RUPEE: Scalable Protein Structure Search using Run Position Encoded Residue Descriptors

Ron Ayoub
School of Computing and Engineering
University of Missouri-Kansas City
Kansas City, Missouri, 64110
Email:ronaldayoub@mail.umkc.edu

Yugyung Lee
School of Computing and Engineering
University of Missouri-Kansas City
Kansas City, Missouri, 64110
Email:leeyu@umkc.edu

***Abstract*—We have developed a fast, scalable, and sequence order-flexible protein structure search combining techniques from information retrieval and big data with a novel approach to encoding sequences of torsion angles. Along the way, we introduce a new torsion angle plot without breaks in continuity while still maintaining traditional torsion angle ranges, to assist in defining separable regions of torsion angles. Subsequently, we introduce a new heuristic we call *run position encoding*, for handling the lack of specificity of items within character sequences containing runs of repeats. Comparing our results to the output of the CATH structural scan, response times are measured in seconds as opposed to minutes and average RMSDs and TM-scores are better. Our approach is a step towards a comprehensive indexing of protein structures scalable to millions of entries. Code and data are available at https://github.com/rayoub/rupee**

I. Introduction

Proteins represent the functional end-product within the central dogma of molecular biology [1]. As such, understanding protein structure is a central goal within structural bioinformatics. Protein structure determination, prediction, alignment, and search all serve to advance this understanding. Below, we present our new approach to a fast, scalable, and sequence order-flexible protein structure search we refer to with the cumbersome but catchy acronym of *RUn Position Encoded Encodings* of residue descriptors (RUPEE).

For comparisons, we examine results from the CATHEDRAL structural scan [2] available at the CATH website [3]. In addition to recognizing domains in multidomain proteins, CATHEDRAL also can be used to find structural neighbors among CATH domains for an uploaded protein data bank (PDB) file, that is, where an identifier is not provided that can be used for retrieval of pre-calculated results. CATHEDRAL is effective at this task but can take upwards of 10 minutes to produce results against CATH s35 representatives.

Restricting our initial approach to protein domains, we have indexed CATH v4.1 domains, consisting of 308,999 structures, and developed a feature similar to CATHEDRAL that performs a protein structure search based on a PDB file, returning results in seconds rather than minutes. To evaluate RUPEE, we have made comparisons to the output of CATHDERAL along two dimensions, quality of results and response times. In both cases, RUPEE performs better on average for the domains we evaluated.

Besides our approach to protein structure search, we introduce a polar plot for torsion angles that may have wider applicability in the study of protein structure. Further, the *run position encoding* heuristic introduced below may have wider applicability to algorithms for character sequences containing long runs of repeats.

For the remainder of this paper, we first discuss some related work to provide context for our approach followed by a description of our method. We end with the results of evaluation with CATHEDRAL along with an analysis and conclusion.

II. Related Work

Pairwise alignment involves finding a set of spatial rotations and translations for two protein structures that minimizes a distance metric. Most commonly, the root mean squared deviation (RMSD) between α-carbons of aligned residues is minimized.

The typical use case of aligning one protein structure to another does not impose tight response time requirements. For this reason, pairwise alignments can focus on accuracy. On the other hand, a protein structure search can involve thousands of comparisons and accuracy is often balanced against speed. In this case, pairwise alignment is still useful for evaluating the results of the search.

For pairwise alignment, Combinatorial Extensions (CE) [4] and FATCAT [5] are among the most popular tools, representing rigid and flexible protein alignments, respectively. CE performs a rigid alignment in order to minimize RMSD and FATCAT allows for a constrained number of twists in the protein chain in order to find a more flexible alignment before minimizing RMSD.

Though CE and FATCAT standout as complimentary methods, rigid and flexible respectively, they are both limited by being sequence order-dependent alignments as touched upon in [6]. While they are not dependent on the specific sequence of amino acids along the backbone, they are dependent on the ordering of the α-carbons. Even this lesser form of dependence can fail to account for divergence between homologous proteins [7].

To address this, topological permutations such as circular permutations, segment-swapping and changing secondary

structures within homologous proteins have been examined in [8]. Subsequently, Combinatorial Extension for Circular Permutations (CE-CP) [9], an extension to CE, was developed to address circular permutations specifically.

Whereas pairwise alignments often only depend on the sequence order of α-carbons, protein structure searches often introduce a further dependence on the specific sequence of amino acids. This approach often takes the form of clustering proteins based on their amino acid sequences and pre-calculating results for pairwise alignments among cluster representatives. Then, these pre-calculated results are used for filtering the number of structures used for comparisons against a query protein. The exact formula for combining the use of representatives and pre-calculated results varies from system to system. However, all systems using this approach share the same disadvantage, an indirect dependence on amino acid sequences through the choice of representatives based on clustering sequences. In the absence of a reliance on representatives and pre-calculated results, and without sacrificing accuracy, response times suffer greatly, often taking upwards of an hour for queries to complete.

For protein structure searches, VAST [10] and the FATCAT server [11] are among the most popular. If given a known protein domain, VAST can return structural neighbors in seconds based on pre-calculated results. However, if uploading a PDB file where pre-calculated results are not used, response times for VAST can exceed 30 minutes. Similarly, the FATCAT server, that does not use pre-calculated results, can take over an hour to send results for a search against PDB-90 representatives [6].

The PhyreStorm server [12] provides another notable protein structure search. Here too, it depends on PDB pre-calculated pairwise comparisons among PDB-40 cluster representatives [6]. Notably, PhyreStorm returns good results within minutes based on an uploaded PDB file.

Given the above, there remains a need for a sequence order-independent protein structure search. For the serendipitous exploration of relations between protein structures performed in the trenches, this search should be fast. Moreover, with a 10% yearly growth rate of solved structures deposited in the PDB [13], this search should be scalable. At a minimum, RUPEE takes a significant step in this direction as will be shown below.

III. Methods

Broadly, we define a linear encoding of protein structure and convert this linear encoding into a bag of features amenable to big data processing. Protein structure searches that use linear encodings are not unique [14]–[16]. The novelty of our approach lies in its remarkable performance given its simplicity. Additionally, elements of our approach can be isolated and found to be useful in their own right.

A. Regions of Torsion Angles

Our first step towards a linear encoding of protein structure is to encode separable regions of permissible torsion angles, but first we introduce a new plot of torsion angles better suited to this effort.

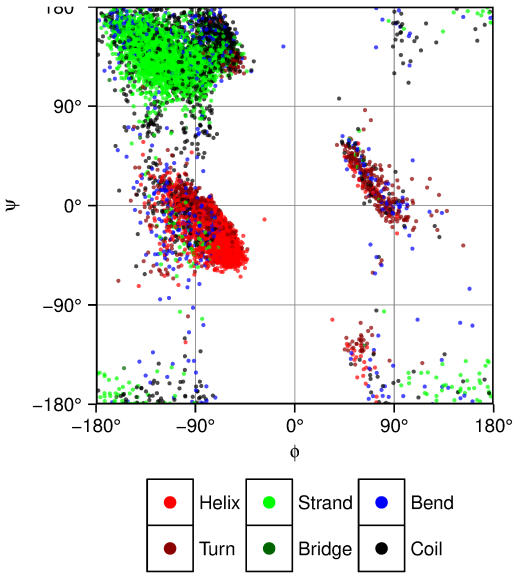

Fig. 1. Ramachandran plot of randomly sampled torsion angles

Torsion angles, first developed in [17] by Sasisekharan in the research group of Ramachandran, have seen near universal acceptance as the primary determinate of protein structure. Likewise, the ubiquitous Ramachandran plot [18] is the most widely used visualization of permissible torsion angle regions.

Despite the utility and familiarity of Ramachandran plots, they represent angular data using a square plot better suited for scalar data. This leads to the unwieldy arrangement where the top part of the plot is continuous with the bottom and the left is continuous with the right.

To identify regions of torsion angles, we randomly sampled 10,000 residues from high-resolution, CATH s35 representatives to account for precision and redundancy, respectively. A Ramachandran plot of the sampled torsions angles is shown in Fig. 1. As can be seen, a single cluster of β-strand residues is represented, at least to some extent, at all 4 corners of the Ramachandran plot.

This continuity problem was partially addressed in [19] using *wrapped* and *mirrored* plots. Both wrapped and mirrored plots take advantage of the sparsely populated vertical strip of the Ramachandran plot at $\phi = 0°$ and the horizontal strip at $\psi = -120°$. However, with larger samples of torsion angles, the vertical strip remains sparse whereas the horizontal strip becomes less sparse. We found the use of a polar plot resolves this elegantly by only requiring one break in continuity along the $\phi = 0°$ vertical.

Fig. 2 plots the same torsion angles appearing in Fig. 1 using the polar plot. In this plot, ϕ corresponds to the radius r and ψ corresponds to the angle θ in traditional polar plots. Notice the β-strand residues appearing at the 4 corners of the Ramachandran plot now appear in one continuous region of the polar plot centered at $\phi = \pm 180°$ and $\psi = \pm 180°$.

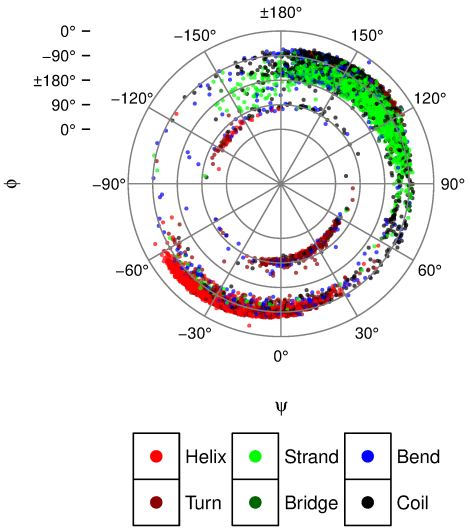

Fig. 2. Polar plot of randomly sampled torsion angles

TABLE I
DSSP CODE/DESCRIPTOR MAP

DSSP Codes	SS	Descriptors
G, H, I	helix	1-4
E	strand	5-7
S, C	bend, coil	8-10
T	turn	11
B	bridge	12

B. Linear Encoding of Protein Structure

The polar plot described above is used to define torsion angle regions for each secondary structure assignment. The eight DSSP secondary structure assignment codes defined in [20] divide into three groups in which torsion angle regions are roughly the same: 'G','H','I', and 'T' corresponding to 3_{10}-helix, α-helix, π-helix, and turn, respectively; 'E' and 'B' corresponding to β-strand and β-bridge, respectively; and 'S' and 'C' corresponding to bend and coil, respectively.

Polar plots for each group of DSSP assignment codes are shown in Fig. 3 with the exception of turns and bridges, which each receive a single descriptor designation independent of torsion angles. For each polar plot, there are well-defined continuous regions of torsion angles that remain continuous in the plots. The only exception is found in the bends and coil plot at $\psi = 60°$ between $\phi = -180°$ and $\phi = 0°$.

TABLE I summarizes the mapping between region and secondary structure assignment combinations and ranges of integer valued descriptors. The specific mapping was arrived at primarily based on quality of results.

Our initial use of Ramachandran plots to identify regions resulted in the frequent case of highly similar domains from the same CATH s95 clusters being considered significantly different. Alignment of these domains in PyMOL [21] and the development of a script to label residues with their descriptors

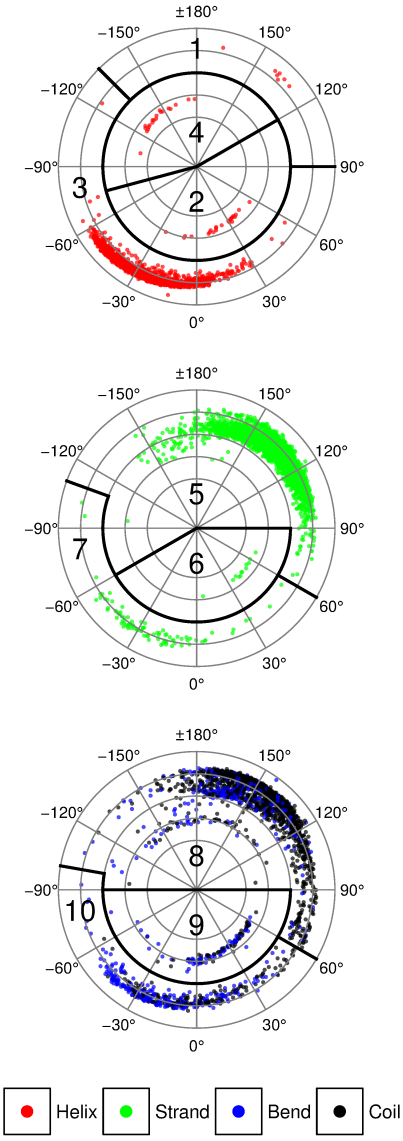

Fig. 3. Polar plots of randomly sampled torsion angles with designated descriptors for region and DSSP code combinations are shown. Descriptors for turns and bridges are not shown since those are not dependent on region.

showed that slight differences in structure often resulted in distinct descriptors being assigned to aligned residues. Once we defined regions using the less confusing polar plots, this discrepancy was no longer observed.

As an example of our linear encoding, the following sequence of descriptors represents a β-turn-β motif followed by bends and coil.

$$[\,5,5,5,5,5,11,11,5,5,5,5,5,8,8,10,9\,] \tag{1}$$

As noted in [22], secondary structure assignments for termini are not always consistent among the variety of secondary structure assignment tools. We also observed this to be the case for DSSP assignments across homologous domains. Here, we introduce a heuristic we call *secondary structure extension* (SSE) to address this variance.

SSE iterates a sequence of descriptors once in both the forward and backward directions. Whenever a helix or strand descriptor is encountered, the next descriptor receives the same assignment as the preceding residue if its torsion angles match the region of the preceding residue, regardless of its own secondary structure assignment.

SSE performed on the sequence found in (1) gives the sequence in (2). Notice, the subsequence $[\,8, 8\,]$ in (1), corresponding to bends with torsion angles falling in the densest region of β-strands, becomes $[\,5, 5\,]$ in (2).

$$[\,5, 5, 5, 5, 5, 11, 11, 5, 5, 5, 5, 5, 5, 5, 10, 9\,] \tag{2}$$

Though [22] lends some support to this heuristic, we use it primarily based on its performance.

C. Bag representation of protein structure

Once a linear encoding for a protein structure is obtained, it needs to be further transformed into a representation suitable for fast and scalable similarity comparisons to other structures. The processing of text documents within Information Retrieval (IR) has long been used to satisfy these requirements using bag representations. There are two distinct categories of representations for documents, syntactic and semantic, and much of the research applying IR to protein structure search has focused on the latter [23]–[25].

We have adapted the syntactic approach to documents, often referred to as shingling [26], to our linear encoding of protein structure. We transform a linear sequence of descriptors into a multiset of shingles consisting of 4 consecutive descriptors. The overlap between shingles ensures some of the order information within the original sequence is preserved in the bag while maintaining sequence order-flexibility.

By shingling, we obtain a multiset of ordered lists from an ordered list of numbers. As an example, the sequence in (2) is transformed into the following bag of shingles.

$$\begin{aligned}\{\,&[5,5,5,5],[5,5,5,5],[5,5,5,11],[5,5,11,11],\\ &[5,11,11,5],[11,11,5,5],[11,5,5,5],[5,5,5,5],\\ &[5,5,5,5],[5,5,5,5],[5,5,5,5],[5,5,5,10],\\ &[5,5,10,9]\,\}\end{aligned} \tag{3}$$

Next, each shingle s is hashed to an integer as shown in (4). The hash function used is a simplification of the hash function used in the Rabin-Karp algorithm [27]. The prime number 13 is used as the base since it is large enough to spread the descriptor values out in hash space without collisions.

$$s_{hash} = s_1 \times 13^3 + s_2 \times 13^2 + s_3 \times 13 + s_4 \tag{4}$$

Subsequently, the multiset in (3) becomes the following bag of integers.

$$\begin{aligned}\{\,&11900, 11900, 11906, 11984, 12992,\\ &26096, 25082, 11900, 11900, 11900,\\ &11900, 11905, 11969\,\}\end{aligned} \tag{5}$$

This final step completes the transformation of an ordered list of descriptors to a multiset of integers that still retains some of the order information present in the original list.

Notice in (5) the value 11900, corresponding to the shingle $[5, 5, 5, 5]$, occurs frequently indicating the presence of a β-strand. Since most proteins are dominated by regular secondary structure, the abundance of shingles for β-strands as well as the three types of helices, end up dominating comparisons. Moreover, since shingles are limited in length, this situation allows for structures with many short β-strands to match structures with fewer long β-strands. The same situation applies to helices.

To address this lack of specificity, we introduce a heuristic we call *run position encoding* (RPE). To distinguish between short and long runs, thereby increasing the specificity of the shingles, we add a factor of 10^5 to each shingle hash as a function of the first residue's position in a run i.

$$runfactor(i) = \begin{cases} i & \text{if } i < \lfloor l/2 \rfloor \\ l - i - 1 & \text{otherwise} \end{cases} \tag{6}$$

where i is zero-based and l is the length of the run. Multiplying by 10^5 places the run factor as the left-most digit in the hash to avoid interference with the digits provided by the hash in (4). This placement is also convenient for visual inspection, since the run factor is isolated as the left-most digit.

The run factors for the sequence in (2) are

$$[\,0, 1, 2, 1, 0, 0, 0, 0, 1, 2, 3, 2, 1, 0, 0, 0\,]. \tag{7}$$

Applied to the bag of integers in (5) gives

$$\begin{aligned}\{\,&011900, 111900, 211906, 111984, 012992,\\ &026096, 025082, 011900, 111900, 211900,\\ &311900, 211905, 111969\,\},\end{aligned} \tag{8}$$

where the leading zero run factors are shown for clarity.

This pyramidal approach preserves matches at the boundaries between secondary structure runs and loops that would not otherwise be preserved in the presence of differences in run lengths of one or more.

To see why RPE run factors are calculated at the descriptor level and factored in at the shingle level, consider shingling a list of RPE run factors themselves, which mirrors applying them at the descriptor level.

The sequence of RPE factors

$[\,0, 1, 2, 3, 2, 1, 0\,]$ becomes

$\{\,[0, 1, 2, 3], [1, 2, 3, 2], [2, 3, 2, 1], [3, 2, 1, 0]\,\}$

and with one less element

$[\,0, 1, 2, 2, 1, 0\,]$ becomes

$\{\,[0, 1, 2, 2], [1, 2, 2, 1], [2, 2, 1, 0]\,\}$

Notice above, there is not a single shingle match for this one-off difference in run length. Now consider shingling a list of RPE factors, but this time all elements in the shingle are

set equal to the first element in the shingle, which mirrors applying them at the shingle level.

The sequence of RPE factors

$[\,0,1,2,3,2,1,0\,]$ becomes

$\{\,[0,0,0,0],[1,1,1,1],[2,2,2,2],[3,3,3,3]\,\}$

and with one less element

$[\,0,1,2,2,1,0\,]$ becomes

$\{\,[0,0,0,0],[1,1,1,1],[2,2,2,2]\,\}$

In this latter case, a one-off difference in run length results in one less shingle match while still serving to increase the specificity of the shingles.

Now that we have a representation of a protein structure as a bag of integers, similarity between any two structures a and b is defined as the Jaccard similarity [28] for multisets,

$$J(a,b) = \frac{\sum_i min(a_i, b_i)}{\sum_i max(a_i, b_i)}, \tag{9}$$

where i ranges over all possible shingle hashes s_i and a_i and b_i give the counts of shingle hash s_i in structures a and b, respectively.

D. Fast and scalable structure search

In IR, the bag of shingles representation of documents is used in the near dupe clustering of documents [29]. One application of near dupe clustering is in the review stage of Electronic-Discovery [30], which is the most expensive stage in a discovery process. Often millions of documents must be examined by a staff of attorneys to make a reasonable effort at providing all documents relevant to the discovery request. Grouping documents into near dupe clusters and assigning all documents within a cluster to a single reviewer reduces duplication of effort.

In the case of near dupe clustering, each document must be compared to every other document in the collection, taking quadratic time. For this task, min-hashing [31] and locality sensitive hashing (LSH) [32] can be combined to reduce this to subquadratic time. Although we do not near dupe cluster domains, we can still leverage the techniques of min-hashing and LSH to speed up protein structure search by a large constant factor.

Min-hashing is used to randomly select items from a set of items by repeatedly randomly hashing the items, sorting the hashes into a list, and then selecting the minimum item in each permuted list. If the same random permutation of items is performed on each set of items in a collection, the key result is that the probability of matching min-hashes is equal to the Jaccard similarity [31]. In order to approximate the Jaccard similarity for a given pair of sets, a sufficient number of min-hashes must be obtained.

In our case, the items are bags of shingle hashes for protein structures from which we obtain 60 min-hashes as described in [33]. Given the key result above, the Jaccard similarity can now be approximated by the proportion of matching min-hashes.

Weighted min-hashing [34] with integral weights is an extension to min-hashing that can be applied to multisets where the frequency of a hash within a multiset is used as its weight.

Next, we use the LSH banding technique as described in [33]. The key result of the banding technique is that if *any* band positions are a match for a given pair of structures, the probability that a specific similarity threshold has been met can be calculated. We use 20 bands of 3-min-hashes where the probability of a Jaccard similarity of 60% or greater is approximately 99%. Banding allows the problem of finding similar items to be parallelized across bands since all that is needed for a match is a single band match.

Together, min-hashing and LSH allow for a fast and scalable protein structure search based on run position encoded shingles of residue descriptors.

IV. Results

Protein structure searches can be evaluated using pairwise alignment scores or by comparison of results against the hierarchy of a protein structure classification database. The RMSD of aligned residues is widely used in evaluations but is not perfectly suited to full-length comparisons between structures since distances between unaligned residues are not factored into the score. On the other hand, the TM-score [35] takes all residues into account. Among protein structure classification databases, SCOPe [36] and CATH [3] are the most popular.

We evaluate RUPEE against the results of the CATHEDRAL structural scan available at the CATH website using RMSD scores and TM-scores provided as outputs from CE-CP and FATCAT pairwise alignments. We also evaluate the precision of RUPEE results against the topology and superfamily levels of the CATH hierarchy.

Given that CATHEDRAL does not depend on pre-calculated results but yet yields decent response times, it is well-suited for a fair comparison against RUPEE. Aside from that, the CATH web site that host CATHEDRAL is among the most popular protein structure classification databases that is currently maintained and regularly updated with new additions to the PDB. Given the integration of CATHEDRAL into the CATH website and with the CATH v4.1 database, it is convenient to compare precision of results against different levels of the CATH hierarchy.

All evaluations are based on structure searches for 94 superfamily representatives of the 100 most diverse CATH superfamilies. We only used superfamily representatives for which CATHEDRAL returned greater than 50 results and also returned the query domain as the first ranked result. This restriction accounts for possible bugs in CATHEDRAL and helps ensure a fair comparison.

For CATHEDRAL, one CATH superfamily representative (1jfbA00 for 1.10.630.10) only returned 14 results. For 5 other CATH superfamily representatives (3sjmA00, 3sovA02, 3zm1A00, 4cayB00, 4q4w300 for 3.40.50.720, 1.10.10.60, 2.10.25.10, 3.90.190.10, 1.10.20.10, 2.60.120.20, respectively), CATHEDRAL failed to return the query domain

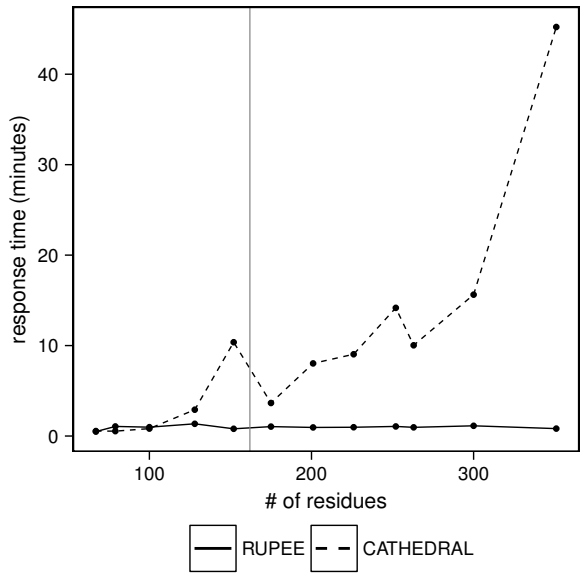

Fig. 4. Response times by residue count for RUPEE and CATHEDRAL

as the first ranked result. For all diverse superfamily representatives, RUPEE returned greater than 50 results and always returned the query domain as the first ranked result.

The search results among both tools are limited to CATH s35 representatives since this is what is returned by the CATHEDRAL structural scan. The plots in Fig. 5 and Fig. 6 are for the top 50 ranked results. For perspective, the minimum number of domains included in a diverse CATH superfamily is 573, the maximum is 19,468, and the average is 1,646. Among CATH s35 representatives, the minimum number of domains included in a diverse CATH superfamily is 1, the maximum is 707, and the average is 95.

A plot comparing response times by residue count among the two systems is shown in Fig. 4. CATHEDRAL returns results for searches against CATH s35 representatives in roughly between 30 seconds and 45 minutes depending on the residue count whereas RUPEE returns results against CATH s35 representatives in roughly 1 second or less, regardless of residue count, while running on an 8-core laptop. For a more balanced understanding of these numbers, the average number of residues in a CATH domain is 162, for which a vertical reference line is shown in Fig. 4. This corresponds to the casual observation that CATHEDRAL usually returns results in somewhere between 1 and 10 minutes.

Fig. 5 shows average cumulative values for each rank and proceeding ranks averaged over all searches. Both RMSD and TM-score values are shown, provided as outputs from CE-CP and FATCAT pairwise alignments. For RMSD, where lower scores are better, results are nearly identical, with RUPEE having a small advantage for ranks greater than 5 and CATHEDRAL having a smaller advantage for ranks less than 5. For TM-score, where higher scores are better, RUPEE has a clear advantage for ranks greater than 5 and CATHEDRAL has a small advantage for ranks less than 5.

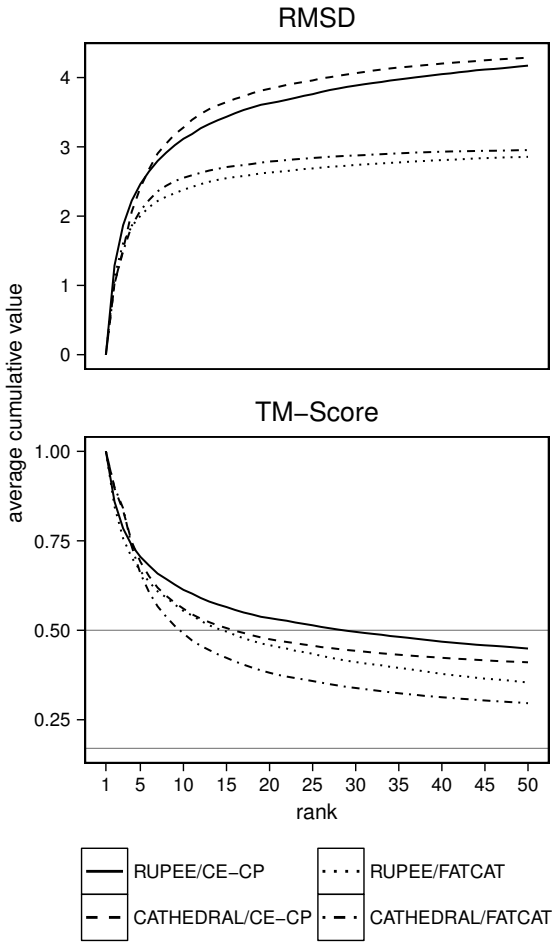

Fig. 5. Scoring for RUPEE and CATHEDRAL for ranked results among CATH s35 representatives

References lines are drawn in the TM-score plot in Fig. 5 at TM-scores of 0.5 and 0.17. A TM-score above 0.5 is a good predictor for whether or not two domains are in the same fold [37]. In CATH, the topology level of the hierarchy corresponds to fold. TM-scores greater than 0.17 are considered potentially meaningful whereas TM-scores less than 0.17 are considered to be due to random alignment [35]. Both RUPEE and CATHEDRAL TM-scores remain well above this level for ranks up to 50.

Fig. 6 shows precision (i.e. positive predictive value or PPV) averaged over all searches, where positive results are defined as domains in the same topology or the same superfamily as the query domain. As review, precision is defined as

$$\frac{TP}{TP + FP}$$

where TP is the number of true positives and FP is the number of false positives. The sum of true positives and false positives is the total number of predicted positives.

For ranks greater than 5, RUPEE has a clear advantage for topology precision and a large advantage for superfam-

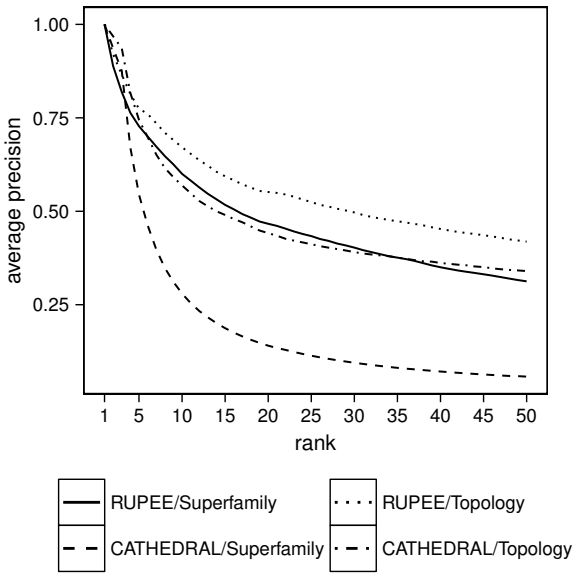

Fig. 6. Precision for RUPEE and CATHEDRAL for ranked results among CATH s35 representatives

ily precision. For both topology and superfamily precision, CATHEDRAL has a small advantage for ranks less than 5.

Overall, RUPEE has a clear advantage for ranks greater than 5 and a large advantage when it comes to performance, especially given that RUPEE was run on an 8-core laptop and CATHEDRAL results were obtained from the website where CATHEDRAL runs in a server environment.

V. Conclusion

With the growth rate of solved structures deposited in the PDB, the need for a fast and scalable structure search is growing. Using run position encoded shingles of residue descriptors combined with min-hashing and LSH, we have shown that RUPEE provides good results in seconds running on an 8-core laptop against the RUPEE index in a PostgresQL database, serviced by a single thread. RUPEE response times may be further improved by the structural near dupe clustering of domains for use in a filtering step not dependent on clusters derived from sequence data.

Aside from response times and scalability, there is room for improvement in RUPEE, especially with respect to encoding unstructured regions of proteins, where it is predicted that 30% of all proteins contain large, unstructured regions [38]. For short loops containing simple motifs such as the β-turn-β motif, shingle matches from these regions correctly factor into RUPEE similarity comparisons. However, with the longer and more complex loop and tail regions, random matching and mismatching takes a toll further down in the list of ranked results.

In addition to RUPEE, we have introduced two items that may find wider applicability. The first item is the introduction of a new polar torsion angle plot that maintains the continuity of permissible torsion angle regions while maintaining the familiar torsion angle ranges used in Ramachandran plots. The next item is the *run position encoding* heuristic, which may find wider applicability due to it simplicity and generality.

Acknowledgment

We would like to thank Prof. Deendayal Dinakarpandian for providing a tour of computational biology and expanding our universe of interesting computational problems to consider.

References

[1] F. Crick, "Central dogma of molecular biology," *Nature*, vol. 227, no. 5258, pp. 561–563, 1970.

[2] O. C. Redfern, A. Harrison, T. Dallman, F. M. Pearl, and C. A. Orengo, "CATHEDRAL: A fast and effective algorithm to predict folds and domain boundaries from multidomain protein structures," *PLoS Comput. Biol.*, vol. 3, no. 11, p. e232, 2007.

[3] C. Orengo, A. Mitchie, S. Jones, D. T. Jones, M. Swindells, and J. M. Thornton, "CATH - a hierarchic classification of protein domain structures," *Structure*, vol. 5, no. 8, pp. 1093–1109, 1997.

[4] I. N. Shindyalov and P. E. Bourne, "Protein structure alignment by incremental combinatorial extension (CE) of the optimal path," *Protein Eng. Des. Sel.*, vol. 11, no. 9, pp. 739–747, 1998.

[5] A. G. Yuzhen Ye, "Flexible structure alignment by chaining aligned fragment pairs allowing twists," *Bioinformatics*, vol. 19, no. 2, pp. 246–255, 2003.

[6] A. Prlić, S. Bliven, P. W. Rose, W. F. Bluhm, C. Bizon, A. Godzik, and P. E. Bourne, "Pre-calculated protein structure alignments at the RCSB PDB website," *Bioinformatics*, vol. 26, no. 23, pp. 2983–2985, 2010.

[7] A. G. Murzin, "How far divergent evolution goes in proteins," *Curr. Opin. Struct. Biol.*, vol. 8, no. 3, pp. 380–387, 1998.

[8] A. Andreeva, A. Prlić, T. J. Hubbard, and A. G. Murzin, "SISYPHUS: structural alignments for proteins with non-trivial relationships," *Nucleic Acids Res.*, vol. 35, no. 1, pp. 253–259, 2007.

[9] S. E. Bliven, P. E. Bourne, and A. Prlić, "Detection of circular permutations within protein structures using CE-CP," *Bioinformatics*, vol. 31, no. 8, pp. 1316–1318, 2015.

[10] J.-F. Gilbrat, T. Madej, and S. H. Bryant, "Surprising similarities in structure comparison," *Curr. Opin. Struct. Biol.*, vol. 6, no. 3, pp. 377–385, 1996.

[11] A. G. Yuzhen Ye, "FATCAT: a web server for flexible structure comparison and structure similarity searching," *Nucleic Acids Res.*, vol. 32, no. 2, pp. 582–585, 2004.

[12] S. Mezulis, M. J. Sternberg, and L. A. Kelley, "PhyreStorm: A web server for fast structural searches against the PDB," *J. Mol. Biol.*, vol. 428, no. 4, pp. 702–708, 2016.

[13] P. W. Rose, A. Prli, A. Altunkaya, C. Bi, A. R. Bradley, C. H. Christie, L. D. Costanzo, J. M. Duarte, S. Dutta, Z. Feng, R. K. Green, D. S. Goodsell, B. Hudson, T. Kalro, R. Lowe, E. Peisach, C. Randle, A. S. Rose, C. Shao, Y.-P. Tao, Y. Valasatava, M. Voigt, J. D. Westbrook, J. Woo, H. Yang, J. Y. Young, C. Zardecki, H. M. Berman, and S. K. Burley, "The RCSB protein data bank: integrative view of protein, gene and 3D structural information," *Nucleic Acids Res.*, vol. 45, no. D1, pp. D271–D281, 2017.

[14] M. Carpentier, S. Brouillet, and J. Pothier, "YAKUSA: A fast structural database scanning method," *Proteins*, vol. 61, no. 1, pp. 137–151, 2005.

[15] P. Daniluk and B. Lesyng, "A novel method to compare protein structures using local descriptors," *BMC Bioinformatics*, vol. 12, no. 1, p. 344, 2011.

[16] L. Mavridis and D. W. Ritchie, *3D-Blast: 3D protein structure alignment, comparison, and classification using spherical polar fourier correlations*. World Scientific, 2012, pp. 281–292.

[17] G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan, "Stereochemistry of polypeptide chain configurations," *J. Mol. Biol.*, vol. 7, pp. 95–99, 1963.

[18] G. N. Ramachandran and V. Sasisekharan, "Conformation of polypeptides and proteins," *Adv. Protein Chem.*, vol. 23, pp. 283–438, 1968.

[19] S. A. Hollingsworth and P. A. Karplus, "A fresh look at the Ramachandran plot and the occurrence of standard structures in proteins," *Biomol. Concepts*, vol. 1, pp. 271–283, 2010.

[20] W. Kabsch and C. Sander, "Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features," *Biopolymers*, vol. 22, no. 12, pp. 2577–2637, 1983.

[21] Schrödinger, LLC, "The PyMOL molecular graphics system, version 1.8," November 2015, The PyMOL Molecular Graphics System, Version 1.8, Schrödinger, LLC.

[22] J. Martin, G. Letellier, A. Marin, J.-F. Taly, A. G. de Brevern, and J.-F. Gibrat, "Protein secondary structure assignment revisited: a detailed analysis of different assignment methods," *BMC Structural Biology*, vol. 5, no. 1, p. 17, 2005.

[23] Z. Aungand and K.-L. Tan, "Rapid 3d protein structure database searching using information retrieval techniques," *Bioinformatics*, vol. 20, no. 7, pp. 1045–1052, 2004.

[24] L. Zhang, J. Bailey, A. S. Konagurthu, and K. Ramamohanarao, "A fast indexing approach for protein structure comparison," *BMC Bioinformatics*, vol. 11, no. Suppl 1, p. S46, 2010.

[25] 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*, vol. 107, no. 8, pp. 3481–3486, 2010.

[26] A. Z. Broder, "On the resemblance and containment of documents," in *Proc. Compression and Complexity of Sequences*, Positano, Italy, 1997, pp. 21–29.

[27] R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," *IBM Journal of Research and Development*, vol. 31, no. 2, pp. 249–260, 1987.

[28] M. LEVANDOWSKY and D. WINTER, "Distance between sets," *Nature*, vol. 234, no. 5323, pp. 34–35, 1971.

[29] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," *Computer Networks and ISDN Systems*, vol. 29, no. 8-13, pp. 1157–1166, 1997.

[30] S. Joshi, D. Contractor, K. Ng, P. M. Deshpande, and T. Hampp, "Auto-grouping emails for faster e-discovery," in *Proceedings of Very Large Databases Endowment 2011*, vol. 4, no. 12, 2011, pp. 1284–1294.

[31] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," in *ACM Symposium on Theory of Computing*, Dallas, USA, 1998, pp. 327–336.

[32] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *ACM Symposium on Theory of Computing*, Dallas, USA, 1998, pp. 604–613.

[33] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2012, ch. 3, pp. 53–70.

[34] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-hash and tf-idf weighting," in *Proceedings of BMVC (2008)*, USA, 2008, pp. 493–502.

[35] Y. Zhang and J. Skolnick, "Scoring function for automated assessment of protein structure template quality," *Proteins*, vol. 57, no. 4, pp. 702–710, 2004.

[36] N. K. Fox, S. E. Brenner, and J.-M. Chandonia, "SCOPe: Structural classification of proteins - extended, integrating SCOP and ASTRAL data and classification of new structures," *Nucleic Acids Res.*, vol. 42, no. 1, pp. 304–309, 2014.

[37] Y. Z. J Xu, "How significant is a protein structure similarity with TM-score = 0.5?" *Bioinformatics*, vol. 26, no. 7, pp. 889–895, 2010.

[38] E. Schad, P. Tompa, and H. Hegyi, "The relationship between proteome size, structural disorder and organism complexity," *Genome Biology*, vol. 12, no. 12, p. R120, 2011.

[39] A. Prlić, A. Yates, S. E. Bliven, P. W. Rose, J. Jacobsen, P. V. Troshin, M. Chapman, J. Gao, C. H. Koh, S. Foisy, R. Holland, G. Rimsa, M. L. Heuer, H. Brandstatter-Muller, P. E. Bourne, and S. Willis, "BioJava: an open-source framework for bioinformatics in 2012," *Bioinformatics*, vol. 28, no. 20, pp. 2693–2695, 2012.