DDPIn - Distance and Density Based Protein Indexing

David Hoksza

***Abstract*— Protein structure similarity and classification methods have many applications in protein function prediction and associated fields (e.g. drug discovery). In this paper, we propose a new protein structure representation method enabling fast and accurate classification. In our approach, each protein structure is represented by number of vectors (based on histogram of distances) equivalent to the number of its C_α residues. Each C_α residue represents a viewpoint from which the distances to each of the other residues are computed. Consequently, we use several methods to convert these distances into a n-dimensional feature vector which is indexed using a metric indexing structure (M-tree is the structure of our choice). While searching, we use single or multi-step approach which provides us with classification accuracy and speed comparable to the best contemporary classification methods.**

I. Introduction

Proteins consist of chains of building blocks called amino-acids and play important role in many biological processes. One of the basic tasks in proteomics is to find out similarity between a pair of proteins since similar (sequentially or structurally) proteins are considered to secure similar functions. Hence, by finding a similar protein to a newly discovered one, the function of the investigated one can be concluded. Originally, protein similarity was based on sequence, hence proteins with similar sequence were believed to be functionally similar. However, it had been shown later that sequence similarity does not necessarily conclude structural (and hence functional) similarity [8][5]. In general, since the protein function is derived from its three dimensional structure, structure is closer to the function than sequence.

The algorithms for determination structure similarity of a query and database of proteins are classifiable into three categories[1]. Firstly, pairwise structural alignment can be based on the level of C_α (or other amino-acids' representative atom) coordinates. Second type of algorithms combines a prefiltering based on secondary structure elements (*SSE*) followed by a more detailed step based on similar principles as the previous class of algorithms. And finally, the third type of algorithms rely on database indexing where feature vectors are extracted from the structure and indexed. In the retrieval step, feature vectors of the query protein are exploited for querying the indexing structure.

The similarity in protein structural databases is usually expressed by a matching (alignment) of pairs of protein structures. The goal of the matching is to determine mutual mapping (equivalence) of atoms in both structures. Such a mapping is then forwarded to an algorithm being able to compute the optimal superposition[2] of the two structures in a linear time [10]. Such a superposition is considered to be optimal, which minimizes a given distance function. In most cases, similarity measures based on a root mean square deviation (RMSD) are utilized.

The methods exploiting C_α coordinates only to match protein structures emerged first. Most well known examples belonging to this group of algorithms are *DALI* [9], *CE* [15], *SAP* [19][20][18], and others. *DALI* uses a matrix of inter-residual distances (distances within pairs of atoms of the opposite structures) and employs an observation that similar structures have similar distance matrices (or submatrices). Hence similar submatrices (contact patterns) are identified and stored in a list of pairs. An alignment is started with a pair from the list and is then refined by adding or removing contact patterns. *CE*, on the other hand, uses intra-residual distances (distances within atoms of a single structure). It identifies similar portions of substructures in proteins to be aligned (AFP - aligned fragment pairs) and these are further extended to minimize one of three ad-hoc defined measures. Final match is optimized by applying a dynamic programming (DP) algorithm. *SAP* is interesting by extensive use of the idea of *Smith-Waterman* [16] algorithm. Each residue is represented as a vector of distances (*view*) to the remaining residues. For each pair of views in the compared proteins, DP matrix is created and optimal path is recognized. In the second step, all the optimal paths from step one are aggregated into a top-level DP matrix and final iteration of DP is applied to get the resulting alignment. Further examples are *PROSUP* [11] or *STRUCTAL* [7][17] which both identify initial seeds and further iteratively extend them (both use DP).

Faster (not necessarily more accurate) methods benefit from a filtration prestep based on SSEs. For example *VAST* [12] is similar to *CE*, however it uses SSEs as vertices of a bipartite graph and algorithm for finding maximal clique is used to find the initial alignment.

To handle increasing sizes of databases, methods based on indexing were discovered. These methods store features learned from the 3D structure and/or sequence. Since algorithms in this group exploit features, they usually give out similarity of the proteins based on their features rather

This work was supported by GAUK grant under project code 57907 and GACR grant under project code 201/09/0683

[1]Although this is not the only possible 'taxonomy' of algorithms for determination protein similarity. Our presented taxonomy is more speed-oriented, in contrast to other that can classify algorithms based on their inner algorithmic behavior.

[2]Translation and rotation of one protein (proteins are seen as rigid bodies) to spatially mount it on the other one.

than the mapping itself. Mapping is not indispensable for classification[3] purposes and therefore such algorithms find use in the classification area. Techniques in this group can be further divided into two categories depending on the type of indexing methodology they use. In one category reside algorithms using indexing trees to speed-up the querying such as *PSI* [3] and *PSIST* [6] and in the other algorithms using geometric hashing [21] - *ProGreSS* [1] or *CTSS* [2]. Since our paper introduces a method falling into the group of algorithms exploiting feature vectors, we describe these algorithms a bit more thoroughly in the following section.

II. Index-based Approaches to Protein Classification

A. Existing Methods

All methods presented in this section use some kind of feature extraction to get feature vectors representing 3D structure of the protein as accurately as possible. These vectors can be viewed as points in a n-dimensional space (where n is the dimension of the vectors) and as such handled by a universal indexing method.

As stated earlier, *PSI* [3] and *PSIST* [6] employ trees for storing their vectors. *PSI* uses SSEs as fundamental objects. Each SSE is represented by its center of mass. For each, the distance to each other SSE is computed. Residues near the investigated residue are then used for creating the feature vector. Finally, vectors are stored in a R*-tree. Unlike *PSI*, *PSIST* utilizes suffix tree to store its features. It slides a moving window along the backbone of the protein and at each position it encodes distance and torsion angle to all the following residues in the window. These representations are encoded as sequences of natural numbers and used as strings for the suffix tree.

Progress [1] and *CTSS* [2] take advantage of geometric hashing [22]. *CTSS* understands protein structure as a spline in 3D space and applies techniques of differential geometry to it. First, it smooths the spline and then for each C_α it records curvature and torsion angle into a hash table. *Progress* is, as far as we know, the only tool of this kind that combines 3D and 1D (sequence) characteristics into one feature. As *PSIST*, also *Progress* uses a sliding window to capture position and sequence difference among the starting residue and content of the window. To these two vectors, Haar wavelet transformation is applied and resulting $2n$-dimensional vector is hashed.

B. DDPIn

We introduce a fast novel method called *DDPIn* - **D**istance and **D**ensity based **P**rotein **In**dexing. Similarly to methods mentioned in the previous section, also our approach employs a feature extraction. We use C_α atoms as the fundamental particles (although any other atom can be used for amino-acids' representation) and operate on distances among them. We get one feature vector for each C_α residue (hence for a protein with n amino-acids we get n feature vectors). In section III-A, we present several semantics of these vectors. All of them are based on clustering the distances/residues into spheres with the center in the examined residue. Some of them make use of radii of those spheres whereas others compute density of C_α residues in the spheres. Further detailed explanation of the feature extraction techniques can be found in section III-A.

After obtaining feature vectors for all the amino-acids in a protein, we deposit them into an indexing structure. In this paper, we employ M-tree although any metric indexing method can be utilized instead, since we use several distance functions which all form metric. Each inserted object consists of the feature vector and the ID of the protein from which the extraction was carried out.

When finding similar proteins to a query protein, feature vectors of the query are extracted and each of them is used as a k-nearest neighbor (kNN) query to the indexing structure. Hence, for a query protein with n amino-acid we get n resultsets of size k. These are then merged to get the most similar proteins to the query one.

The main goal of *DDPIn* in this paper is to provide an efficient classification. As the determinant of accuracy of our method we use the SCOP classification [13] which is widely accepted as the gold standard for classification since it is generated by human experts. Thus, after finding similar proteins to the query one, a ranking scheme takes place (based on the nearest neighbors) to obtain the correct SCOP category (usually superfamily).

To increase the accuracy of the classification process, we add so called *healing step*. In this phase, cross-reference query into another indexing structure is used (another feature extraction type), the protein is ranked considering this result and resultsets of both scans are merged. Proteins not having unambiguous classification are than "healed" by another method and acquired classification is considered to be the final and correct one.

Hence, searching is divided into two steps - simple search and so called healing. Accuracy of the first step is up to 93% for classifying a protein into the *SCOP* superfamily. Resulting accuracy after inclusion of the healing step rises up to 99%, which is an outcome comparable to the best up to date methods.

III. Protein Indexing

The indexing runs in two steps - first of all, feature vector is created for each amino-acid of each protein and subsequently the vector is inserted into a metric indexing structure. Both steps, together with metrics used for computing distance between objects (feature vectors), are described in this section.

A. Protein Representation Methods

DDPIn is based on the principle of nested three-dimensional balls. For each C_α residue r (called *viewpoint*), set of balls having their centers in r is created (Fig. 1). In this way, set of 3D rings arises, each ring containing C_α residues

[3]Determination to which category in *SCOP* [13] or *CATH* [14] classifications an unknown protein should be classified.

being in the "appropriate" distance from r. We suggest a few ways how to acquire proteins' features based on perimeters of the balls (which is equivalently definable with the help of widths of the rings) and how to extract features from these representations. Vector of features acquired in this way is called *viewpoint tag* (*VPT*) because it is a blueprint of the protein according to a viewpoint (it is similar idea to the notion of *view* in *SAP*).

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.

1) *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.
2) *sRadNorm*: For *sRadNorm* (radius based semantics) holds:
   - $\forall i, j : dens(vp[i]) = dens(vp[j]) = p$
   - $\forall i : VPT[i] = rad(vp[i])/|pt|$

   where p is a user-defined parameter representing percentage of amino-acids in the protein, and $|pt|$ is number of amino-acids in a protein pt (for which VPT is being computed).
3) *sRadSSE*: *sRadSSE* is identical to *sRad* 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* VPT semantics.
4) *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.
5) *sDensSSE*: *sDensSSE* is a density based equivalent of the *sRadSSE*.
6) *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.

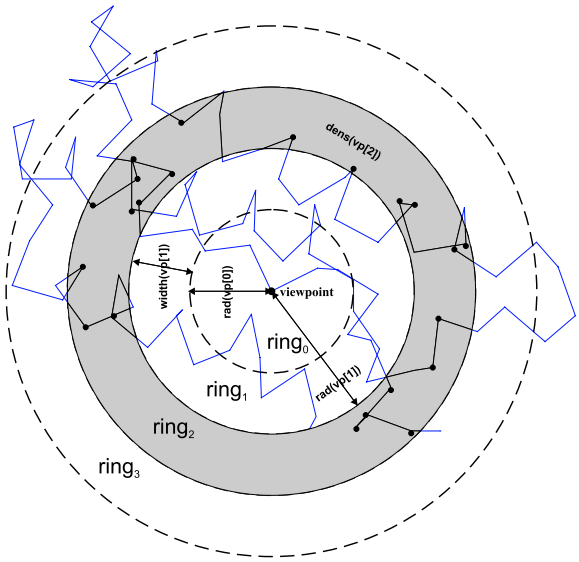

Fig. 1. DDPIn visualization of a protein with PDB ID *1apc* in 2D (vertices on the curve correspond to C_α residues of individual amino-acids ($dens(vp[2])$ equals number of dots in $ring_2$).

B. Indexing Structures

To speed-up the search for similar VPTs, we build an indexing structure upon the database of VPTs. Selected indexing structure of our choice is the M-tree [4], but any metric indexing structure can be used instead. M-tree, which resembles R-tree, belongs to the class of so called metric access methods (MAMs[4]). However, in contrast to R-tree's MBRs M-tree uses n-dimensional spherical regions. Each inner node represents number of *routing entries* each of which is defined by its center (which is one of the indexed objects), perimeter of the ball covering all the descendant objects in the tree and pointer to its subtree. Leaf nodes consist of *ground entries* (VPTs).

Most abundant query types for the M-tree are *range* and *kNN* queries. For the range query, one enters a query object and radius (range) and all DB objects in the distance within the range from the query object are returned. M-tree can effectively handle such queries in logarithmic time by virtue of metric properties. Its nodes are traversed hierarchically from the root node and such a descendants are taken into account, which have nonempty intersection with the query ball (query object and its range). kNN queries are processed similarly but the query radius is not known in advance, hence it is being adjusted (decreasing from the initial value infinity) interactively as the query traverses the tree.

C. Metrics

We are to apply metric indexing methods to the VPTs, thus we need to utilize distances that fulfill metric properties

[4]MAMs employ metric functions to handle objects. In this approach, a metric is used as a black box which accepts a pair of objects as its input and outputs distance of these objects. Hence the methods are independent on the choice of distance function to the extent that it fulfills metric properties.

(non-negativity, identity of indiscernibles, symmetry, triangle inequality). The metrics we tried to use in DDPIn follow.

1) L_2 Distance: The fundamental metric function is the Euclidean distance also known as L_2 distance. In its generalized form, it is usable for n-dimensional space where n is dimension of the VPTs.

2) Weighted L_2 Distance: Since we suppose that direct structural neighborhood of the viewpoints is more predicative for similarity determination, we use weighted Euclidean distance as the similarity function. In this approach, each partial difference in the distance computation is multiplied by a respective weight. We try various weights with the emphasis on favoring first few coordinates (see section V-B.3).

IV. Querying

DDPIn can answer two types of queries - 'find the most similar protein in the DB to the query one' and 'classify a query protein into a group of proteins' (the classification query is built atop the similarity query). These queries can be answered by our two-step search. First step represents a simple scan, while the second plays the role of a cross-validation process where probably wrong predictions are identified and possibly corrected.

A. One-step search/classification

First of all, we extract VPTs from the query protein in the same way indexing is done. Then we take each of the VPTs and run kNN query against the database of VPTs (VPT semantics of the query and the DB objects has to be the same) where k is the object of the experimental evaluation (but in general, best results were obtained with k going from 20 to 40 depending on the feature extraction type and its parameters). In the resultset, we obtain k most similar VPTs. In order to sort the proteins (those whose VPTs appear in the resultset), we came out with the ranking scheme where i^{th} VPT in the result contributes with value $k - log(i)$ to the overall score of the protein it represents. Hence, the overall similarity score for a query (q) and indexed (p) protein is:

$$s(p,q) = \sum_{i=0}^{|q|-1} \sum_{j=0}^{|p|-1} \begin{cases} k - \log(z), z \in \{1 \dots k\}, \\ \qquad if \;\; VPT_j^p \in NN(VPT_i^q), \\ 0, \qquad otherwise \end{cases} \tag{1}$$

where VPT_b^a stands for a^{th} VPT in protein b, $NN(x)$ for nearest neighbor set of VPT x and z for position of the given VPT in the NN set.

Finally, the protein with the highest similarity score according to the query protein is evaluated as the most similar to the query one. If our goal is to classify the protein, we append the classification step. We approached the problem in the most straightforward fashion - we consider the nearest neighbor's group as the resulting group. Other alternatives did not turn out to be noticeably more effective.

B. Two-step search/classification

Solution described in the previous section reaches up to 93% accuracy in classifying a protein into SCOP superfamilies[5]. We realized that sets of proteins not being classified correctly are not always identical for various VPT semantics or miscellaneous parameters of the same semantics. If we are able to distinguish the correctly classified proteins we might be able to reclassify the rest. Such a reclassification is called *healing* in DDPIn's terminology.

For recognizing wrongly classified proteins, we use cross validation. We execute two one-phase scans with different VPT semantics (two different indexing structures) and according to the results we decide, whether the resulting protein is classified correctly or should undergo healing. The workflow of the whole process is schematically depicted in Fig. 2.

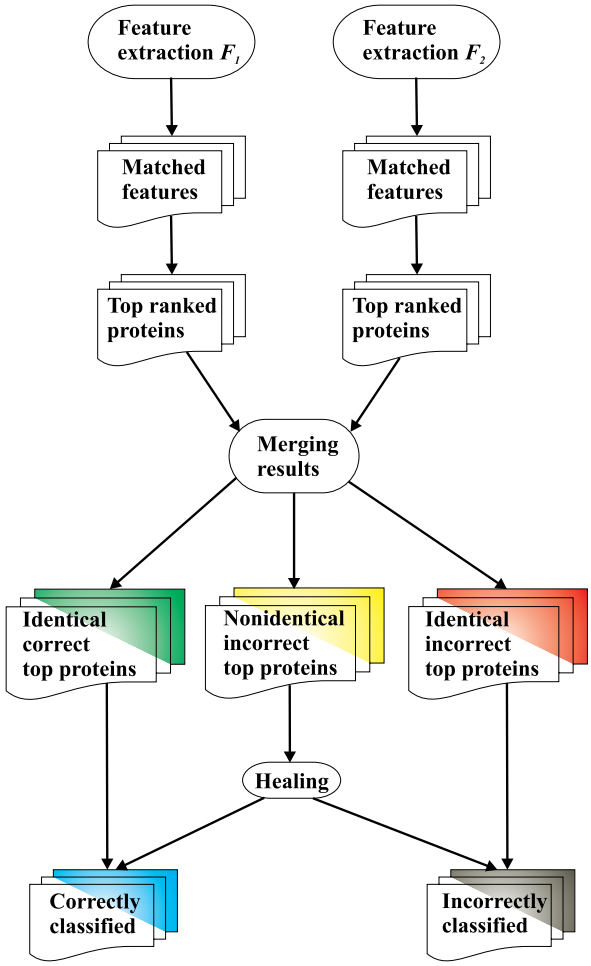

Fig. 2. General workflow of the DDPIn method.

Let's define S_1 and S_2 as index scans one and two and $p(S_i)$ as the classification obtained from S_i. Then if $p(S_1) = p(S_2)$ we suppose the prediction is correct. Otherwise, we employ the healing process. Since the only

[5]The SCOP classification divides proteins hierarchically into *classes*, *folds*, *superfamilies* and *families*.

piece of information we have in that case is that $p(S_1)$ and $p(S_2)$ differ, we can not conclude whether $p(S_1)$ is correct or whether $p(S_2)$ is correct or whether even both are wrong. So we apply an entirely different method to pick up the final classification - *Smith-Waterman* alignment [16]. It operates purely on primary structure of the proteins. Hence, we take the query and align it with set of DB sequences D. Proteins in D can be chosen from the partial results of S_1 or S_2, from their fusion, or even they can represent the whole database. Size of D has a serious impact on the resulting time, since Smith-Waterman is of quadratic complexity according to the sequence length.

Experiments have shown that the healing phase can seriously increase accuracy. Determinants of the accuracy are the semantics of S_1, S_2 and the parameters of the healing process. Suitability of using a pair of VPT semantics (which defines the pair of indexing structures for S_1 and S_2) can be found out by studying pivot tables based on Tab. I.

	$p(S_1)$ or $p(S_2)$ correct	$p(S_1)$ and $p(S_2)$ wrong	$\sum$
$p(S_1) = p(S_2)$	Q_1	Q_2	$Q_1 + Q_2$
$p(S_1) \neq p(S_2)$	Q_3	Q_4	$Q_3 + Q_4$
$\sum$	$Q_1 + Q_3$	$Q_2 + Q_4$	

TABLE I
PIVOT TABLE FOR DECISION WHETHER TO HEAL.

Q_1 represents number of queries that are correctly predicted by both searches and do not need to be healed. Problematic query proteins are those contributing to Q_2. We can not heal these proteins since $p(S_1) = p(S_2)$ but they are determined incorrectly. Hence Q_2 is a value contributing to the inaccuracy and we head toward using such a combination of VPT semantics that would decrease this value. And finally, Q_3 and Q_4 are proteins to be healed. Hence, in general we are trying to use such pairs of VPT semantics with high Q_1 and low Q_2.

C. Multi-step search/classification

Generalizing the idea of the two step classification leads us to multi-step classification. We can use multiple VPT semantics for mutual cross-validation. The idea can be to accept a prediction arising from a scan S if at least further $(n-1)/2$ scans from the overall number of n scans (n-step classification) classify the query into the identical group. Considering Tab. I, Q_1 might rise significantly, but on the other hand Q_2, which we try to minimize, will probably rise too. In the experimental evaluation we present a pivot table showing the influence of adding further scans into the process of classification.

V. EXPERIMENTAL EVALUATION

In order to experimentally verify the concept of DDPIn and to evaluate the influence of various parameters mentioned in the sections above, we carried out thorough tests. The testing platform was based on 2.66 GHz Intel Core(TM)2 Duo CPU, with 2GB of RAM, running Windows XP. To compare our method with the previously released ones, we reused the testbed presented also in [6]. It includes proteins from superfamilies (each having at least 10 proteins) coming from four SCOP classes (all α, all β, $\alpha + \beta$ and α/β). Since version 1.65 of SCOP is used in these experiments and from each superfamily 10 representatives were chosen, the total number of proteins in the database was 1810 (181 superfamilies). When querying, we used two query sets. To be able to compare the classification's accuracy with other methods, we used the query set occurring in their experiments - each superfamily contributes with 1 protein to the query set, hence the set contains 181 proteins. But to thoroughly examine effects of the various parameters, we used the whole database as the query set for experiments focusing on inner settings of DDPIn.

The main object of our tests was the classification accuracy when using single and multi-step approaches. Within the individual approaches we tested the influence of VPT semantics, metric distance measures and healing VPT semantics' combinations on the overall accuracy. Since the multi-step approach consists of multiple single-step scans, we use these to present characteristics of individual VPT semantics. But we precede that the accuracy of the single-step approach alone is far from optimum (as we show in section V-C where one can also find comparison with the other methods).

A. Indexing

First experiments concerned sizes of indexes needed to store feature vectors of all the proteins. The results are presented in Tab. II. Although number of indexed objects is dependent solely on the dimension of the VPT, sizes of indexes with various semantics having the same dimension vary because of the variance in utilization of the M-tree nodes. On the other hand, variance of the utilization is not very high and so the number of objects is approximately $dimension \times 1810 \times c$, where c is a small constant dependent on the M-tree. Yet, the dimension can be considered as the main determinant of the index size.

VPT semantics	*dimension*	*#items*	*index size*
Radius ($dens(vp[i]) = 7$)	7	312338	24.88MB
Radius($dens(vp[i]) = 7$)	14	312330	37.46MB
Density ($width(vp[i]) = 3$)	7	312263	24.74MB
Density ($width(vp[i]) = 3$)	14	312990	37.48MB

TABLE II
INDEX SIZE.

B. Single-step Classification

1) kNN queries: We tried to find out the optimal value of k for kNN queries. Fig. 3 shows the accuracy of classification proteins into superfamilies for different values of k. In all the cases, weighted L_2 metric was used where weight $\log(dim - i + 1)$ was assessed to the i^{th} dimension. We can observe that low k values cause loosing correct VPTs and hence decrease of the accuracy. On the other hand,

for k above 5 not all of the semantics behave identically. In general, it can be said that optimal values are reached between 20 and 40 but there is no value k, that is optimal for all the semantics. E.g. optimal value of k for *sRad* having density 7 is 30, but changing density to 6 causes increase of the optimal k to 40. When comparing the individual semantics in further experiments, we decided to use $k = 30$, since such a value seems to be close to the optimum for most of the semantics.

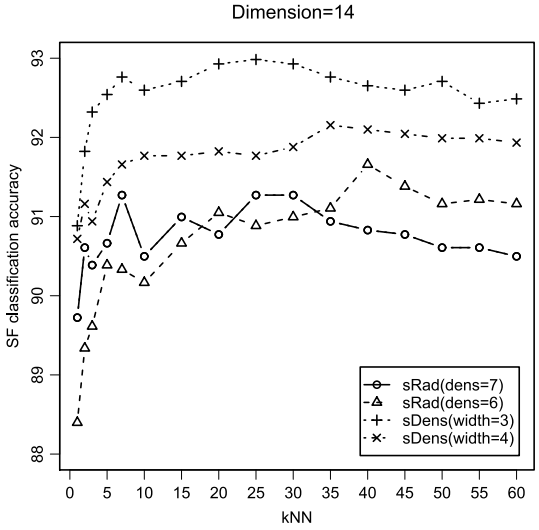

Fig. 3. Increasing k in kNN queries.

2) VPT Semantics: In section III-A, we presented few VPT semantics. Figures 4a and 4b represent overview of the influence of their parameters on the quality of classification. Comparison of Fig. 4a and Fig. 4b supports the superiority of the *sDens* semantics which was observable in the previous experiment, too. We can also notice an interesting property of our representations - if we add more information into it (enrichment by the SSE type), the accuracy deteriorates (while keeping the dimension the same). We attribute this phenomenon to low density of C_α residues. Splitting (already not very rich in information) rings into more segments causes further loss of information which makes the VPTs insufficiently discriminative (informative).

Normalizing the individual components of VPTs does not provide us with additional accuracy either (see *sRadNorm* results in Fig. 4a). The reason stems probably from loosing information about differently long proteins (longer proteins have higher values of VPT coordinates, since $p\%$ of their aminoacids lay further from the viewpoint) which is hence highly discriminative argument for the VPT similarity.

Fig. 5 represent the effect of dimension's growth. For most VPT semantics, best results are obtained for dimension 14. For the *sRad** semantics, the highest reachable dimension for density 7 is 14. For higher dimension, 15^{th} and following dimensions do not have any residues, because width corresponds to percentage of the overall number of residues in the given protein. Hence, *sRad** semantics are actually not defined for those dimensions, but for transparency we show

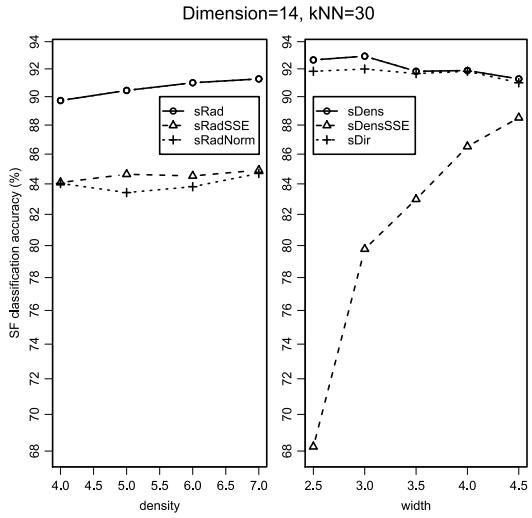

Fig. 4. Influence of VPT semantics properties to accuracy.

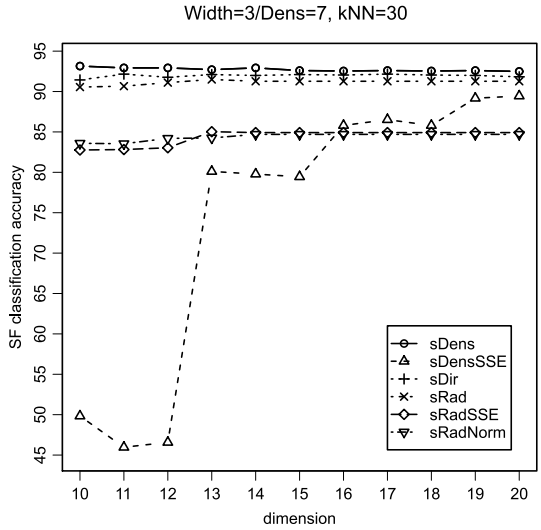

Fig. 5. Impact of the VPTs' dimension on the accuracy.

their accuracy in the figure as constant values identical to the last defined combination of width and dimension. Clearly, the dimension has some effect on the number of correctly predicted proteins but the effect is most significant except for the *sDensSSE* semantics. The reason is that dimension of the **SSE* semantics is actually two-times lower than for the other semantics because for each ring, two values are stored. And in line with this fact, the results of *sDensSSE* are comparable (although still slightly worse) to other density-based semantics when the dimension comes to 20.

3) Metrics: We chose three best VPT semantics to try out the behavior of various metric functions. We used L_2 and weighted L_2 function together with several different sets of weights. The outcomes can be seen in Tab. III. It turns out that weighted L_2 with $\log(dim - i + 1)$ and $(dim - i)/dim$ weighting systems work best since they favor direct neighborhood of viewpoints (further coordinates obtain lower weights). Opposite approach (favoring more distant residues)

Metric	*sRad* (dens[i]=7)	*sDens* (width[i]=3)	*sDir* (width[i]=3)
L_2	1631(90.1%)	1674(92.5%)	1657(91.6%)
Weighted L_2 $(\log(dim - i + 1))$	**1652(91.3%)**	1682(92.9%)	1665(92%)
Weighted L_2 $((dim - i)/dim)$	1651(91.2%)	**1683(93%)**	**1670(92.3%)**
Weighted L_2 (i/dim)	1592(87.8%)	1659(91.7%)	1648(91.1%)
Weighted L_2 $(\log(i + 1))$	1612(89.1%)	1670(92.3%)	1655(91.4%)

TABLE III
METRICS' COMPARISON (DIM=14).

turned out to perform considerably worse (as awaited).

C. Two-step Classification

With the knowledge of the best VPT semantics (among the proposed ones) and their optimal parameters we are ready to use them within the two-step approach. We remind that apart from calibrating the characteristics of the partial scans, in two-step approach we can also optimize the healing set's attributes. Tab. IV shows results achieved when the whole nearest neighbor set of the first scan was used as the healing set (the semantics at the left edge of the table is considered as the first scan). Hence proteins that had at least one VPT in one of the query's VPTs' NN-set are aligned with the query with the help of the *Smith-Waterman* algorithm and the most similar is chosen as the nearest neighbor. Each cell of the Tab. IV shows pivot table (see section IV-B) for the respective pair of semantics. Under it, resulting accuracy of the two-step approach in the form of the absolute number of correctly predicted proteins together with the resulting superfamily classification accuracy is presented. We can notice that the combination of *sDir* and *sDens* is not suitable for healing since the semantics of both approaches are similar (based on density of residues in rings) and hence Q_2 value is about two-times higher then for the other combinations.

	sRad	*sDens*	*sDir*
sRad		1624 | 20 90 | 76 $\|HS\| = 321$ $\overline{a}$=1744(**96.4**%)	1611 | 24 102 | 73 $\|HS\| = 321$ $\overline{a}$=1735(95.9%)
sDens	1624 | 20 90 | 76 $\|HS\| = 179$ $\overline{a}$=1738(96.0%)		1643 | 44 64 | d $\|HS\| = 176$ $\overline{a}$=1724(95.2%)
sDir	1611 | 24 102 | 73 $\|HS\| = 216$ $\overline{a}$=1734(95.8%)	1643 | 44 64 | 59 $\|HS\| = 215$ $\overline{a}$=1726(95.4%)	

TABLE IV
ACCURACY OF ANTISYMMETRIC FULL-SIZED HEALING SETS.
($|HS|$ ... HEALING SET SIZE, $\overline{a}$ ... ACCURACY)

The table presents the anti-symmetric approach (as we call it) meaning the healing set forms the NN-set of one of the scans. It is also possible to use union of the NN-sets, results of which are presented in Fig. 6. The union here was produced by subsets of given size of the NN-sets. Size of each subset is defined by the multiplication factor, thus each set contributes to the union with portion corresponding to the given factor. We emphasize the NN-sets are sorted according to the distance to the query hence the union contain always the most similar proteins. Best results were achieved for the *sRad-sDens* combination using factor 0.4 (96.4%). The results show interesting observation that higher size of the healing set does not necessarily lead to improved accuracy. This leads us to the idea of using the whole database (1809 sequences) as the healing set. It turns out that apart from the substantial slow-down, such a modification moreover decreases the accuracy (95.8% for *sDens-sRad* semantics combination). Hence using healing sets emerging from the NN-sets is a good choice because it filters out sequentally similar but structurally disimilar proteins.

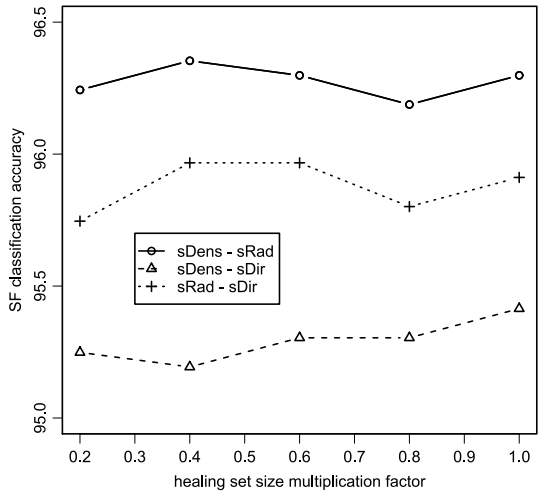

Fig. 6. Influence of healing set size on the accuracy.

1) Comparison to other tools: Here, we took the optimal combination of VPT semantics and their parameters (*sDens*: $width[i] = 7, dim = 14, k = 30$ - *sRad*: $dens[i] = 3, dim = 11, k = 30$) and compared it to the best achieved classification results of other tools as presented in [6]. We underline that the used query set consists of 181 queries only unlike 1810 queries in the previous experiments so the superfamily classification accuracy differs.

Algorithm	*Superfamily*	*Fold*
PSI	88%	N/A
ProGreSS	97.2%	98.3%
PSIST	97.8%	99.4%
DDPIn	**98.9%**	**100%**

TABLE V
SCOP SUPERFAMILY AND FOLD LEVEL ACCURACY COMPARISON.

Tab. V shows that on the reduced query dataset DDPIn's accuracy increases up to the level where it beats the other methods. The presented DDPIn's superfamily accuracy corresponds to two wrong predictions. Nevertheless, although the nearest proteins fall into wrong superfamilies they appear in

the same fold and hence fold classification accuracy is 100% for DDPIn.

D. Multi-step Classification

The final accuracy experiments concerned the accuracy of the multi-step approach. For the experiment, *sRad*, *sDens* and *sDir* semantics were used with the same settings as in the previous experiments and multiplication factor of the healing set was set to 0.4 (the way of getting the healing set and decision whether a query should proceed into the healing step was described in section IV-C). We present here results of the 3-step approach only but we believe the results can be generalized into the higher steps too. The acquired pivot table supports this belief. Q_1 value grows to 1676 but Q_2 grows too. The value of Q_2 is 68 which considerably deteriorates the possible accuracy in such a way that there is no advantage in using the multi-step approach (the accuracy after performing the healing-step is 94.7% for this approach).

E. Performance evaluation

Tab. VI provides information about the running time of DDPIn per query which is divided into the time spent in the scans and time spent in the healing step. Moreover, we present percentage of nodes traversed in the M-tree when querying. This number is directly proportional to the scan time. In the table, multiplication factor 0.4 was used - if the whole DB was used for healing (maximum multiplication factor), the time per query (healing) would be *202* s. Whilst the main determinant of the second step is the size of the healing set, run time of the first step is directly proportional to the percentage of the tree nodes of the total number of nodes that have to be traversed. We can see there is a clear trade-off between speed and accuracy when comparing *sRad* and *sDens* semantics. The table also shows that using a index is inevitable since sequential scan would be too time-consuming.

Method	*Index scan (s)*	*Index scan (% of tree nodes)*	*Healing step (s)*	*Overall time (s)*
sDens-sRad	2.2 + **0.8**	8.3 (12.1, **4.5**)	1.8	**4.8**
sDens-sDir	2.2 + 3.2	14.2 (12.1, 16.2)	**1.0**	6.4
sDir-sRad	3.2 + 0.8	10.4 (16.2, 4.5)	2.2	6.2

TABLE VI

TIME COMPARISON (HS MULTIPLICATION FACTOR = 0.4).

We did not implement methods from [6] by ourselves and hence the speed comparison can be only very approximate. PSIST is presented in [6] as the most accurate method and its time gets around 4 seconds per query (when the best accuracy is requested) on a 2.8GHz CPU PC with 6GB of RAM. Hence DDPIn would be comparable to PSIST in the run time if ran on the same machine.

VI. CONCLUSION

In this paper we presented a novel representation of protein structures based on distances among amino-acid residues called DDPIn. Protein characteristics are stored in feature vectors which are further indexed and queried by a metric indexing method. Finally, a multi-step approach was proposed to increase the accuracy of the classification. DDPIn outperforms the previously released methods in accuracy while using comparable amount of time.

REFERENCES

[1] A. Bhattacharya, T. Can, T. Kahveci, A. Singh, and Y. Wang. Progress: Simultaneous searching of protein databases by sequence and structure, 2004.

[2] T. Can and Y.-F. Wang. Ctss: A robust and efficient method for protein structure alignment based on local geometrical and biological features. *csb*, 00:169, 2003.

[3] O. Çamoglu, T. Kahveci, and A. K. Singh. Towards index-based similarity search for protein structure databases. In *CSB '03: Proceedings of the IEEE Computer Society Conference on Bioinformatics*, page 148, Washington, DC, USA, 2003. IEEE Computer Society.

[4] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *VLDB'97*, pages 426–435, 1997.

[5] Z. K. Feng and M. J. Sippl. Optimum superimposition of protein structures: ambiguities and implications. *Fold Des*, 1(2):123–132, 1996.

[6] F. Gao and M. J. Zaki. Psist: Indexing protein structures using suffix trees. In *CSB '05: Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference*, pages 212–222, Washington, DC, USA, 2005. IEEE Computer Society.

[7] M. Gerstein and M. Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the scop classification of proteins. *Protein Sci*, 7(2):445–456, 1998.

[8] A. Godzik. The structural alignment between two proteins: is there a unique answer? *Protein Sci*, 5(7):1325–1338, July 1996.

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

[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] P. Lackner, K. W. A., S. M. J., and D. F. S. Prosup: a refined tool for protein structure alignment. *Protein Eng*, 13(11):745–752, November 2000.

[12] T. Madej, J. F. Gibrat, and S. H. Bryant. Threading a database of protein cores. *Proteins*, 23(3):356–369, November 1995.

[13] 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.

[14] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton. Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108, August 1997.

[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. *Jurnal of molecular Biology*, 147:195–197, 1981.

[17] S. Subbiah, D. V. Laurents, and L. M. Structural similarity of dna-binding domains of bacteriophage repressors and the globin core. *Curr Biol*, 3(3):141–148, March 1993.

[18] W. R. Taylor. Protein structure comparison using iterated double dynamic programming. *Protein Sci*, 8(3):654–665, March 1999.

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

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

[21] H. Wolfson and I. Rigoutsos. Geometric hashing: An overview. *IEEE Computational Science and Engineering*, 4:10–21, 1997.

[22] H. J. Wolfson and I. Rigoutsos. Geometric hashing: an overview. *Computational Science and Engineering, IEEE [see also Computing in Science & Engineering]*, 4(4):10–21, 1997.