science of Charles University. Now he interests in protein databases with focus on the properties of protein similarity measures and is working on his Ph.D. studies at the Charles University in Prague and is professor at The University of Finance and Administration.