Density-Based Classification of Protein Structures Using Iterative TM-score

David Hoksza, Jakub Galgonek
Department of Software Engineering
Charles University in Prague, Faculty of Mathematics and Physics
Prague, Czech Republic
Email: {david.hoksza, jakub.galgonek}@mff.cuni.cz

***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 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 SCOP classification accuracy. Regarding the accuracy, our solution outperforms the contemporary solutions at two out of three tested levels of the SCOP hierarchy.**

***Keywords*-protein structure, TM-Score, SCOP, classification**

I. 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 PDB [2] a need for certain form of organization of these structures emerged. From all the classifications, manually curated hierarchical evolutionary classification SCOP [12] 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 asses 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.

This research has been supported in part by Czech Science Foundation (GAČR) project Nr. 201/09/0683 and by institutional research plan number MSM0021620838.

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 catch 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 the 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 [11].

In this paper, we approach both problems - we propose an algorithm for pairing amino acids (sections II-A, II-B) and this pairing is forwarded into an improved algorithm for computing TM-score (section II-C).

Lets revisit few well known algorithms used for comparison of protein structures. DALI [8], 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. CE [15] 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 SSAP [17] which utilizes dynamic programming (DP) to asses similarity to pairs of amino acids which are represented by so-called views (vector of distances to all the other amino acids). Optimal 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 [19], since it is the solution with which the TM-score [18] is usually presented. TM-score is considered as the standard for

evaluating similarity of two structures given an alignment (see section I-A). 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 [3] uses global DP solution where scoring matrix is based on features obtained from Voronoi tessellation. The same collective of authors presented later a solution called PPM [5]. PPM identifies core blocks (blocks in the two structures sufficiently similar) which are then used to create a graph of core blocks. That path in the graph is chosen, that minimizes the cost of mutation. Finally, the most recently presented solution is Vorometric [14]. 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.

A. 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 [10]. In *RMSD*, distances (after translation) among paired amino acids are computed and then aggregated to get the distance:

$$\text{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} {d_i}^2}. \tag{1}$$

N stands for the length of the alignment, d_i for the distance between i^{th} 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 [18] was proposed which puts emphasis on closer pairs by weighting them higher. The superposition is obtained by its own algorithm exploiting *Kabsch* (see section II-C for details and optimalizations). TM-score value (to evaluate quality of the superposition) is defined as:

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

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 i^{th} pair of aligned amino acids and d_0 is a normalization parameter (see [18] for details).

II. Methods

In this section we describe our approach to protein structure alignment which consists of finding best possible alignment of amino acids based on density of residues. Afterwards we revisit the TM-score algorithm for finding optimal superposition based on given alignment presented in [18].

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 NP-hard, 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.

A. 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 [7]. 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 3-dimensional 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. The description of the VPT semantics follows.

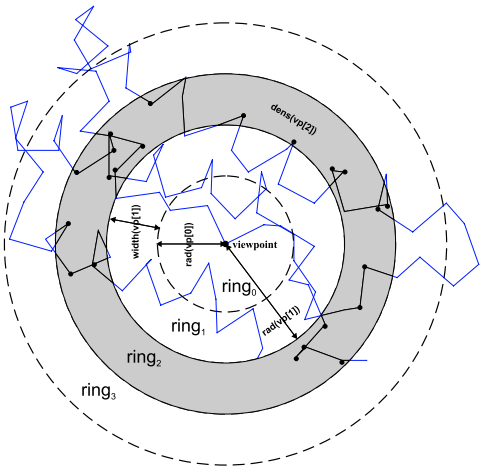

Figure 1. Visualization of a protein with PDB ID *1apc* in 2D (vertices on the curve correspond to C_α residues of individual amino acids; $dens(vp[2])$ equals number of dots in $ring_2$).

Let vp represent a particular viewpoint, then $vp[i]$ stands for the i^{th} ring, $rad(vp[i])$ for the distance from the viewpoint to the further edge of the i^{th} 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 i^{th} ring (see Fig. 1). Finally, let $VPT[i]$ be the i^{th} coordinate of the feature vector (viewpoint tag). Based on these terms, we propose several VPT semantics.

- ***sRad***: For *sRad* (radius based semantics) 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) 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.
- ***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 and moreover residues from distinct SSE types are stored separately. Hence dimension of the VPT increases twice. i^{th} ring is represented by $VPT[2i]$ (α type residues) and $VPT[2i+1]$ (β type residues). $VPT[i]$ is defined equivalently to the *sRad* (*sDens*) VPT semantics.
- ***sDir***: For *sDir* (direction based semantics) holds:
  - $\forall i, j : width(vp[i]) = width(vp[j]) = w$
  - $\forall i : VPT[i] = \sum$(*pairs of consequent residues in the* i^{th} *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 i^{th} coordinates of the feature vectors, we define weighting scheme w as $w(i) = n - log(i)$.

B. Finding an Alignment

Using VPTs and distance functions from section II-A we apply DP to find optimal pairs of amino acids (similar viewpoints) following sequence order. This solution can resemble SSAP [17], 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.

1) Needleman-Wunsch (Global Alignment): Needleman and Wunsch (NW) [13] 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 i^{th} and j^{th} letters should be aligned of whether a gap should be introduced into one of the sequences. Originally, NW was proposed as a maximization problem where similar amino acids scored higher and gap penalties were usually negative. In order 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) Smith-Waterman (Local Alignment): To be able to concentrate more on conserved parts of compared protein structures we also use Smith and Waterman (SW) algorithm [16]. SW was also proposed for protein sequences to find highly similar (conserved) subsequences.

Using SW with our algorithm is not so straightforward as using NW. If minimization would be the only change to the algorithm than, because of adding zero (minimum value ensuring locality of the alignment) to the recursive formula, each alignment of length at least one would score higher then zero and hence would be impossible to achieve. In order 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 $\frac{3}{4} * \mu$ (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.

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.

C. 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 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 in order to increase speed of the algorithm causing only a small or no decrease of the classification accuracy.

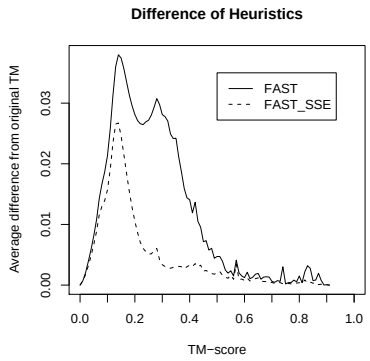

Figure 2. The average difference of FAST and FAST_SSE modifications according to the full TM-score (y-axis shows the average decrease of the score).

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 [18] 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.

In order 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 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 alignment.

In general, FAST and FAST_SSE approximations show worse results 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, which are interesting for us, full TM, FAST and FAST_SSE give comparable results.

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

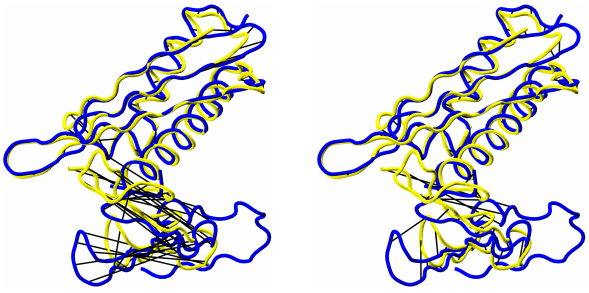

Figure 3. Superposition and alignment (aligned residues connected by black bars) of the proteins d1nyef_ and d1n2fa_ before (a) an after (b) iterative DP (with TM-scores 0.56553 and 0.70329).

In order 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 III-B). 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 with the query are further examined
- *range filtration* - only proteins in a given distance from the most similar protein are further examined[2]

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 [19], we employ NW with scoring matrix S, defined as:

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

d_{ij} being the euclidean distance of the i^{th} and j^{th} 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 21). 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 after two DP iterations demonstrates Fig. 3b.

[2]Being more effective than the more common method considering proteins in a given distance from the query since it is more query-specific.

[3]Image generated by VMD [9].

III. Experimental Evaluation

To evaluate our method and to compare it to 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 (8 rings each having width 3Å) to extract feature vectors[4]. The machine running experiments was 2.3 GHz 4xQuad-Core AMD Opteron CPU, 64GB RAM with Red Hat Linux Server 5.3.

Using dataset originally presented in [3] (also used in [5] and [14]) (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 [4] (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 4357 objects, evaluating number of correctly classified proteins into family, superfamily and fold according to the SCOP classification.

A. Classification Accuracy (CA)

Methods to be compared to our solution include BLAST [1] (used with its standard settings), CE [15], PPM [5], TM-align [19], Vorolign [3] and Vorometric [14] (its TM variant applying TM-score superposition).

In Tab. I, 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.

[4]The values or parameters of the semantics are based on results in [7].

Table I
Classification accuracy on SCOP levels.

	Family	Superfamily	Fold
db-iTM	86.4	**95.6**	**98.0**
db-iTM$_{orig}$	87.0	**95.8**	**98.1**
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

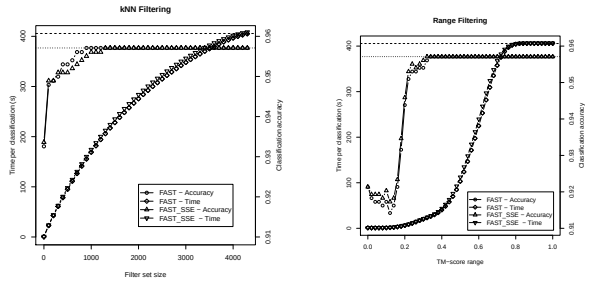

Figure 4. Impact of the prefiltration candidate set size on the runtime and CA (the upper dashed line represents runtime of the full TM version, the lower dotted line represents its CA).

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.

B. Speed-up Evaluation

Tab. I shows that with the *db-iTM* we are able to achieve comparable accuracy to the *db-iTM*$_{orig}$. The advantage of TM_FAST is its speed which is counterbalanced with the accuracy decrease (*db-iTM*$_{orig}$ takes 406s whereas *db-iTM* takes only 12s). But we are able to noticeably decrease runtime of the *db-iTM*$_{orig}$ by prefiltration (see II-C1) whilst keeping the original accuracy. Fig. 4a shows that for prefiltration set size of about 850 (hence for the whole DB *db-iTM* is computed and 850 top proteins are verified by *db-iTM*$_{orig}$) the accuracy is identical to *db-iTM*$_{orig}$. For such a set size the runtime is 152s, hence less than 50% of the runtime of the full version. Using range prefiltration, the results are even better since when CA of the filtration method hits the accuracy of full TM the runtime is 26s in case of using TM_FAST for prefiltration and 24s using TM_FAST_SSE. That means we are able to achieve accuracy of the full TM method in only 6% of its time.

C. 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 $\frac{\#correctly\ classified\ proteins}{\#retrieved\ proteins}$

and recall as $\frac{\#correctly\ classified\ proteins}{\#proteins\ in\ the\ query's\ "family"}$. In Fig. 5[5] precision-recall curves are shown belonging to *sRad(SSE)*, *sDens(SSE)* and *sDir* semantics 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).

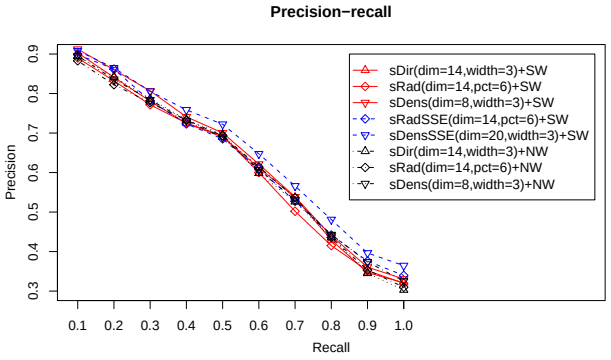

Figure 5. Precision-recall curves for various VPT semantics (*dim* = number of rings, *width* = width of a ring, *pct* = percentage of amino acids in a ring).

IV. 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 dynamic programming. 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[6].

References

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

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

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

[6]Preliminary research on topic of further speed-up has already been discussed in [6].

[3] F. Birzele, J. E. Gewehr, G. Csaba, and R. Zimmer. Vorolign - Fast Structural Alignment using Voronoi Contacts. *Bioinformatics*, 23(2):e205–e211, 2007.

[4] J. Chandonia, G. Hon, N. Walker, L. Conte, P. Koehl, M. Levitt, and S. Brenner. The ASTRAL compendium in 2004. *Nucleic Acids Research*, 32:D189–D192, 2004.

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

[6] J. Galgonek and D. Hoksza. On the effectiveness of distances measuring protein structure similarity. In *SISAP*. IEEE, 2009.

[7] D. Hoksza. DDPIn - Distance and Density Based Protein Indexing. In *CIBCB*, pages 263–270. IEEE, 2009.

[8] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *J Mol Biol*, 233(1):123–138, September 1993.

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

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

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

[12] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol*, 247:536–540, 1995.

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

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

[15] I. N. Shindyalov and P. E. Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Eng*, 11(9):739–747, September 1998.

[16] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197, March 1981.

[17] W. R. Taylor and C. A. Orengo. A holistic approach to protein structure alignment. *Protein Eng*, 7(2):505–519, 1989.

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

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