A novel entropy-based hierarchical clustering framework for ultrafast protein structure search and alignment

Baris Ekim[1,2]

***Abstract*— Identification and alignment of three-dimensional folding of proteins may yield useful information about relationships too remote to be detected by conventional methods, such as sequence comparison, and may potentially lead to prediction of patterns and motifs in mutual structural fragments. With the exponential increase of structural proteomics data, the methods that scale with the rate of increase of data lose efficiency. Hence, new methods that reduce the computational expense of this problem should be developed. We present a novel framework through which we are able to find and align protein structure neighbors via hierarchical clustering and entropy-based query search, and present a web-based protein database search and alignment tool to demonstrate the applicability of our approach. The resulting method replicates the results of the current gold standard with a minimal loss in sensitivity (2.64%) in a significantly shorter amount of time (440x faster), while ameliorating the existing web workspace of protein structure comparison with a customized and dynamic web-based environment. Our tool serves as both a functional industrial means of protein structure comparison and a valid demonstration of heuristics in proteomics.**

I. INTRODUCTION

Predicting the function of a protein is key to understanding life at a molecular level. Conventionally, comparison of primary sequences of proteins has long been a widely-used method for detecting proteins that share a similar function and modeling phylogenetic trees [1, 2]. However, sequence comparison alone is not purposive for detecting distant evolutionary connections between proteins, which may elucidate useful functional information [3 - 5]. Sequence similarity, although simpler and more streamlined, is a far less accurate predictor of functional similarity than structure comparison, which is significantly more powerful for identifying cases where the evolutionary progress of the individual subject protein precludes the comparison of the protein sequences. Investigating structural similarity is also functional for analyzing cases with insufficient sequence similarity [6] or identifying mutual functional protein sites and motifs [7, 8]. Problems regarding similarity search also arise in many different applications in computer science, such as knowledge discovery and data mining (KDD) [9], vector quantization, and pattern recognition and classification [10]. Similarity search is not only fundamental to data science, but it also lies within problem domains in various other fields where data are not represented numerically, such as the statistical analysis of sets of molecular descriptors [11], investigation of cross-generational mutative patterns [12], and the representation of environmental sound waveform features [13].

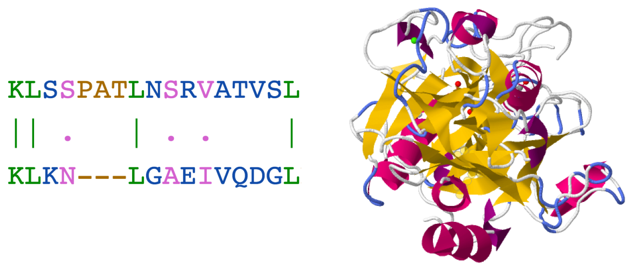


Fig. 1: Sequence (left) and structure (right) alignment of α-trypsin (1AKS) and flavodoxin (1AKR).

In the recent years, an explosion of newly discovered protein structures has been witnessed. Consequently, the Protein Data Bank (PDB), the largest protein structure database available online, has come to surpass 110,000 structural entries, with an increase of around 7,000 in 2015. As the amount of information in the PDB becomes overwhelmingly high, the need for methods to organize and classify structures for structure neighbor identification arises [14]. One method for the identification of structural neighbors is structural alignment, which attempts to discover similarity between two multiple protein structures based on their three-dimensional conformation. Performing structural alignment between the subject protein and all other proteins in the Protein Data Bank (PDB) via a structural alignment tool such as Protein Structure Comparison by Alignment of Distance Matrices (DALI) [15], Flexible Structure Alignment by Chaining Aligned Fragment Pairs Allowing Twists (FATCAT) [16], or Multiple Alignment with Translations and Twists (MATT) [17] can elucidate the discovery of the function of a protein of known structure but unknown function.

Many methods evaluate similarity between structures of

[1] Armand Hammer United World College of the American West (UWC-USA), Montezuma, NM 87731, USA

[2] Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA

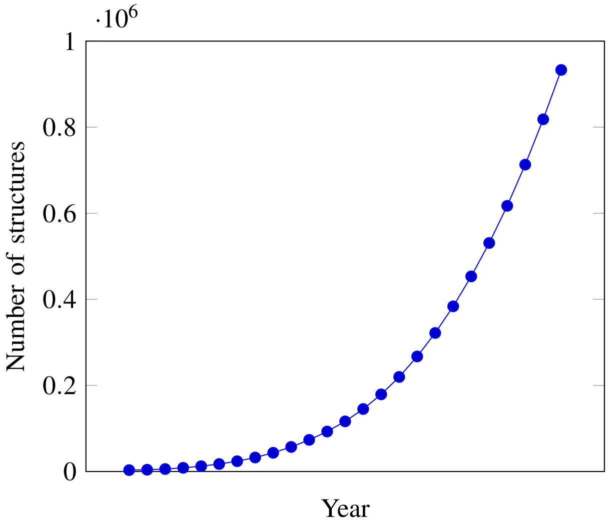


Fig. 2: Number of protein structures in the Protein Data Bank (PDB) between 1991 and 2015.

proteins via the comparison of α- or β-carbon coordinates of amino acids as a representation for the subject proteins. After the amino acids of subject proteins are paired, subject proteins are superimposed and the original similarity function is modified to obtain a final solution, often optimizing the distances between already paired amino acids while preserving the three-dimensional conformations of all subject proteins. The final solution is always quasi-optimal as the aforementioned problem has been proven, in principle, to be NP-hard with no exact solution [18]. Thus, it is possible to see the vast differences of heuristics and scoring criteria, as noted by Mayr et al. (2007), in all protein structure alignment methods, which diversifies the so-called ”optimal” solution for the structural alignment of a certain set of subject proteins [19]. One example of the heuristics used in alignment methods is the alternating approach to macromolecular docking: Many existing methods [15, 20 - 23] consider the subject proteins as rigid bodies, and attempt to minimize the deviation between the mapping of identified structures in all subject proteins while maximizing the number of mappings, describing the cumulative success in superimposition in terms of Root Mean Square Deviation (RMSD):

$$\mathrm{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\delta_i^2} \tag{1}$$

where δ is the distance between N pairs of α- or β-carbon atoms.

Menke et al. (2008) showed that introducing flexibility to structural alignment would enable to align better with particularly marginal three-dimensional states (the three-dimensional folding of a protein changes accordingly to the number, location, or character of ligands attached to it), which cannot be achieved through rigid-body structural alignment. Performing structural alignment while allowing a flexible-body conformational state with twists and translations also facilitates the management of pairwise distortions outside the stable backbone [17]. Menke et al. (2008) also provided a pairwise structural alignment tool, namely Multiple Alignment with Translations and Twists (MATT), to demonstrate flexible-body structural alignment. Nevertheless, flexible structural alignment is computationally more exhaustive than rigid-body structural alignment: Menke et al. (2008) reported that Multiple Alignment with Translations and Twists (MATT) runs in $\mathcal{O}(k^2n^3 \log n)$ time [17] compared to $\mathcal{O}(k^2n^3)$ during rigid-body structural alignment, reported by Konagurthu et al. (2006) [24], where k is the number of protein structures being aligned, and n is the primary structure length of the longest protein included in the alignment [17]. It is important to note that for small values of n, the discrepancy in running time is infinitesimal; but the aforementioned increase in protein structures in the PDB and the discovery of proteins of unknown function demand more exhaustive alignments, which in turn makes the rigid-body approach more preferable.

Looking at both commercialized and non-commercialized (open-source) pairwise and multiple structural alignment methods available, we report an arising problem in the daily use of these tools: Given the need for better methods to detect structural neighbors to a newly-discovered protein, the current methods do not offer a starting point to the user in terms of proteins of similar structure. Current methods take at least two protein structures as input, rendering the user optionless if the user has an unknown protein to which they would like to find structurally similar other proteins. In such a case, an exhaustive search through the PDB is optimal, as demonstrated by Holm et al. (1993) [15], and Godzik et al. (2004) [16]. Given the high-order time complexity of structural alignment, the running time of a full database search for a single query protein is too long to detect structurally similar proteins real-time (an average running time of 20 minutes up to an hour for a single query [25]). Protein structure database search methods available also assume a non-redundant library, neglecting the possibility of finding interesting structural and functional variability in newly discovered structures [26]. Furthermore, these techniques can only be executed for annotated structures in the PDB, which hinders the potential use of structural alignment in determining the function of an unannotated protein that do not yet have a structural neighbor in the library. None of the PDB search methods available today do a complete and exhaustive search of the PDB, which is computationally expensive and demanding as it requires naive $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ comparisons; Furthermore, to our knowledge, no protein structure search and alignment tool that produces statistically significant flexible-body structural alignments using the database hits currently exists.

The aim of this study is establish a novel, robust, and accurate framework for protein structure database search and multiple structural alignment, through which we are able to search the entire Protein Data Bank (PDB) to find structural homologs of an input protein, and conduct pairwise structural alignment with the nearest structural neighbors allowing flexibility in shape and sort the resultant structural neighbors in descending order of statistical significance. We also present a web-based database search and structural alignment tool with a dynamic user interface to actualize this approach. We build upon some techniques proposed by Budowski-Tal et al. (2009) [27] and Yu et al. (2015) [28], while genuinely optimizing both the database search and the structural alignment.

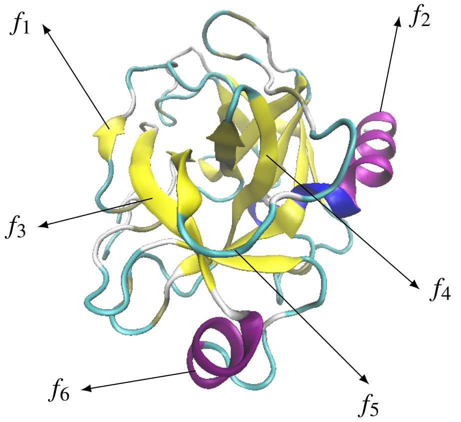
We approach the problem of searching the entire Protein Data Bank in an efficient way using the ”filter and refine” method [29], where a computationally inexpensive search is conducted to choose a small set of potential structural neighbors, followed by a more exhaustive and demanding alignment technique. Usually via the representation of structures as vectors in a k-dimensional hyperspace, techniques to search the PDB prove to be faster than raw structural alignment methods: Namely, Choi et al. (2004) use distance matrices inside the structure as a vector representation of frequencies, and quantize similarity by computing the distance between each vector (LFF) [30]; Rogen et al. (2003) utilize and adapt knot theory in algebraic topology and homotopy theory to engineer the Scaled Gauss Metric (SGM), representing structures as a vector of 30 topological quantities [31]; various techniques used a finite, orderly string of fragments, and consider the comparison and alignment of these vectors as a channel to detect structural similarity [32 - 34]. Although these filter methods easily outperform single-channel rigid structural alignment in running time, they report accuracy benchmark classifications that easily favor using full structural alignment over filter and refine methods: LFF, SGM, and PRIDE2 by Gaspari et al. (2005) [34] report classification accuracies of 68.7%, 69.1%, and 48.4% respectively, benchmarked to the SCOP protein classification database [35], which are easily outperformed by full (both rigid-body and flexible) structural alignment tools. We take a filter-and-refine method initially proposed by Budowski-Tal et al. (2009) [27], which represents objects as an unordered collections of local structural features, where the query is described as a vector of the occurrences of the structural motifs in the query protein backbone, as a starting point for a quick and accurate structural neighbor retrieval method.

Our best filter method easily outperforms other filter-and-refine approaches, and, more importantly, full-body structural aligners such as DALI [15] and FATCAT [16]; it can potentially be used to both efficiently, accurately, and rapidly identify good candidates of structural neighbors, and generate a set of potential structural neighbors for a protein with known structure and unknown function.

II. APPROACH

A. Parameter optimization.

We elaborate on the unordered vector space approach to protein structure similarity search proposed by Budowski-Tal et al. (2009), FragBag [27], which expresses proteins as a collection of frequencies of structural features with no specific order. FragBag utilizes a non-redundant library of structural fragments and motifs within proteins in the Protein Data Bank to collectively compute how similar two query proteins are, where the two query proteins are described as a collection of its contiguous structural fragments that overlap. The protein is then simply represented as a ”bag-of-fragments”, which is a k-dimensional vector that holds the frequency of each contiguous overlapping structural fragment as a separate entry within the vector.



$$\vec{v_p} = [s(f_1), s(f_2), s(f_3), s(f_4), s(f_5), s(f_6)]$$

Fig. 3: A simplified FragBag representation of porcine α-trypsin (1AKS). After each unique contiguous backbone fragment of the query protein is identified, the occurrences s of each segment is counted and put as entries in a k-dimensional query vector. For porcine α-trypsin (1AKS), the resultant FragBag vector $v_{\vec{AKS}}$ would have six non-zero dimensions: [3, 1, 3, 6, 11, 1]. Note that all 400(11) library FragBag vectors put in a high-dimensional vector space representative of the protein structure dataset S are 400-dimensional; yet, only the structures that are present in the backbone have a non-zero occurrence s.

FragBag ultimately notes two benefits to representing a query protein in terms of its backbone segments:

- To implement a fast and efficient estimation of full database search of the Protein Data Bank;
- For structure predictions where there is partial (fragment-wise), albeit no complete structural similarity.

Both of these aspects are favored in any approach to structural similarity; yet, we provide conceptual and methodological motivation to select FragBag as a starting point over other filter-and-refine methods: Namely, during the task of similarity search for structural neighbors in the high-dimensional vector space S constructed by representative data points, as we widen our search radius, the vector space tends to not drastically increase, showing point density. This is to be elaborated later, particularly in II.B.

FragBag remains to be the gold standard in filter-and-refine methods mimicking full-body structural aligners, identifying structural neighbors on a par with some state-of-the-art full-body structural aligners within the 400(11) non-redundant structural fragment library. Budowski-Tal et al. (2009) reports robustly generated results from similar rankings with varying definitions of structural similarity, and report a cumulative success of identifying over 75% of true positive structural neighbors benchmarked to the SCOP classification database [27] in all distance metrics benchmarked, with a maximum score of 89% within the 400(11) non-redundant library.

B. Mechanics of entropy-based search.

We generalize and justify a novel method that can be applied to not just similarity search within potential structural neighbors of a query protein but any kind of exhaustive search of -omics data. We approach the task of similarity search within the protein structure dataset by defining search performance in terms of the difference in novelty between new data and already existing data, *entropy*, hence the name ”entropy-based clustering”. This approach enables the re-construction of the protein structure dataset, so that both the amount of time and space that the task of similarity search requires are linearly proportional to the entropy of the dataset, thus sublinearly proportional to the increase in size of the dataset itself.

We define two key elements of a dataset that will be vital to our approach: *metric entropy* and *fractal dimension*. We provide rigorous definitions for both of the concepts in Appendix A, but we are able to succinctly define metric entropy as describing the amount of dissimilarity a database appears to provide within itself, and fractal dimension as describing the relation between the number of clusters needed to cover all unique data points in a database and the respective radii of the clusters (note that the concept of metric entropy is different from that of a distance metric, which is a measure of distance applicable to any database). Via the analysis and generalization of these two elements of any database, we show that if the database in question (in our case, the protein structure database) appears to exhibit asymptotically low metric entropy and fractal dimension, the method outperforms naive full-body structural alignments and heuristically optimized filter-and-refine methods. The main advantage of optimizing entropy-based clustering and similarity search using metric entropy and fractal dimension is that it allows for a mathematical, instead of an experimental, evaluation of the approach in terms of efficiency and accuracy. We also show that the entropy-based clustering approach results in zero loss in sensitivity.

The entropy-based similarity search is a quasi-k-means clustering algorithm, followed by an n-step hierarchical search that consists of four main steps.

1) Exhaustively analyze the database entries and define a high-dimensional vector space S, mapping each unique database entry onto unique points in this space S.
2) Use this space S and a quantized measure of similarity to group data points into clusters.
3) To search for potential candidates, perform an initial search to identify the clusters that could possibly contain similar data points to the query.
4) Do a search within these clusters to find the closest data points to the query.

We support this initial framework of hierarchical search by providing a conceptual supplemental rationale. We verbally approximate entropy as the vector distance between data points in this high-dimensional vector space S (protein structures represented as vectors); hence, if S exhibits low entropy, data points added to S tend to not be distant from points already existing in S. We quantize the distance between data points in S using generic distance metrics: Namely, Euclidean and cosine distance:

For given queries $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$, the Euclidean distance is:

$$d_e(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{2}$$

and the Cosine distance is:

$$d_c(p,q) = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}} \tag{3}$$

We experimentally evaluate both metrics in our process of creating a hierarchical clustering framework later.

Many studies defined the Protein Data Bank to be highly redundant [36 - 40], but it is vital that we define what being redundant for the protein structure data set signifies. Smith et al. (2015) posited that many of the data points of protein structures may be exact duplicates; that is, they may appear to have the same construction of representative vectors after the filter-and-refine method. This case can easily be solved by using a non-redundant library of proteins, as demonstrated by Ye et al. (2004) [16]. Maybe the representative data points exist in only a minimal number of dimensions; namely, showing low-dimensionality. If the dimension of the high-dimensional vector space is low enough, it can be divided into countable units, which would then ameliorate the running time for similarity searches. Although it is important to note that for datasets where empty space fills the vector space so that data points are sparsely distributed, many empty cells will be included in the search, significantly increasing running time. Furthermore, Yu et al. (2015) noted that many biological datasets do not reside in low-dimensions; instead, they arise from a highly chaotic high-dimensional vector space, properly characterized as a ”tree of life” [28]. In these datasets, local low-dimensionality can be observed while the whole vector space is high-dimensional. This local low-dimensionality can be harnessed to search through specific regions of low-dimension with entropy-based similarity search.

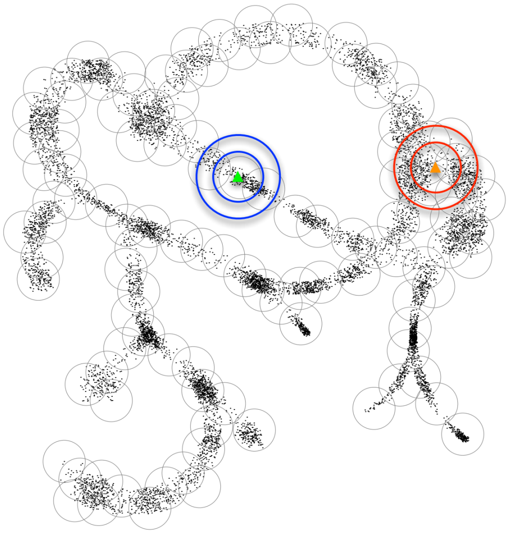


Fig. 4: Depiction of representative data points in an arbitrary high-dimensional vector space exhibiting local low-dimensionality and global high-dimensionality by Yu et al. (2015). Reprinted from [28].

As seen in Figure 2, -omics data reside in high-dimensional vector spaces, but given a coarse scale to observe in this high-dimensional space, the density and the distribution of the representative data points are almost unidimensional. It is important to note that, based on the aforementioned succinct definition of fractal dimension, the high-dimensional vector space S exhibits low local fractal dimension within the blue circles around the green query, and high fractal dimension within the red circles around the orange query. The blue circles around the green query point illustrate low fractal dimension: the larger-radius circle contains only linearly more points than the smaller one, rather than exponentially more. In contrast, the red circles around the orange query point illustrate higher local fractal dimension.

Approaching this juxtaposition of different amounts of dimensionality in the same database, we may be able to exploit the local and global discrepancy: Using a wide enough search radius, the entropy-based approach assumes a unidimensional space, and as the search radius is reduced, we can gradually start to consider the branches of high-dimensionality. For this high-dimensional vector space S, we assume a coverage with spheres with radius r_c, where r_c is equals the low-dimensional branch width in question, and

$$E_r = \sum_{i=1}^{k} S_i \tag{4}$$

where E_r is the metric entropy of the high-dimensional vector space S and S_i is the number of spheres needed to cover S. We note that by the use of a spherical cluster, we assume that the points on this high-dimensional vector space S are very close, thus can be encoded in terms of one another, which is in parallel with the aforementioned redundancy in the Protein Data Bank. Thus, using the triangle inequality, we are able to search all points within r by only looking at adjacent spheres with cluster centers within $r + r_c$ of the query. For a wide search radius, we see that the spheres needed to cover S extend along a minimal number of dimensions: As the radius decreases, the depth of the sphere matters. We call this property of local low-dimensionality the fractal dimension d of the high-dimensional vector space S at the radius scale r_c. Local fractal dimension d in S is computed via the increase between radii r_1 and r_2 over the points added to S_{r_1}, let us call these new points n_2 and previous points n_1, by the increase in radii. The local fractal dimension is then simply:

$$d = \frac{\log(n2/n1)}{\log(r2/r1)} \tag{5}$$

Intuitively, we note that when $d = 1$, the number of points added to the sphere S_{r_1} is linear to the increase in radii, thus, our approach is maximally efficient when the fractal dimension d and metric entropy k is minimal. Yu et al. (2015) reports $2 < d < 3$ for average local fractal dimension of FragBag vectors [28], thus providing a rationale to investigate our approach over the protein structure high-dimensional vector space. When we search in a wider radius around a query, the number of points added to the spherical cluster covering the points around the query within the radius grows exponentially with the fractal dimension; implying that this growth will not hinder the efficiency provided by an entropy-based search. We provide a rigorous investigation in Appendix A: Theoretical Foundations that given a high-dimensional vector space S with a fractal dimension d, an entropy k, and an initial search radius r_c, the time complexity $T(n)$ of the entropy-based similarity search is

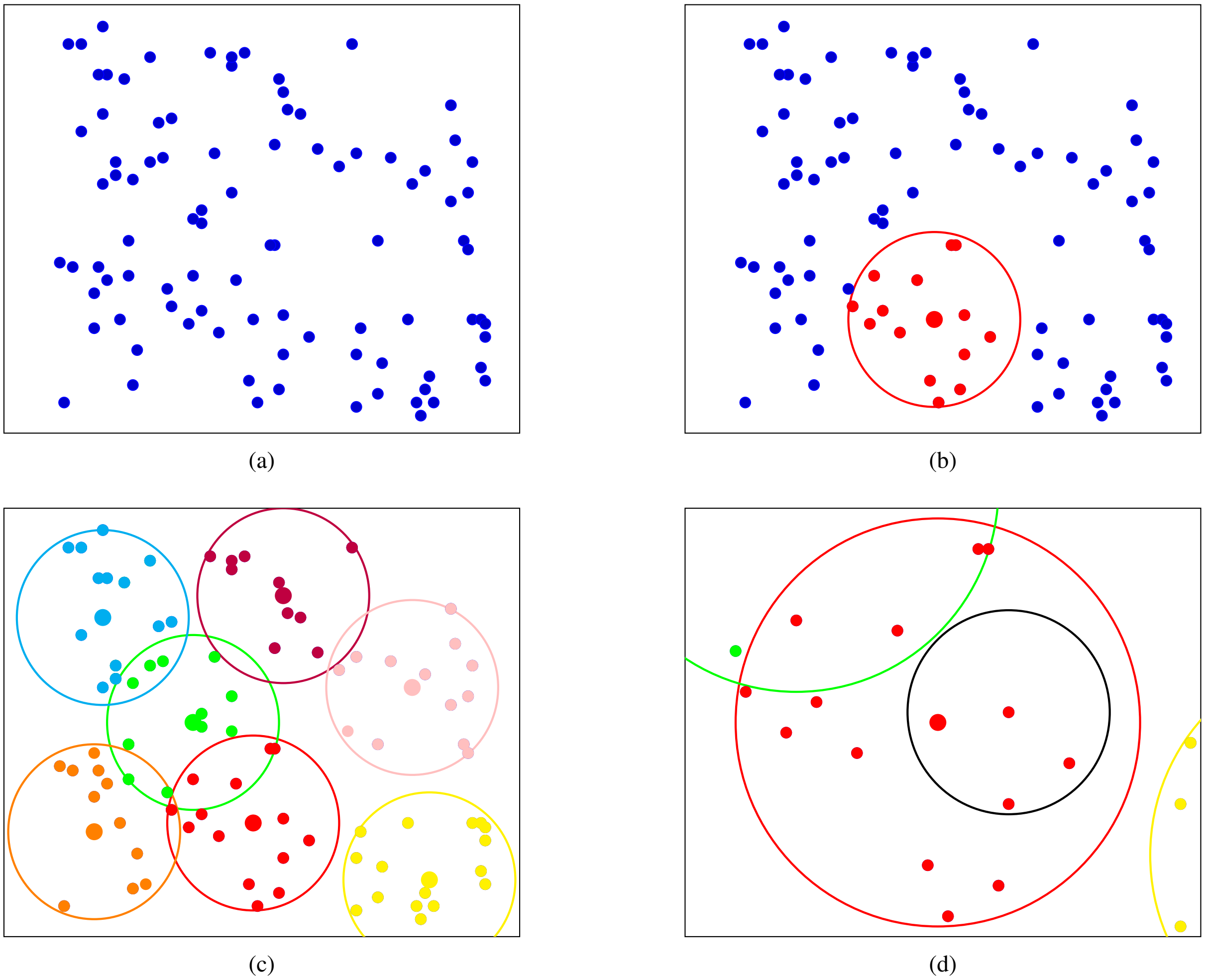
$$T(n) = \mathcal{O}(k + D_C(q,r)(\frac{r+2r_c}{r})^d) \tag{6}$$

which is asymptotically linear to k with minimal values of output size and fractal dimension.

C. Clustering algorithm.

When the problem involves comparing properties of a data point to one another, the most straightforward way to reduce computational expense is to cluster already existing data points in the database prior to the search. To reduce the running time for a query as much as possible, we used hierarchical divisive clustering as it offers a complexity reduction from $\mathcal{O}(n)$ to $\mathcal{O}(\log\ n)$ for query search. The goal of our approach is to balance the computational expense on both sides; for a reduction from $\mathcal{O}(n)$ to $\mathcal{O}(\log\ n)$ for query search compensates for $\mathcal{O}(2^n)$ on our side since the user can withstand much less computational burden.

We present a novel algorithm that relies on the hierarchical exhaustive logic that drives every single data point to eventually be part of a cluster. Once the real-world objects (proteins) are expressed as vectors and are put


(a)

(b)

(c)

(d)

Fig. 5: A simplified visual representation of the entropy-based flat divisive hierarchical clustering algorithm utilized. Each protein is represented by a unique data point in a high dimensional vector space (a). A random cluster center k is chosen and the points within a radius r are put in this cluster (b). Then, iteratively all points are clustered via the same approach, and this process repeats until all points are in a cluster (c). Once the first level of clustering is done, data points within a cluster are clustered with a smaller radius r' (d). Note that the data points at the intersection of overlapping clusters are assigned based on the order of clustering: In (a), the red cluster is the first cluster formed, thus containing data points that are also in green and orange clusters.

in a high-dimensional vector space S (the whole Protein Data Bank), an initial cluster center k is randomly chosen in this high dimensional space S. k is a data point in S itself. A user defined radius of r defines a circular grid in which all the points are considered a part of the cluster by the user. In other words, once k is chosen in S, the user decides to what extent should the data points be represented by the cluster center k by giving r as an input.

A data point i is part of the cluster C_k with the cluster center k, where the distance value between i and k $D_{ik} \leq r$. Thus, a cluster is formed based on a user-defined r. Then, the cluster is temporarily removed from S and another randomly chosen cluster center k' is investigated for data points with distance less than r. This is repeated until every data point i in the high dimensional vector space S is part of a cluster with a center k. It is irrelevant to project the amount of initial clusters after input r since it very much depends on the data set and the magnitude of r and does not affect the output since the only prerequisite for the next step is to make sure every data point is part of a cluster.

Once every data point in S is a member of a cluster, the dummy timer $t = t + 1$, and the user-defined radius r is reduced by a user-defined function $R(r)$ where $t = t + 1 \rightarrow R(r) = r'$ for both the search and the clustering. Some options of r is automatically given to the user for the query search, such as $\sqrt{r}$ and $r/2$, and users are able to define $R(r)$ as long as it outputs a value where $r' < r$. Within each cluster, another randomly assigned cluster center k' and the decreased radius r' is put in to find data points similar to k' and include them in the new cluster $C_{k'}$ where now data point $j \in C_{k'}$ if $D_{jk'} \leq r$. Once this process is done, the dummy timer $t = t + 1$.

```
**Require:** $r \neq 0, r \in \mathbb{Z}^+$;
**Ensure:** all points $n \in S$;
$k \leftarrow$ random point in $S$;
$C_k \leftarrow$ cluster with center $k$;
$D_{ik} \leftarrow$ distance between $i$ and $k$;
*top*:
**if** $\exists C_k \mid n \in C_k$ **then quit**;
**else**
*loop*:
    **if** $D_{nk} \leq r$ **then**
        $n \in C_k$;
    **goto** *loop*;
    **close**;
$r \leftarrow r'$;
**hide** $C_k$;
**goto** *top*.
```

Fig. 6: Pseudocode for the hierarchical clustering algorithm.

Once all the clustering within all existing clusters completes, the clustering continues with radius r getting reduced by both the user-defined function $R(r)$ and the dummy timer $t = t + 1$. After each data point in the vector space S is a cluster within itself, the process stops. t yields the number of levels of clustering that can depend on the data set and radius r and also, incidentally, the level of the deepest point (the point within the most number of clusters). The algorithm assumes that the system is unsupervised in the sense that it will always reach the static state where there is no clustering that needs to be done. To make the algorithm more flexible, an optional user-defined operator d is introduced where d denotes how deep the clustering should go. The algorithm quits when $d = t$.

Ultimately, the high dimensional vector space S will include at least n clusters for n data points, and probably more. Yet, this final specificity is redundant as the only member of the final cluster is the data point itself: When d is specified, the algorithm quits at $d - 1$ to prevent redundant search of clusters C_n where $n = 1$. Once a query is submitted to the system, cluster centers are identified in a user-defined radius r. Cluster centers which are in this circular grid of radius r is only searched when $t = t + 1$ for radius r'.

III. Results

A. Scaling behavior.

To obtain an optimal distance metric to be used in our entropy-based search, we investigated the increase in speed between both Cosine and Euclidean distance functions. The resulting values imply that the acceleration across metrics depend largely on the initial cluster radius. We produced databases for initial cluster radii of 0.1, 0.2, 0.3, 0.4, and, 0.5 and 10, 20, 25, 50, and 100, for Cosine and Euclidean distance respectively, with a unit change of 0.01 (50 unique values of cluster radius in total). We ran similarity searches for 4 query proteins with varying size and number of structural neighbors: Porcine α-trypsin (1AKS), flavodoxin (1AKR), HIV-1 reverse transcriptase (1FKO), and hemoglobin-A (1ASH). We used reduction functions $\sqrt{r}$ and $r/2$ for $R(r)$; yet, due to the minimal change in results, reported an average of the two functions. We ran each query protein 5 times through the entropy-based structural similarity search for each cluster radius and included the averages for each cluster radius as one data point in our investigation.

TABLE I: Clustering time using Cosine distance

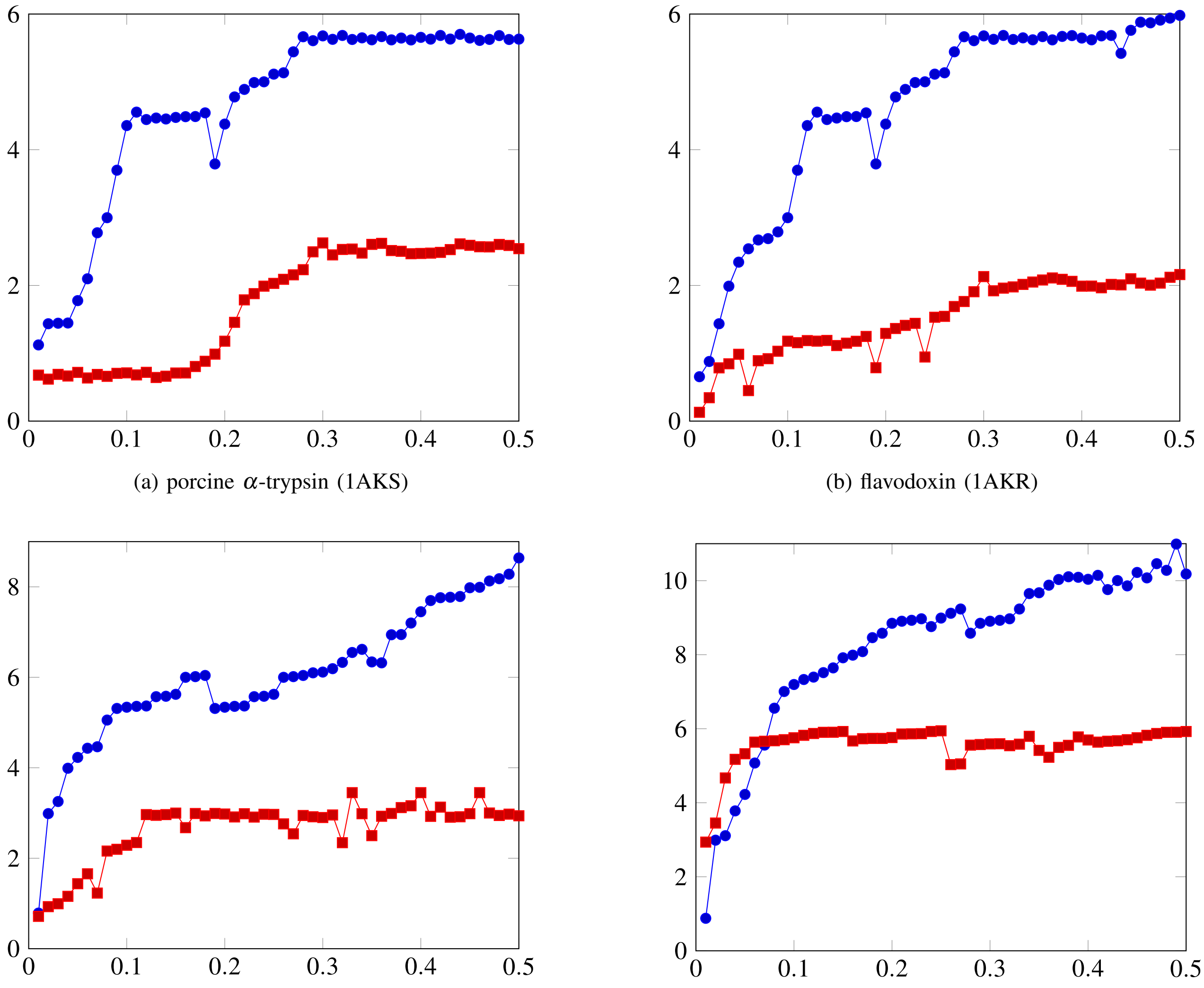
Radius	0.1	0.2	0.3	0.4	0.5
Clustering time (s)	22546	13243	7645	5433	4087

TABLE II: Clustering time using Euclidean distance

Radius	10	20	25	50	100
Clustering time (s)	24407	3154	679	204	87

We also report a comparison of clustering time of both Euclidean and Cosine distances with aforementioned unique cluster radii. We compared the average clustering time over the whole high-dimensional vector space S across five trials for each unique cluster radius. We used all 20 threads on an Intel Xeon x5690 (12-core) while clustering, compared to only one during search. We have concluded that the running time for the task of similarity search in our entropy-based hierarchical clustering largely depends on the magnitude of the initial cluster radius r_c, and the number of structurally similar proteins that are around the query protein in the high-dimensional vector space S. As we demonstrate using the aforementioned 4 test query proteins with different sizes and degrees of membership in S, the acceleration across uniquely different values of r_c decreases in different rates. We posit that this effect is most likely to be caused by the high-dimensional vector space S having a lot of local density regions that are distinctly separated from other more uniformly distributed regions (being more ”spiky” in largely populated, dense regions compared to peripheral regions). A protein that is a neighbor to many others in a densely populated region shows little to no acceleration with an increase in radius since a small increase covers a large part of the neighbors, destroying any acceleration. Modifying r_c may enable the user to effectively search for proteins that are less likely to be a part of these aforementioned dense populations; yet, as r_c increases, the running time increases proportionally.

Based on our investigation of scaling behavior, we note that Cosine distance generally yields better acceleration, and the Euclidean distance guarantees 100% sensitivity via Triangle Inequality [28]. Hence, we will be using Cosine distance as our speed benchmark candidate and Euclidean distance for our accuracy benchmark candidate.



(a) porcine α-trypsin (1AKS)

(b) flavodoxin (1AKR)

(c) HIV-1 reverse transcriptase (1FKO)

(d) Ascaris hemoglobin (1ASH)

Fig. 7: Comparative scaling behavior of two different distance metrics, Cosine (red) and Euclidean (blue) used in entropy-based clustering and search (x-axis denotes the initial cluster radii, and y denotes running time in seconds). It is important to note that across a cluster radius of 0.5, d ultimately approaches zero, providing the motivation to investigate redundancy within the protein structure high-dimensional vector space (as the cluster radius increases, more points are introduced to the search, but the rate in which these points are introduced decreases rapidly, approaching zero).

B. Benchmarking classifications

We evaluated the accuracy and running time of our method by benchmarking the raw results to gold standards in protein structure similarity search; ideally, our method would outperform filter-and-refine methods (that are compromised in accuracy compared to full-body structural aligners) in speed, and outperform full-body structural aligners (which are compromised in speed compared to filter-and-refine methods) in accuracy. We have established the respective sets of gold-standards as being the most successful performers of both groups: FragBag [10], SGM [31], and TM-Align [41]; and DALI [15], FATCAT [16], and 3D-BLAST [42], for speed and accuracy benchmark classifications respectively. For the following results, we benchmarked our method, Esperite, along with other respective methods for both types of benchmark classifications.

When comparing techniques, it is vital to realize that the monitoring of the relative performance of two methods rely heavily on not only the nature of the query and the database, but also the heuristics used in the technique that the subject method is being benchmarked to. We assume an independent classification database, the SCOP classification database [35] to distinctly separate our method from constructing derivations or improvements on other existing methods. We assume that these protein structure classification databases provide perfectly similar protein structural neighbors as the gold standard (no false positives or negatives). We use the SCOPe_FAMILY group as the definition of closely related proteins, and the fold and super-family groups for distant proteins. We utilize and evaluate receiver operating

TABLE III: Exhaustive structural comparison running time (s) for Esperite and other benchmark filter-and-refine methods

	FragBag	TM-Align	SGM	**Esperite**
Processing	1324	3409	3386	**992**
Running	1657	6876	113	**1312**
Total	2981	10285	3499	**2304**

characteristic (ROC) curves as a binary computation of the performance of classifier methods: The curve depicts true positive rate (TPR) (number of SCOP structural neighbors of the query found) vs. the false positive rate (FPR) (number of SCOP structural non-neighbors of the query found) at various thresholds. To cumulatively present the accuracy of a method, we averaged the benchmarking results both across different queries and different classification methods within the classification database. We trained and evaluated the accuracy of Esperite across SCOPe_FAMILY (2,623 domains in 903 superfamilies; 591 folds).

1) Search benchmarks.: For benchmarking over searching for structural neighbors across the SCOP database, we plot the fraction coverage of the whole database versus the cumulative amount of false positives found, as a derivation to the traditional receiver operating characteristic (ROC). We plot the quasi-ROC curve for all SCOP subdatabases: for family, super-family, and fold classification levels.

2) SCOPe classification correlation: Based on the existing classifications in the SCOPe classification database, we assessed the performance in replicating protein structural neighbors within SCOPe. We evaluate the receiver-operating characteristic (ROC) curve and precision-recall curve (PRC) analysis. The receiver-operating characteristic curve describes the amount of true positives identified against the amount of false-positives. The precision-recall curve, as the name suggests, draws a correlation between the precision against the overall coverage of the classification database. We used the area under the ROC curve (AUC) and the area under the PRC (AUPRC) as measures of agreement with the SCOPe classification.

3) Computing time benchmarks.: We compare how efficient Esperite is compared to other filter-and-refine methods Esperite is being benchmarked to in performing an exhaustive structural *NxN* comparison of the 2,623 entries in SCOPe_FAMILY.

C. Correlation with Multiple Alignment with Translations and Twists (MATT)

Since Esperite only provides a list of potential structural neighbors, a full-body multiple structural alignment tool is needed to maximize the functionality of the web-based environment. We integrated the tool Multiple Alignment with Translations and Twists (MATT), engineered by Menke et al. (2008) [17] into the Esperite kernel, where we are able to conduct full-body structural alignment after narrowing the high-dimensional protein structure vector space down to a quite precise list of potential structural neighbors.

Yet, it is crucial to make the distinction to need full structural alignment, and offer full structural alignment to the user. Hence, we conducted a correlation study between Esperite and MATT to decide whether or not the web-based tool need to run full structural alignment to get as accurate results as a computationally expensive structural alignment tool like MATT.

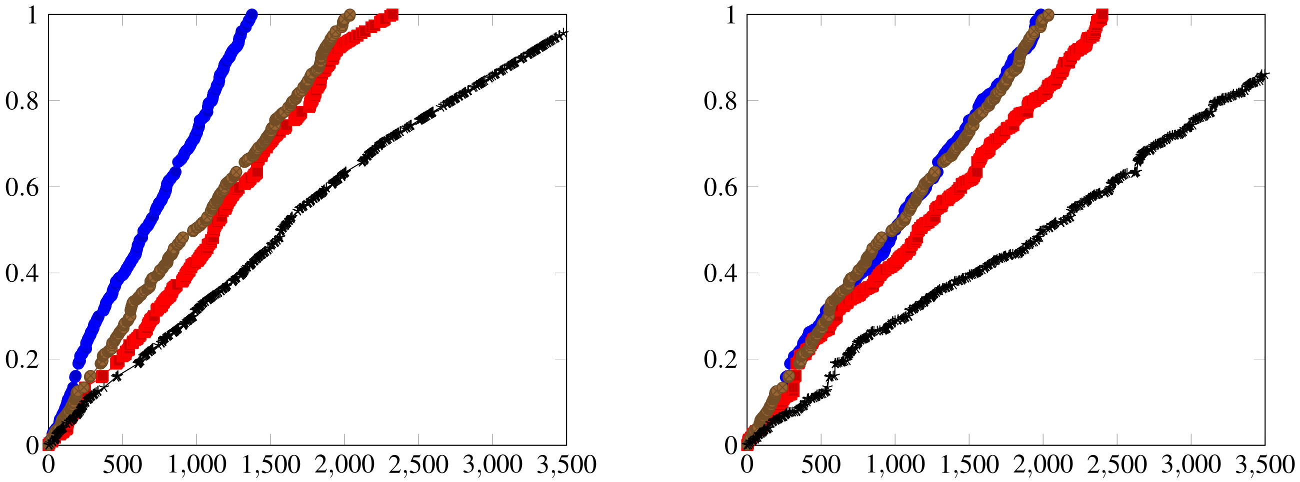
MATT uses a unit length (block) that is the set of α-carbon atoms in between five and nine amino acid residues in a protein. Given a block B, b_h is the very first residue and b_t is the very last residue across the block. For a pair of blocks BC, T_CB is the minimum RMSD transformation to trigger the second structure to align its α-carbons to the first, and RMS_T is the RMSD of the two blocks under T. Then for BC,

$$Score(BC) = -logP(RMSD_T) \tag{7}$$

Since both the $p-$value and the Cosine distance is bounded by [0,1], and Cosine distance is our accuracy benchmark classifier, we used a sample size of $N = 250$ hits for trials with 5 different unique proteins to measure Cosine distance hits, executed MATT on the hits, and averaged the values.

IV. ESPERITE: THE WEB SERVER

We present Esperite, a web server devoted to applying our entropy-based clustering and search for protein structure neighbors in a user-centered, web-based environment. Esperite serves as a real-time web-based tool through which structural neighbors can be identified in a fast, efficient and computationally expensive way. Using this algorithm, the bag-of-words representations of the proteins in the Protein Data Bank are clustered for fast and efficient structure homolog search in the database. Esperite is freely available to the public and is hosted by the MIT servers. The web server runs real-time commands for database search - so that the tool is available when the files hosted on the server are damaged or deleted. Indeed, the files are retrieved directly from the Protein Data Bank (PDB) and uploaded to the server without any prior modification. Whenever the pdb file format and/or content changes in the database, the PDB file used on the web server is updated and uploaded automatically. Every user starts a session with a unique session ID which would then be used to reach results without entering parameters. Once the session is over, the temporary files are deleted from the local host. All session IDs are contained in a log file. Esperite is up and running at http://esperite.csail.mit.edu and the source code is also available on GitHub.



(a) SCOPe_fold

(b) SCOPe_family

Fig. 8: ROC curves plotted over false positive rate (x) and fractional coverage over the database (y), for Esperite (blue), DALI (brown), FATCAT (red), and 3D-BLAST (black).

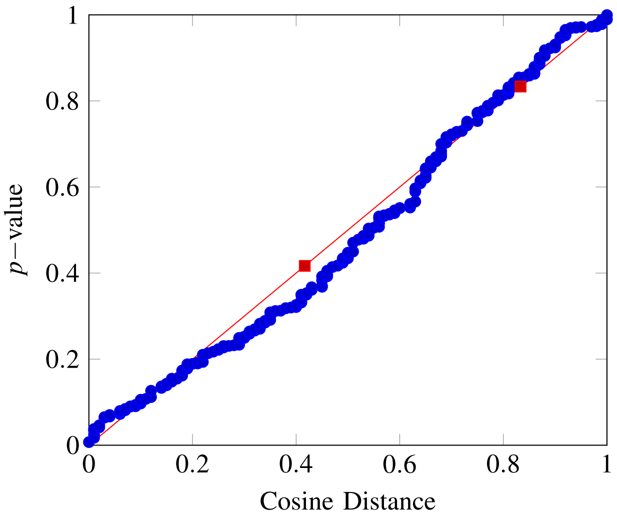
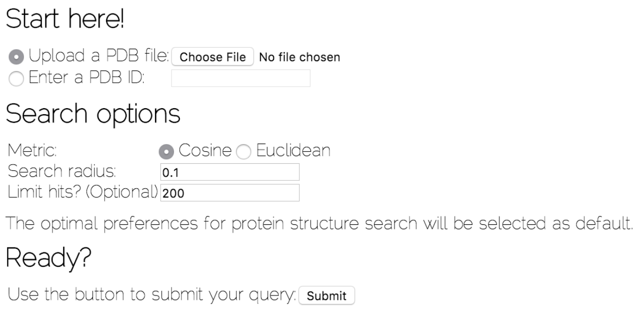


Fig. 9: $p-$value generated by Multiple Alignment with Translations and Twists plotted with Cosine Distance for the same sample size by Esperite. $R^2 = 0.87$.



Start here!

● Upload a PDB file: Choose File No file chosen
○ Enter a PDB ID:

Search options

Metric: ● Cosine ○ Euclidean
Search radius: 0.1
Limit hits? (Optional) 200

The optimal preferences for protein structure search will be selected as default.

Ready?

Use the button to submit your query: Submit

Fig. 10: Query interface of our web-based ultrafast protein structure search tool, Esperite.

A. Server Workflow.

1) Input: A PDB file contains the atomic coordinates for either a protein or other biomacromolecule. Using different methods, structural biologists determine the location of each atom of the molecule on the plane relative to one another, annotating and finally releasing the resultant data as a PDB file to the public. The local shell command takes a PDB file as an input for the structure search. The web server also allows for identification of the input PDB file remotely from the Protein Data Bank itself: The user is presented with the option of entering the PDB ID which is then sent to the Protein Data Bank to locate the actual PDB file. The PDB file is then uploaded temporarily to the local host and is deleted once the users session ends. Once the file is uploaded to the local host, the database search shell command is run from the server.

2) Search options: The web server offers options to customize the database search for structure homologs of a PDB file. These options include: The metric that will be used to express distance between two BOW vectors (Cosine distance or Euclidean distance), the initial search radius r (as an integer), the function that the initial search radius will be replaced with (current options include $\sqrt{r}$, and $r/2$), and the depth of divisive clustering d (d also denotes the level of the last cluster created). The user is also able to enter a function of their own (provided that it complies with the syntax and the range of the dynamic radius r'). The user is also free to choose the default settings already set where $r = 10000$, $r' = \sqrt{r}$, used metric is Euclidean distance and d is not specified.

3) Output: After the job is submitted, the web server will display a self-refreshing page with the task status until the job is completed. All files generated during the session

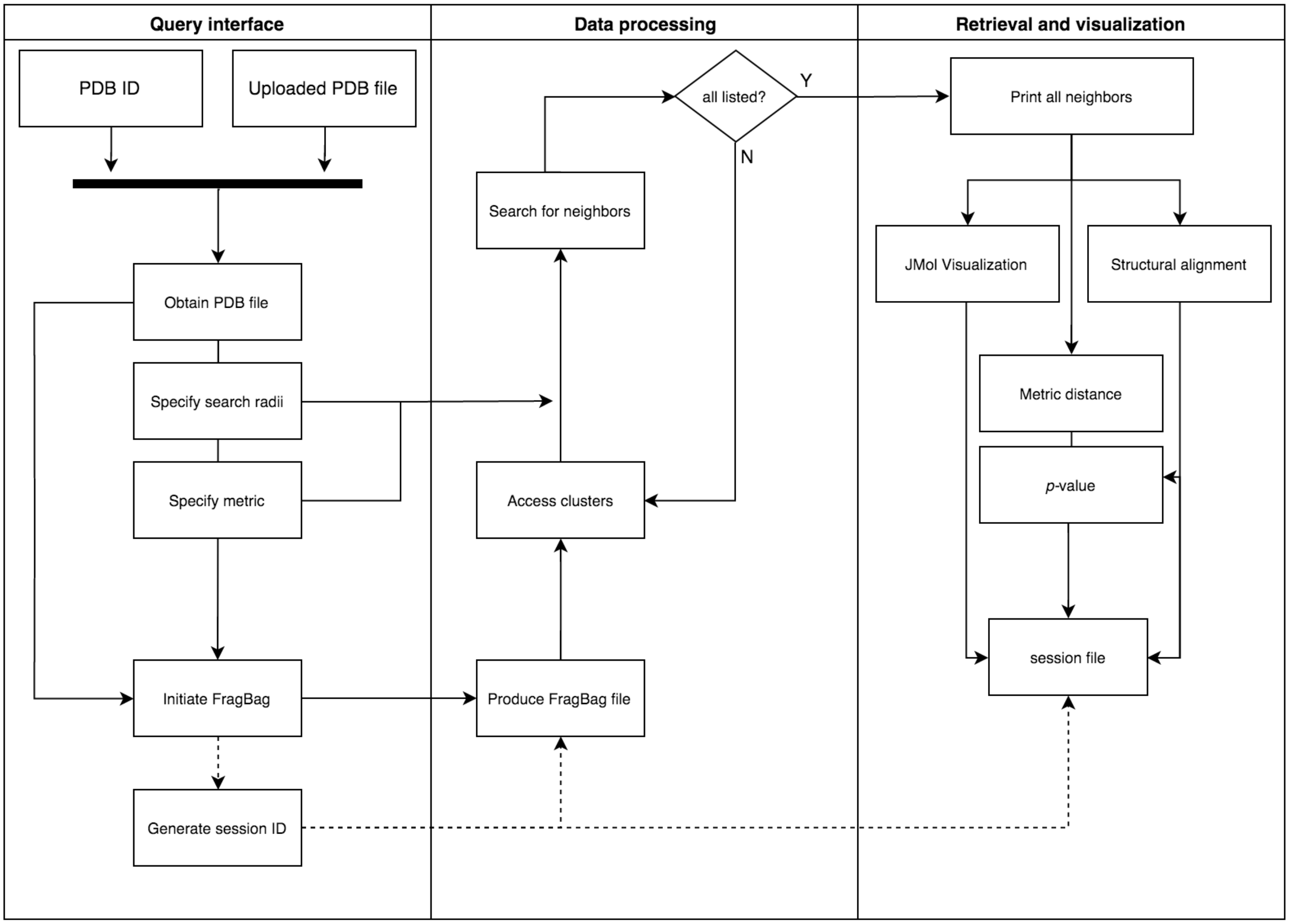


Fig. 11: Query interface of our web-based ultrafast protein structure search tool, Esperite.

are marked with a unique ID for future reference, which is reported to the user. The output window consists of a set of drop-down menus to display the PDB output file of each hit, to run MATT for each hit and the input protein, and to see the PDB file in a new tab. A dynamic AJAX JSMol window is also embedded to the results screen so the user can access the real-time multiple flexible alignment quickly.

V. Conclusion

It has been recognized that predicting the function of a protein is key to understanding life at a molecular level. Sequence comparison, which is the most frequently used method to predict the function of a protein of known sequence but unknown function, has failed to identify relationships where the subject proteins may exhibit differences in function, regardless of the significant similarity in sequence. These relationships, hence, need to be investigated through another method which would ensure prediction of a function of an unannotated protein with We introduced a novel entropy-based hierarchical framework to protein structure database clustering and search, allowing for linear-time similarity search even with an exponential increase in data. We rigorously proved that our approach scales linearly with the entropy of the database, thus sublinearly with the database itself.

Given the drastic exponential increase in size of the Protein Data Bank, the biggest protein structure database online, the need for methods to scale in a sublinear proportion with the increase of data arises. We present an approach through which we are able to exploit the redundancy present in the Protein Data Bank, which has a plausible tendency to exhibit low-dimensionality on a local level.

The main motivation behind the applicability of our approach is the minimal metric entropy and fractal dimension the protein structure database exhibits, which is demonstrated by the scaling behavior of the Cosine and Euclidean distances with increasing initial cluster radius.

Furthermore, we contribute to the arising interest and knowledge on already existing filter-and-refine protein structure alignment methodology by a number of ways:

- To our knowledge, our approach is the **fastest** filter-and-refine protein structure alignment method to date, outperforming even state-of-the-art full-body structural alignments by a large margin.

- To our knowledge, it is the **most accurate** filter-and-refine method that preserves the accuracy of the state-of-the-art full-body structural alignments to date.
- It is the **only** approach to date whose time and space complexity bounds can be mathematically justified without any experimental background.
- It is the **only** approach to scale sublinearly with the protein structure database to date, allowing for control over the increase of the database.
- It is the **only** approach that allows for the discovery and investigation of structural neighbors of a newly-discovered protein, allowing for interesting biological elucidation.
- It is the **only** approach that can construct structural neighbor families and superfamilies *ab initio*, i.e., with no prior training with or starting from experimental data.
- The web server Esperite is the **only** protein structure alignment server who offers full flexible structural alignment using hits from a filter-and-refine method.

When we discuss the problem of finding the structural neighbors of a protein while the amount of information that needs to be processed increases, we are inclined to stay in the concept of proteomics. Yet, our approach is easily implementable to and versatile for any other ”big data” problem our society is facing today. Any -omics data that are bounded in an exponentially increasing rate can be controlled and harnessed using entropy-based hierarchical clustering.

VI. FUTURE WORK

A. Thorough benchmark classifications

Esperite is currently being thoroughly investigated by the Computer Science and Artificial Intelligence Laboratory (CSAIL) at the Massachusetts Institute of Technology (MIT) and the Department of Computer Science at Tufts University Graduate School of Engineering, both of which have endorsed Esperite to be the protein structure discovery tool of the respective computational biology labs in order to conduct more thorough benchmark classifications and detailed engineering of the tool.

B. Machine learnable fold space

Corral-Corral et al. (2015) raise an important question: Given two representative data points of real-world objects, would the existence of a proximal similarity based on the dimensionality of the high-dimensional vector space in which the data points are clustered imply that the aforementioned two real-world objects are, in fact, similar? [43] In order to begin to address this very frequently appearing issue in clustering and similarity search, Corral-Corral et al. (2015) posit that the lower and upper boundaries by which the proteins with different folds are surrounded can be investigated and numerically obtained via the training from empirical data and machine learning approaches.

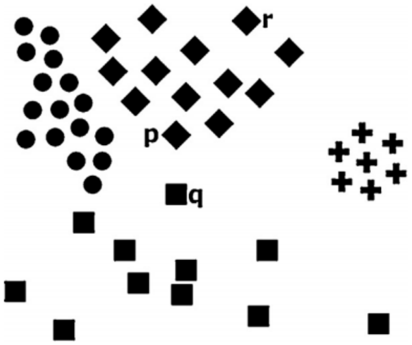


Fig. 12: A visual depiction of the problem initially raised by Corral-Corral et al. (2015), where proximity in a high-dimensional representative vector space does not necessarily imply membership to a specific class. Rhombus class members *p* and *r* has a greater distance in between than that of *p*and square class member *q*. This high-dimensional vector space is already clustered, and intuitively, we note that any characteristic of the given cluster is highly effective on the proximal errors a high-dimensional vector space can yield. Reprinted from [44].

C. Working towards predicting the function of uncharacterized proteins

We have established earlier that one of the biggest advantages of our entropy-based clustering and search method for protein structure comparison is that we are able to find structurally similar proteins to a protein of newly-discovered structure. Esperite is able to find structural neighbors *ab initio*, so no empirical data on the structure of the query protein is needed. The Protein Data Bank is filled with proteins with newly-discovered structures every day, and Esperite appears to be the optimal utility to begin to identify and/or predict the function of various proteins.

Using Esperite, we identified four uncharacterized proteins whose function can potentially be elucidated by further investigation. On the protein structural neighbor pairs shown in Figure 10, we continue our investigation at the New Mexico Highlands University (NMHU) in Las Vegas, NM, using nuclear magnetic resonance spectroscopy.

VII. AVAILABILITY

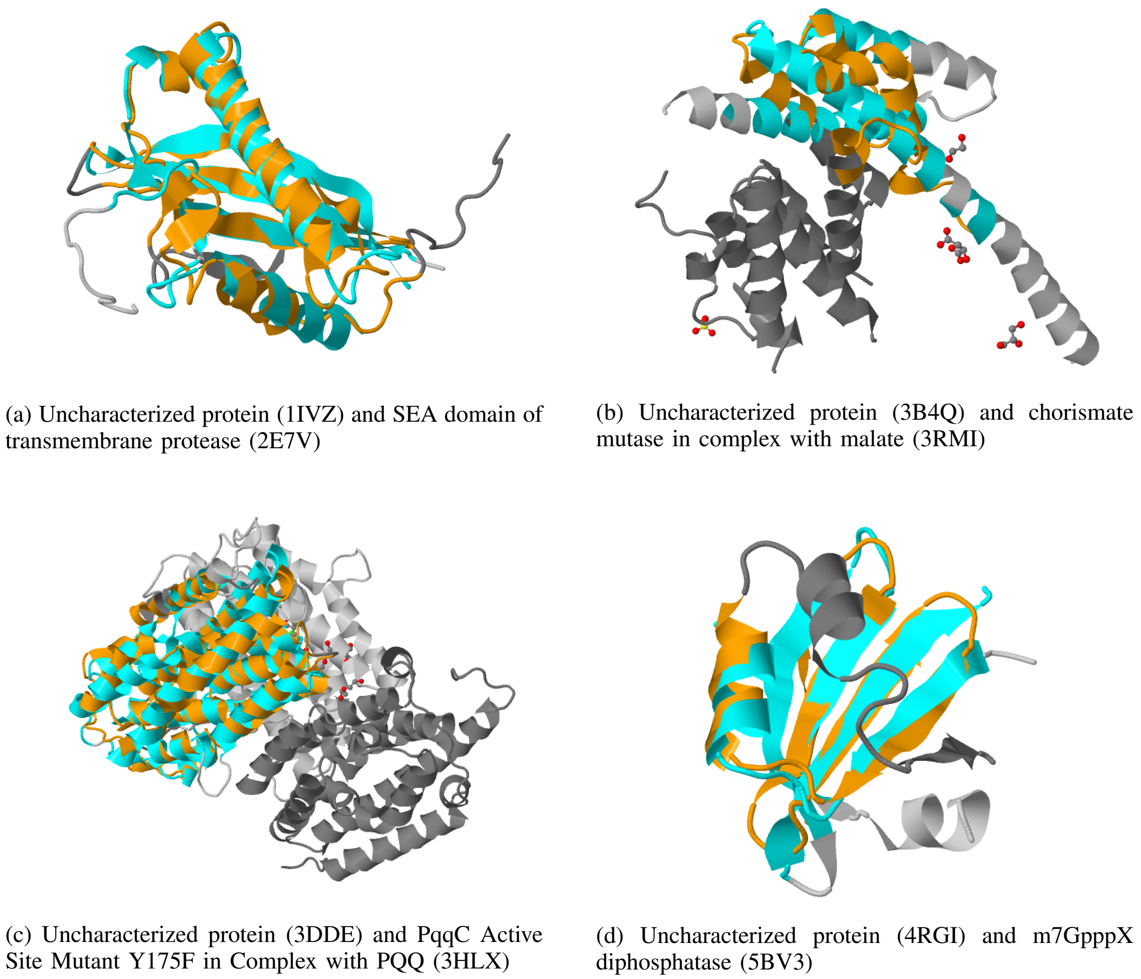
All of our software is available in GitHub, under the GNU General Public License, and our tools can be integrated into existing frameworks and the code and methodology can be incorporated into other software.

Esperite is up and running at http://esperite.csail.mit.edu.

APPENDIX A: THEORETICAL FOUNDATIONS

Evaluation of time complexity.

To facilitate and streamline the analysis of time complexity, we consider the high-dimensional vector space as a set



(a) Uncharacterized protein (1IVZ) and SEA domain of transmembrane protease (2E7V)

(b) Uncharacterized protein (3B4Q) and chorismate mutase in complex with malate (3RMI)

(c) Uncharacterized protein (3DDE) and PqqC Active Site Mutant Y175F in Complex with PQQ (3HLX)

(d) Uncharacterized protein (4RGI) and m7GpppX diphosphatase (5BV3)

Fig. 13: Four uncharacterized proteins (1IVZ, 3B4Q, 3DDE, and 4RGI) and their closest structural neighbors (2E7V, 3RMI, 3HLX, and 5BV3, respectively), printed as output by Esperite (Cosine distance between any one of the pairs were 0.0), and to be investigated further through protein nuclear magnetic resonance spectroscopy.

S of the collection of n unique points:

$$D_S(q,r) = p \in S : ||q-p|| < r \tag{8}$$

The problem at hand then becomes to compute $D_S(q,r)$ for a query protein q and a given non-zero radius r. It is important to note that a distance metric, particularly, is needed to only avoid a non-zero loss in sensitivity.

Initially, a random cluster center k is defined in this high-dimensional vector-space S. The points around this cluster center $k$within a user-defined radius r_c are then clustered and assigned to the cluster center k. Iteratively, a set C of k cluster centers are defined so that there are no clusters with a cluster radius greater than a user-defined radius r_c. We assign and note each cluster to its center, so the set C is also the exhaustive set of clusters in the high-dimensional vector space S. For a given task of similarity measurement between a query protein q and all data points within distance d, the approach consists of n iterative steps, combined with two real-time variables r_c and r that change values according to another user-specified parameter p_r, where

$$r = p_r r_c \tag{9}$$

For ease of analysis, we will work on a hierarchical framework where $n = 2$, i.e., only two layers of clustering is done in the high-dimensional vector space S (note that a deeper clustering will result in a depth of kn, where k is a non-zero integer, thus scaling linearly with the running time of the search). The overall search is then split into 2 stages of searches, with radii of $r+r_c$ and r, respectively. The similarity task of the first stage of the search, or the *coarse search*, is then $D_C(q,r+r_c)$, and the union of all defined clusters C_k with center k for the radius $r+r_c$ is then

$$U_C = \bigcup_{C_k \in D_C(q,r+r_c)} C_k \tag{10}$$

Triangle Inequality, which all distance metrics obey, implies that the second stage of the search, or the *fine search*, $D_F(q,r) \subseteq U_C$. Intuitively, as $U_C \subseteq S$,

$$D_F(q,r) = D_C(q,r) \tag{11}$$

So the same defined clusters C_k with centers k and radius $r+r_c$ in the coarse search can be used in fine search with radius r.

In an entropy-based construction of the dataset, the distance function used is only a metric to avoid disobeying the Triangle Inequality. It has been noted by Yu et al. (2015) that the use of many distance functions are plausible, yet decrease the amount of sensitivity the approach exhibits. More particularly, if a triplet of data points T_α over all triplets T_S in the high-dimensional vector space S do not satisfy the Inequality, the sensitivity is then $1-\frac{T_\alpha}{T_S}$ [28]. As experimentally shown in the results, this loss is infinitesimal, and inclined to lose significance as $r+r_c \to \infty$.

If the fractal dimension of the high-dimensional vector space S is low, the n-stage hierarchical clustering approach has a running time increase that is linearly proportional with the metric entropy of the database. We note that this database, along with any newly-added points to the database, can also be stored in a space complexity linearly proportional to the metric entropy of the database.

Bounds of complexity.

Definition 1. Let S be a high-dimensional vector space, D a subset of S, and r a non-zero radius. The *metric entropy* $E_r(S)$ of the high-dimensional vector space S is then the minimum number of points $p_1, p_2, ..., p_n$ so that spherical clusters $S(p_1,r), S(p_2,r)..., S(p_n,r)$ include all points in S [44].

Definition 2. Given any high-dimensional vector space S, the Hausdorff-Besicovitch dimension is

$$dim_H = \lim_{r\to 0} \frac{\log E_r(S)}{\log 1/r} \tag{12}$$

Intuitively, we note that for all cases, $dim_H = 0$ since S is finite and countable. We then utilize a more narrow definition of fractal dimension located around a distance scale with large radii:

Definition 3. Given any high-dimensional vector space S, and a radius scale $[r_1, r_2]$, the *fractal dimension* is

$$dim_f = \lim_{r\to r_2} \frac{\log \frac{E_r(S)}{E_{r_1}(S)}}{\log \frac{r_1}{r}} \tag{13}$$

Intuitively, we note that after the initial clustering with radii r_c and the construction of a set C of k cluster centers in the high-dimensional vector space S, $k \geq E_{r_c}(S)$, since the criterion that no cluster has a radius greater than r_c is satisfied. Thus, set of cluster centers k is bounded by metric entropy $E_{r_c}(S)$ of the high-dimensional vector space S.

Still following the specific case of n-stage hierarchical clustering approach where $n=2$, we note that:

- For a given query q, the coarse search will be exhaustively completed only after k comparisons.
- The fine search is bounded in the union U_C in (5), which is the union of clusters $r+r_c$ away from query q. Thus, the running time for the $n=2$ search overall is $\mathcal{O}(k+|U_C|)$, as the sum of running times for both the coarse and the fine search.
- $U_C \subset D_C(q, r+2r_c)$ by the triangle inequality (proof is trivial); thus, U_C is bounded by $D_C(q, r+2r_c)$.
- Recall Definition 3. Given the non-rigorous definition of fractal dimension, we note that for a scale of radii $[r_\alpha, r_\beta]$, and an increase from r_α to r_β, any newly-discovered point will be covered by fractal dimension d in a running time of $\mathcal{O}(\frac{r_\beta}{r_\alpha})^d$.

Thus, the overall time complexity for $n=2$ is

$$T(n) = \mathcal{O}(k + D_C(q,r)(\frac{r+2r_c}{r})^d) \tag{14}$$

Note that for asymptotically small values of fractal dimension and a linear output size, the running time converges to

$$T(n) = \mathcal{O}(n\, log\, n) \tag{15}$$

as k is linearly scaled with entropy, thus logarithmically with the dataset; and the output size n is linear.

APPENDIX B: MOTIVATION FOR DIVISIVE HIERARCHICAL CLUSTERING

Two distinct methods can be used for hierarchical clustering: agglomerative, and divisive. Agglomerative clustering is when each data point is initially accepted as a cluster within itself, and these clusters are merged using the similarity quantification via distance metrics. Clustering continues until the high-dimensional vector space $S=1$.

Divisive clustering can be thought of as the inverse: The high-dimensional vector space S is initialized as the first cluster, and closer data points in the cluster create a new sub-cluster. Divisive hierarchical clustering continues until every single data point in S is a cluster within itself.

Because the algorithm starts with the high dimensional vector space S being the initial cluster, and ends when all data points are clusters within themselves or user-defined d equals timer t, the system is based on flat divisive hierarchical clustering. Although these two methods may seem like they are supposed to replicate results when timer t is reversed, when the distance metric and the similarity criterion are introduced, using divisive hierarchical clustering is proven to be more optimal:

- Assume you have a cluster center k and a radius r when $t=0$ and $R(r)=2r$. Every data point n where $D_{nk} = r-s$ where $s \to 0$ is included in cluster C_k.
- If this cluster center k is $2r-s$ further away from another cluster center l, then when $t=1$, C_k and C_l should be merged.
- At this point, we want the data points which are members of C_l to be less than or equal to $2r$.

Yet when we merge the two clusters, n is almost $3r$ away from l but is still a member of C_l.

Agglomerative clustering is flawed in similarity search; it is not error-free for radius r, thus the cluster centers are not entirely reliable as they include more distant points in each step. The advantage of starting clustering from a high-dimensional vector space S is that since each level of clustering happens only within the clusters from the previous level, clusters that have elements that are too far from one another are not merged. It is also a more succinct implementation on top of the existing initial clustering.

ACKNOWLEDGMENTS

The author thanks Dr. Bonnie Berger and Dr. Noah Daniels at Computer Science and Artificial Intelligence Laboratory (CSAIL) at the Massachusetts Institute of Technology (MIT) for their mentorship and guidance, Dr. Semsettin Turkoz, Mr. and Mrs. Mustafa Basar Arioglu, and Mr. Kemal Tunc for their sponsorship, Joseph Dexter and Marie Herring for their suggestions and support, Andrew Jin and Daniel Yang for their feedback, Andrew Gallant and Yun William Yu for FragBag implementation, and the related groups for making the following packages available: the Jmol/JSMol package (http://jmol.sourceforge.net), Go Programming Language (http://golang.org), and the PDB (http://rcsb.org/pdb).

REFERENCES

[1] A. Pastore and A. M. Lesk, Comparison of the structures of globins and phycocyanins: Evidence for evolutionary relationship, *Proteins: Structure, Function, and Bioinformatics*, 8:133-155, 1990.

[2] W. A. Koppensteiner, P. Lackner, M. Wiederstein, and M. J. Sippl, Characterization of novel proteins based on known protein structures, *Journal of Molecular Biology*, 296:1139-1152, 2000.

[3] C. Chothia and A. M. Lesk, The relation between the divergence of sequence and structure in proteins, *The EMBO Journal*, 5:823-826, 1986.

[4] R. Eisner, B. Poulin, D. Szafron, P. Lu, and R. Grenier, Improving protein function prediction using hierarchical structure of the gene ontology, *Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2005.

[5] N. S. Boutonnet, M. J. Rooman, M. E. Ochagavia, J. Richelle, and S. J. Wodak, Optimal protein structure alignments by multiple linkage clustering: Application to distantly related proteins, *Protein Engineering, Design & Selection*, 8:647-662, 1995.

[6] C. P. Ponting, Issues in predicting protein function from sequence, *Briefings in Bioinformatics*, 2:19-29, 2001.

[7] A. Godzik, The structural alignment between two proteins: Is there a unique answer?, *Protein Science*, 5:1325-1338, 1996.

[8] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Advances in knowledge discovery and data mining, *AAAI/MIT Press*, 1996.

[9] A. Gersho and R. M. Gray, Vector quantization and signal compression, *Boston: Kluwer Academic*, 1992.

[10] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, An efficient k-means clustering algorithm: analysis and implementation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:7, 2002.

[11] A. Brahme, Comprehensive biomedical physics, *Newnes*, 1:136, 2014.

[12] B. Ekim, A machine learning approach to cross-generational single-point mutation patterns across the E.coli genome, unpublished.

[13] M. Sunouchi, Y. Tanaka, Similarity search of freesound environmental sound based on their enhanced multiscale fractal dimension, *Proceedings of the Sound and Music Computing Conference*, 2013.

[14] M. J. Sippl and M. Wiederstein, A note on difficult structure alignment problems, *Bioinformatics*, 24:426-427, 2008.

[15] L. Holm and C. Sander, Protein structure comparison by alignment of distance matrices, *Journal of Molecular Biology*, 233:123-138, 1993.

[16] Y. Ye and A. Godzik, FATCAT: A web server for flexible structure comparison and structure similarity searching. *Nucleic Acids Research*, 32:582-585, 2004.

[17] M. Menke, B. Berger, and L. Cowen, MATT: Local

flexibility aids protein multiple structure alignment. *PLoS Computational Biology*, 4:10, 2008.

[18] R. H. Lathrop, The protein threading problem with sequence amino acid interaction preferences is NP-complete, *Protein Engineering, Design & Selection*, 7:1059-1068, (1994).

[19] G. Mayr, F. S. Domingues, and P. Lackner, Comparative analysis of protein structure alignments, *BMC Structural Biology*, 7:50, 2007.

[20] R. Mosca, B. Brannetti, and T. R. Schneider, Alignment of protein structures in the presence of domain motions, *BMC Bioinformatics*, 9:352, 2008.

[21] Z. Lu, Z. Zhao, and B. Fu, Efficient protein alignment algorithm for protein search, *BMC Bioinformatics*, 11:34, 2010.

[22] T. Kato, K. Tsuda, K. Tomii, and K. Asai, A new variational framework for rigid-body alignment, *Lecture Notes in Computer Science*, 3138:171-179, 2004.

[23] E. Roberts, J. Eargle, D. Wright, and Z. Luthey-Schulten, MultiSeq: unifying sequence and structure data for evolutionary analysis, *BMC Bioinformatics*, 7:382, 2006.

[24] A. S. Konagurthu, J. C. Whisstock, P. J. Stuckey, and A. M. Lesk, MUSTANG: a multiple structural alignment algorithm, *Proteins: Structure, Function, and Bioinformatics*, 64:559-574, 2006.

[25] L. Holm and J. Park, DaliLite workbench for protein structure comparison, *Bioinformatics*, 16:566-567, 2000.

[26] M. Kosloff and R. Kolodny. Sequence-similar, structure-dissimilar protein pairs in the PDB, *Proteins: Structure, Function, and Bioinformatics*, 71:891-902, 2008.

[27] I. Budowski-Tal, Y. Nov, and R. Kolodny, FragBag, an accurate representation of protein structure, retrieves structural neighbors from the entire PDB quickly and accurately, *Proceedings of the National Academy of Sciences of the United States of America*, 107:3481-3486, 2009.

[28] Y. W. Yu, N. M. Daniels, D. C. Danko, and B. Berger, Entropy-scaling search of massive biological data, *Cell Systems*, 1:130-140, 2015.

[29] Z. Aung and K. L. Tan, Rapid retrieval of protein structures from databases, *Drug Discovery Today*, 12:732-739, 2007.

[30] I. G. Choi, J. Kwon, and S. H. Kim, Local feature frequency profile: A method to measure structural similarity in proteins, *Proceedings of the National Academy of Sciences of the United States of America*, 11:3797-3802, 2004.

[31] P. Rgen and B. Fain, Automatic classification of protein structure by using Gauss integrals, *Proceedings of the National Academy of Sciences of the United States of America*, 1:119-124, 2003.

[32] I. Friedberg, T. Harder, R. Kolodny, E. Sitbon, Z. Li, and A. Godzik, Using an alignment of fragment strings for comparing protein structures, *Bioinformatics*, 23:219-224, 2007.

[33] C.H. Tung, J. W. Huang, and J. M. Yang, Kappa-alpha plot derived structural alphabet and BLOSUM-like substitution matrix for rapid search of protein structure database, *Genome Biology*, 8:31, 2007.

[34] Z. Gaspari, K. Vlahovicek, and S. Pongor, Efficient recognition of folds in protein 3D structures by the improved PRIDE algorithm, *Bioinformatics*, 21:3322-3323, 2005.

[35] M. Gerstein and M. Levitt, Comprehensive assessment of automatic structural alignment against a manual standard, the SCOP classification of proteins, *Protein Science*, 7:445-456, 1998.

[36] F. Krull, G. Korff, N. Elghobashi-Meinhardt, and E. W. Knapp, ProPairs: A data set for protein-protein docking, *Journal of Chemical Information and Modeling*, 55:1495-507, 2015.

[37] K. P. Smith, K. M. Gifford, and J. S. Waitzman, Survey of phosphorylation near drug binding sites in the Protein Data Bank (PDB) and their effects, *Proteins: Structure, Function, and Bioinformatics*, 83:25-36, 2015.

[38] E. Cukuroglu, A. Gursoy, R. Nussinov, and O. Keskin, Non-redundant unique interface structures as templates for modeling protein interactions, *PLoS One*, 9:86738, 2014.

[39] D. Mary-Rajathei and S. Selvaraj, Analysis of sequence repeats of proteins in the PDB, *Computational Biology and Chemistry*, 47:156-166, 2013.

[40] S. C. Bull, M. R. Muldoon, and A. J. Doig, Maximising the size of non-redundant protein datasets using graph theory, PLoS One. *PLoS One*, 8:55484, 2013.

[41] Y. Zhang and J. Skolnick, TM-align: A protein structure alignment algorithm based on TM-score, *Nucleic Acids Research*, 33:2302-2309, 2005.

[42] C. H. Tung, J. W. Huang, and J. M. Yang, Kappa-alpha plot derived structural alphabet and BLOSUM-like substitution matrix for fast protein structure database search, *Genome Biology*, 8:31, 2007.

[43] R. Corral-Corral, E. Chavez, G. Del Rio, Machine learnable fold space representation based on residue cluster classes., *Computational Biology and Chemistry*, 59:1-7, 2015.

[44] T. Tao, Product set estimates for non-commutative groups, *Combinatorica*, 2006.