Similarity Search for an Extreme Application: Experience & Implementation

[1]Vladimir Mic (✉),[2]Tomáš Raček, [2]Aleš Křenek, and [1]Pavel Zezula ⋆

[1]Faculty of Informatics, Masaryk University, Brno, Czech Republic
[2]Institute of Computer Science, Masaryk University, Brno, Czech Republic
xmic@fi.muni.cz

Abstract. Contemporary challenges for efficient similarity search include complex similarity functions, the curse of dimensionality, and large sizes of descriptive features of data objects. This article reports our experience with a database of *protein chains* which form (almost) metric space and demonstrate the following extreme properties. Evaluation of the pairwise similarity of protein chains can take even tens of minutes, and has a variance of six orders of magnitude. The minimisation of a number of similarity comparisons is thus crucial, so we propose a generic three stage search engine to solve it. We improve the median searching time 73 times in comparison with the search engine currently employed for the protein database in practice.

Keywords: Similarity search in metric space · efficiency · distance distribution · dimensionality curse · extreme distance function

1 Introduction

The similarity search is quite well developed for traditional domains like texts, images, videos, and many of their sub-domains like photos of human faces and irises. Contemporary challenges can be seen in complex and quickly developing data domains studied within interdisciplinary research. This article describes our experience with the similarity search in *protein chains*. However, apart of the similarity (*distance*) function which is domain specific, we approach the problem in a generic way, as a similarity search with difficult distance distribution and expensive distance computation.

We address the search in the worldwide *Protein Data Bank* (PDB, [4]), specifically in its European version (PDBe) [2] which contains about 500,000 protein chains, and tens of thousands are added every year. The protein chain is a long sequence of *amino acids* connected with chemical bonds, entangled into a complex 3D shape (see Fig. 1 with an example). The 3D shape of a protein chain is

⋆ V. Mic and P. Zezula – This research was supported by ERDF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822). Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

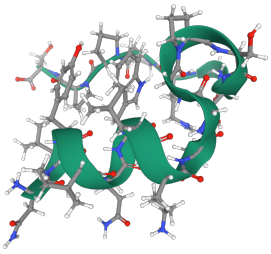


Fig. 1: 3D shape of a protein (PDB ID: 1L2Y) with a single chain. The green ribbon built upon the CA atoms presents a simplification of a complex shape of the protein chain. (Balls = atoms, sticks between them = bonds)

sufficiently described by coordinates of carbon atoms of the chain backbone – the *alpha carbons* (CA). Consequently, the similarity of two protein chains can be assessed by finding matching pairs of CA atoms, aligning them in 3D Euclidean space in the best possible way, and computing their distances.

Searching for protein chains with similar 3D structure is of utmost importance, since similar proteins are likely to have a similar biological function. The current similarity search [18] employed at the PDBe database [2] is slow as it does not use any index. Instead, it scans the whole dataset, and the distance computation is skipped just if the sizes of compared chains are as different as they prevent the chains from being similar.

An efficient protein chains search is very challenging for several reasons:

- sizes of descriptors of protein chains vary by two orders of magnitude,
- computation times of chains similarity vary by six orders of magnitude,
- the distribution of protein chains distances is extremely skewed, making the similarity search difficult,

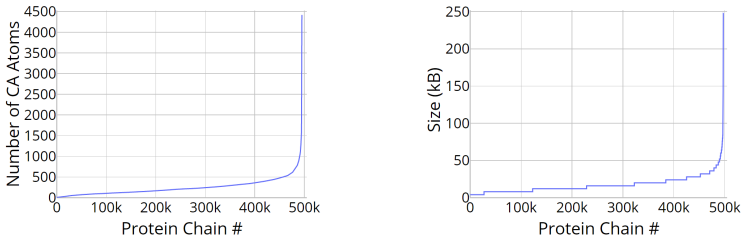
These features make an efficient generic similarity search even impossible for some query objects.

This article presents the novel search engine that runs a three-step gradual search and is available at https://similar-pdb.cerit-sc.cz. While all search techniques have been published in the past, we elaborate on their novel combination within a search engine to maximize user contentment. We define and justify our design choices to maximize the search speed and achieve top search quality.

The article is organised as follows. The similarity of protein chains and challenges of the search are described in Sec. 2. Sec. 3 describes our approach to build the search engine that maximises the user satisfaction, Sec. 4 summarizes our experiments and Sec. 5 provides conclusions of the article.

2 Similarity of Protein Chains

To formalise a pairwise similarity of protein chains, we use the metric space (D, d) defined by the data domain D and the *distance function* $d : D \times D \mapsto \mathbf{R}^+$.



(a) In number of CA atoms (b) In kB on a disk

Fig. 2: Sizes of protein chains – all 0.5 million chains are sorted according to their size, and x-axis depicts their order

The interpretation is that the bigger the distance, the less similar chains, and distances meet the properties of non-negativity, symmetry, identity, and triangle inequality [21]. This article describes the search for the most similar protein chains to an arbitrary given $q \in D$ in the dataset $X \subset D$ which consists of 495,085 protein chains from the PDBe database [2]. A snapshot was taken in September 2020.

2.1 Properties of Descriptors

The pairwise similarity of complex objects, e.g., multimedia, is not usually evaluated directly using the raw data. Instead, the characteristic features (*descriptors*) are extracted to describe objects from a specific perspective. Most of the contemporary descriptors have a fixed size which brings advantages for their processing. Especially, distances of descriptors are evaluated in almost the same time.

Descriptors which are sufficiently rich to express the 3D shape of protein chains tend to be of size that follows the size of chains. Big variance of descriptors sizes causes extreme differences in distance computation times [8]. Fig. 2a depicts the number of CA atoms in all 495,085 chains $o \in X$ after dropping extremely small chains, i.e., with less than 10 CA atoms. This is a common practice as such chains are biologically irrelevant. The median protein chain size is 207 CA atoms, but 0.97 % of chains are bigger than 1,000 CA atoms, and 0.03 % of them are bigger than 4,000 CA atoms. Fig. 2b depicts the size of protein chains on SSD, which is varying from 4 kB to 248 kB with a median 16 kB. The total size of the dataset is 8.2 GB. The variance of protein chain size has important consequences for assessing their similarity, as we discuss in the following section.

2.2 Similarity Score

There is no general agreement on a universal measure of the protein chain similarity [20, 19, 8]. We follow the method [10] implemented in the current search

service of PDBe, which is based on the *Qscore*:

$$\mathrm{Qscore}(o_1, o_2) = \frac{N_{align}^2}{\left(1 + (\mathrm{RMSD}/R_0)^2\right) \cdot N_1 \cdot N_2} \tag{1}$$

where N_1, N_2 are numbers of CA atoms in chains o_1, o_2, R_0 is an empirical constant ($3\,\text{Å} = 3 \times 10^{-10}\,\text{m}$), and N_{align} is the size of subset of CA atoms from both chains which are aligned on each other (by shifting and rotating in 3D space) to minimize their root mean square distance (RMSD), which is defined:

$$RMSD = \sqrt{\sum_{i=1}^{N_{align}} \delta_i / N_{align}}$$

where δ_i is the actual distance (in 3D Euclidean space) between corresponding CA atoms.

The alignment and RMSD computation is fairly easy [9] – the difficult part is the choice of CA subsets to minimize Eq. 1. Systematic search of all possible subsets is practically impossible due to its $O(2^N)$ complexity, therefore heuristics must be use. We use the heuristic of [11] as the current PDBe search. Review of protein alignment algorithms in [8] concludes that virtually all modern methods follow the pattern of minimizing some metric (the score) over all possible subsets of residues (i.e. 1:1 with CA), hence they have to overcome the same complexity problem, and they are comparable in speed.

2.3 Transformation of Qscores to Distances

The range of the Qscore is $[0, 1]$, so we can easily transform it to the distance:

$$d(o_1, o_2) = 1 - \mathrm{Qscore}(o_1, o_2) \tag{2}$$

This function is not a metric distance function due to the imperfection of the heuristic that estimates the Qscore. We examined 250 million pairs $d(o_1, o_2)$, $d(o_2, o_1)$ to reveal that they are equal just in 86,8 % of cases. Differences $d(o_1, o_2) - d(o_2, o_1)$ are rather small: 96.3 % of pairs differ by at most 0.01, and 99.7 % differ up to 0.05. These differences cause violations of the triangle inequality rule, and thus the similarity search based on the filtering by triangle inequalities introduces false negatives errors. While these imperfections could be almost fixed by a double distance computation: $d_m = \min\left(d(o_1, o_2), d(o_2, o_1)\right)$, it does not pay off due to the complexity of distance evaluations.

Small violations of metric postulates motivate us to use techniques based on *pivot permutations* since they are robust enough to deal small violations of metric postulates. Pivot permutation based techniques use just the order of several closest reference chains (*pivots*) to each chain to approximate its position in a space [6, 1, 16]. Usage of this type of techniques has an important connotation with the Qscore-to-distance transformation given by Eq. 2. Many transformations of a score to distance exist, and they usually swap the order:

$$\left(\mathrm{Qscore}(o_1, o_2) < \mathrm{Qscore}(o_2, o_1)\right) \implies \left(d(o_1, o_2) > d(o_2, o_1)\right) \tag{3}$$

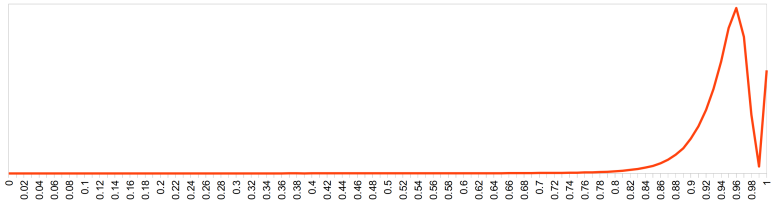


Fig. 3: Distance density

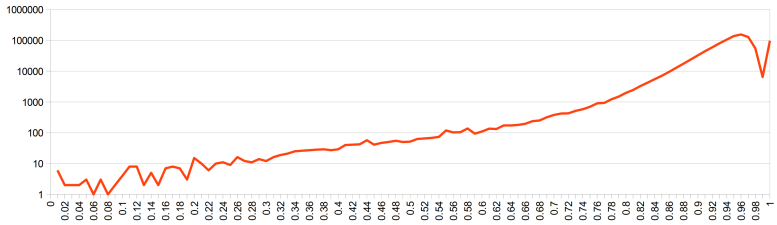


Fig. 4: Histogram of sampled distances with the logarithmic scale of y-axis

Different score-to-distance transformations just change the distribution of distances – see e.g. *the convex transforms* of distances [5, 7, 17]. Ordering of the closest pivots to an arbitrary given chain $o \in D$ is, however, the same for all score-to-distance transformations that meet Eq. 3. It is thus meaningless to elaborate on a more sophisticated Qscore-to-distance transformation, if we always use the pivot-permutation-based techniques to search the protein chains.

Beside the solutions described in this article, we also tried the filtering based on triangle inequalities. It is ineffective due to the dimensionality curse described in the following section. We also tried to apply convex transforms to the distance function given by Eq. 2. We observed a small ratio of triangle violations which, however, leads to an inadequate false reject rate despite a slow searching.

2.4 Curse of Dimensionality

The efficiency of the similarity search in complex data suffers from the "*curse of dimensionality*": The volume of the searched space quickly increases with increasing distances of nearest neighbours to query chain q. The efficiency of the similarity search thus decreases. Besides, the efficient pruning of the searched dataset is getting harder with decreasing variance of distances $d(q, o), o \in X$. Fig. 3 illustrates the density of the distance function defined by Eq. 2 – the curve is made of a sample of million distances $d(o_1, o_2), o_1, o_2 \in X$. The distribution of distances is as skewed, as 98.86 % of distances are bigger than 0.8, and 89.8 % of them are bigger than 0.9. The variance of distances is just 0.002. The *k-nearest neighbours* (kNN) queries searching for k objects $o \in X$ with minimum distances $d(q, o)$ thus cannot be evaluated efficiently in practice for query chains $q \in D$ that have kth nearest neighbour in a large distance.

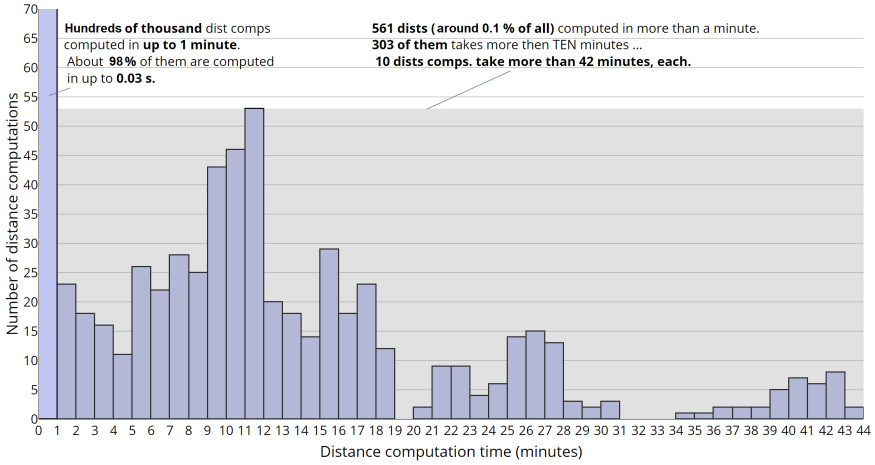


Fig. 5: The extreme times of distance computations

The protein chain descriptors are, however, actual 3D models of protein chains, and the distance function directly expresses their best alignment. The distance of chains thus well corresponds to the perceived similarity of protein chains – which often is not a case of contemporary metric space similarity models with descriptors from neural networks. The searching radius 0.5 thus figures an acceptable limitation for practitioners searching protein chains, since more distant chains are always too dissimilar. We thus focus on the searching for chains $o \in X$ that are within distance $d(q, o) \leq 0.5$, and we consider at most $k = 30$ of them. Similarity queries in this article are meant mainly as the 30NN query limited by range 0.5. In the web application, we allow redefining the k value arbitrarily, but the maximum searching radius is 0.7 to basically limit the query execution times.

Fig. 4 illustrates the distribution of the distances with the logarithmic scale of the y-axis. This plot depicts the same curve as Fig. 3, but bins of the width 0.01 are used to create the histogram of distances for which the range of the y-axis can be meaningfully depicted. The figure reveals that there are protein chains within small distances, and thus the similarity search with a limited range, e.g., 0.5, can be meaningful. This is experimentally confirmed in Sec. 4.

2.5 Distance Function Complexity

We evaluate the distance of protein chains by a heuristic [11] that estimates the Qscore. Its evaluation is more efficient than the precise Qscore evaluation, but still, it has a complexity $\mathcal{O}(N_1 \cdot N_2)$ where N_1, N_2 are the chain sizes. Therefore the distance computation time may explode if two extremely big protein chains are compared – see Fig. 2a with the protein chain sizes. Fig. 5 illustrates the extreme times of distance computations. These data are gathered from our online running search engine which temporarily stores all distances evaluated in

more than 30 ms, and persistently stores those evaluated in more than a second. Fig. 5 depicts the stored distance computations that relate to 460 different query chains – please notice that the vast majority of query executions do not store any distance computation. Axis x and y depicts the times in minutes per one distance computation and the number of observed distances, respectively. The first column visualising the distances evaluated below a minute should have a height of around half a million samples[1], and approximately 98 % of these distance evaluations take less than 30 ms. We observed 5,600 distance computations that take more than a second and less than a minute. The biggest problem is the extreme tail of around 0.1 % of distance computations which take minutes or even tens of minutes, each. Until now, we observed 10 distance computations which took more than 42 minutes, each.

We tried to skip all these extreme distance evaluations, but this results in an inability to find even very similar protein chains to some of the biggest query chains. We decided not to employ such skipping since the newly identified protein chains in the PDBe database are rather bigger ones, so the quality of the search engine could be perceived as decreasing in the future. We also observed a moderate Pearson correlation +0.46 between the distance computation times and the returned distances, which could be a motivation to skip long-lasting distances. Nevertheless, this correlation seems insufficient and influenced by an inability to effectively sample the extreme values: for instance, 6 out of 10 observed distance evaluations that take more than 42 minutes are returning distances smaller than 0.21. These are thus distances between very similar protein chains – see Fig. 4.

The best way to search the protein chains that we found is to minimize the number of the Qscore evaluations and to cache expensive distance computations.

3 Gradual Similarity Search

We provide users with three gradual query answers of increasing quality to maximise the search engine usefulness. We denote these three consecutive parts of the query execution as *phases*, and each of them returns a query answer.

- The first phase is always finished in a few seconds since it evaluates just 61 distances $d(q, p)$ of q to pivots. It is usually evaluated below 0.5 s.
- The second phase uses just 489 distances $d(q, p)$ to pivots, including those 61 from the first phase. It takes usually less than 4 s including the first phase.
- The third phase requires a variable number of distance computations with the median value 702, including those 489 from previous phases. The whole search usually takes less than 8 s, but with an extreme and necessary tail.

The results of the first and second phases thus add a negligible overhead to the third phase, since the IDs of chains likely to be similar to q figure more or less the intermediate result of the query execution.

[1] This is an estimation made as an extrapolation from other query executions. Our search engine evaluates approximately 1,000 distances per average query. The vast majority of distances is not stored, so we do not know the precise number of distances evaluated in less than a second.

3.1 Data Preprocessing & Sketches

The whole search engine is based on 512 pivots $p \in D$ that approximate the position of each protein chain in a space. We select the pivots *uniformly randomly* with respect to the number of CA atoms in protein chains. Specifically, we sort the chains $o \in X$ according to the number of CA atoms, split this list into 512 parts of the same size, and randomly pick one protein chain from each part as a pivot. All distances $d(o, p), o \in X$ between chains o and pivots p are precomputed and stored in the DB during the data preprocessing.

(a) GHP to set values in one bit (b) Two GHPs to set values in two bits

Fig. 6: The generalised hyperplane partitioning to define bits of sketches of chains

We also create and store the *sketches* of protein chains. Sketch $sk(o), o \in X$ of protein chain o is a bit-string in the Hamming space $(\{0,1\}^\lambda, h)$, and the Hamming distance of sketches approximates the distance of protein chains. We use the sketching technique denoted here as *GHP_50*, which is defined in [12]. The GHP_50 uses a single instance of the *generalised hyperplane partitioning* [21] to define each bit of all sketches $sk(o), o \in X$: a given bit of sketches $sk(o), o \in X$ expresses which of the two pivots is closer to o, so λ hyperplanes define sketches $sk(o), o \in X$ of length λ bits – see schema in Fig. 6. Pivot pairs are selected by a heuristic from a set of pivots to define approximately *balanced* and *low-correlated* bits of sketches $sk(o), o \in X$, i.e., each bit of sketches splits dataset X approximately into halves, and bits of sketches $sk(o), o \in X$ have pairwise Pearson correlations as close to 0 as possible [12].

We use two types of sketches for each protein chain $o \in X$. The *small sketches* have length 320 bits, and they are defined using just 61 pivots out of those 512 preselected. We offered 64 pivots to the heuristic to define the hyperplanes, and it did not use 3 of them. The probable cause is a vast majority of distances 1 between these 3 pivots and protein chains from the dataset X which prevent all hyperplanes defined by these pivots from defining the balanced bits. The 64 pivots that we offered were selected again *uniformly randomly* according to the size from 512 preselected pivots. In practice, sketches maximizes the memory usefulness iff λ is a multiple of 64, since we use java class *BitSet* to store sketches as an array of *longs*. For the first search phase, sketches of 320 bits created from 64 pivots provide a suitable trade-off between the time needed to create sketches, and their quality.

Table 1: The mapping of distances that we use for sketches of length 1,024 bits and $\pi = 0.75$ (the majority of lines is omitted)

Ham. dist. b	orig. dist. t	Ham. dist. b	orig. dist. t	Ham. dist. b	orig. dist. t
0 − 144	**0**	211	0.03	238	0.1
145	0.0003	...	...	...	...
146	0.0006	222	0.04	270	0.2
147	0.001	223	0.05	...	...
...	...	...	...	290	0.3
149	0.002	227	0.06	...	...
...	...	...	...	307	0.4
172	0.01	231	0.07	...	...
...	...	...	...	**340**	**0.5**
199	0.02	236	0.08	...	...
...	...	237	0.09	**562 − 1024**	**1**

We also use the *large sketches* of $o \in X$ that have length 1,024 bits and are defined by 489 out of 512 preselected pivots. Similarly, we offered all 512 pivots to the heuristic to define large sketches, and it used just some of them. Our database thus contains the following metadata for each chain $o \in X$: (1) 512 chain-to-pivots distances, (2) small sketch of o, and (3) large sketch of o.

3.2 First Phase of the Query Execution

Following sections describe the query execution, so we consider an arbitrary given query chain $q \in D$. The first phase of the search evaluates just 61 distances $d(q, p)$ to create the small sketch of q, and executes the kNN query on the small sketches. We use just a sequential evaluation of all 495,085 Hamming distances $h(sk(q), sk(o)), o \in X$; such evaluation takes about 0.15 s (per query), so the execution time of the first phase is practically given by the evaluation of 61 distances $d(q, p)$ to create small sketch $sk(q)$. Since none of these pivots is extremely big, the first phase is evaluated in up to a few seconds for an arbitrary $q \in D$.

The first phase answer consists of IDs of k chains $o \in X$ with the smallest Hamming distances $h(sk(q), sk(o))$. These IDs are immediately shown in the GUI, and we start the parallel and asynchronous evaluation of k distances $d(q, o) = 1 - \text{Qscore}(q, o)$, as well as the second phase of the search. When $\text{Qscore}(q, o)$ is evaluated, the alignment of the protein chains q and o is displayed since the Qscore computation involves the best alignment of protein chains in 3D Euclidean space. Asynchronous evaluation allows to provide some results quickly, and we stop remaining evaluations when the results of the second phase are delivered. The results are sorted dynamically according to the distances $d(q, o)$. If the distance is bigger than the specified radius of the query, the ID of chain o is hidden from the results.

3.3 Second Phase of the Query Execution

The second phase of the query execution is similar to the first one, but it uses large sketches $sk(o), o \in X$. Specifically, 489 distances $d(q, p)$ between q and pivots p are evaluated to create a large sketch of q, and all 495,085 Hamming distances $h(sk(q), sk(o)), o \in X$ are evaluated to return the result of the second phase. We, however, utilise also the minimum required Qscore to define the searching radius in the Hamming space of large sketches, and we evaluate the kNN queries with the limited searching radius in the Hamming space. We use the probabilistic model from the article [13] that approximates the mapping of distances $t = d(q, o)$ to the Hamming distances $b = h(sk(q), sk(o))$ of sketches created by the *GHP_50* sketching technique, such that:

$$\mathbf{P}\left(d(q,o) \geq t \mid h(sk(q), sk(o)) = b\right) \approx \pi \tag{4}$$

where π is the probability empirically set to 0.75 [13]. Intuitively, having the Hamming distance of sketches, the mapping estimates the minimum probable distance of the protein chains. Table 1 gives examples of the mapping that is used in our web application. While the mapping of distances is used just to set a disposable Hamming radius to search the large sketches in the second phase, it plays a crucial role in the third phase.

3.4 Third Phase of the Query Execution

We use a high-quality pivot permutation based index called *the PPP-codes* [16] and the *secondary filtering* by large sketches [13] in the third phase.

The *PPP-codes* index [15, 16] uses 4 independent Voronoi partitionings [21] of the metric space (D, d). Each Voronoi partitioning uses 128 pivots that are disjunctive subsets of all 512 preselected pivots. Each protein chain is indexed using all these partitionings in the main memory. Given a query chain q, the chains $CandSet(q)$ that are likely to be similar to q are determined (*the candidate set*) by a selective combination of individual Voronoi partitionings [16]. The candidate set size is given as a parameter in advance.

Usage of PPP-codes clarifies the number of 512 pivots that we use. Each Voronoi partitioning uses $512/4 = 128$ pivots to approximate position of each chain, and we need to have a few distances between 128 pivots and each protein chain smaller than 1. We found 512 pivots as the optimum number, since 768 and 124 pivots provide practically the same results as presented with 512 pivots.

Particular position of the query chain q significantly infers the performance of similarity indexes. A fixed candidate set size used for all query chains $q \in D$ does not take into account different density of chains $o \in X$ around given $q \in D$, and thus decreases the quality, or unnecessarily increases the number of distance computation in case of many query chains [13]. The *secondary filtering* of the $CandSet(q)$ by sketches can effectively reduce the $CandSet(q)$ dynamically, using the current searching radius given either by the query assignment or by the distance to the kth nearest neighbour, found so far.

We evaluate the third phase of the query execution as follows. When the second phase of the query execution is finished, we evaluate all the remaining distances of q to 512 pivots p, and search for the candidate set $CandSet(q)$ of size 5,000 ($\approx 1\,\%$ of the dataset) by the PPP-codes. When a protein chain $o \in X$ is confirmed to be in the $CandSet(q)$, we evaluate the Hamming distance $h(sk(q), sk(o))$ of large sketches. This Hamming distance is used together with the mapping illustrated by Table 1 to estimate the minimum probable distance of protein chains. If this estimated distance is bigger than the current searching radius, o is discarded. Otherwise, we evaluate the distance $d(q, o)$, and we gradually build the answer of the third phase. Building the answer is finished when the whole $CandSet(q)$, i.e. 5,000 chains, is processed.

The need to minimise the number of distance computations also clarifies the need to limit the searching radius from the beginning, i.e., to evaluate kNN + range queries instead of kNN queries. If a kNN query is evaluated, the secondary filtering is not utilised until k distances are evaluated to fill the query answer. Then the searching radius decreases gradually, usually from a high value. Immediate range limitation thus enables an effective secondary filtering from the beginning of the query execution which effectively decreases the number of distance computations. Similarly, the k value improves the effectiveness of the secondary filtering since if the query answer is full, the searching radius is given by the distance to the kNN instead of the original query range, so the secondary filtering power increases dynamically.

Table 1 illustrates an extreme power of the secondary filtering with the *GHP_50* sketches. The implicit searching radius in our application is 0.5, which is a very small distance considering the distance density depicted in Fig. 3. Distance 0.5 is mapped to the Hamming distance 340, i.e., if sketches $sk(q)$, $sk(o)$ of length 1024 bits differs in at least 340 bits, we skip the evaluation of distance $d(q, o)$. Lemma 1 and Theorem 2 in article [12] defines the mean and the variance of the Hamming distances on *GHP_50* sketches: the mean is $\lambda/2$, i.e. 512, and the variance decreases towards $\lambda/4$, i.e. 256, with the decreasing pairwise bit correlations – and they are minimised by the *GHP_50* sketching technique. Therefore, the Hamming distance of large sketches $sk(q)$ and $sk(o)$ as small as 340 is very improbable, and the secondary filtering usually discards a vast majority of the $CandSet(q)$ identified by the PPP-codes.

4 Experiments

Since we focus on kNN queries with limited radius 0.5, we report the number of nearest neighbours within this radius. We use 1,000 query chains in our experiments, that are selected in the same way as pivots, i.e., uniformly randomly with respect to the protein chain size. None of the query chains is used also as a pivot. Fig. 7 illustrates the number of nearest neighbours in the *ground truth* (i.e., the precise answer) for all 218 query chains q that have less than 30 nearest neighbours within distance 0.5. All other 782 query chains have at least 30 nearest neighbours within the distance 0.5, and we use 30 closest of them as their

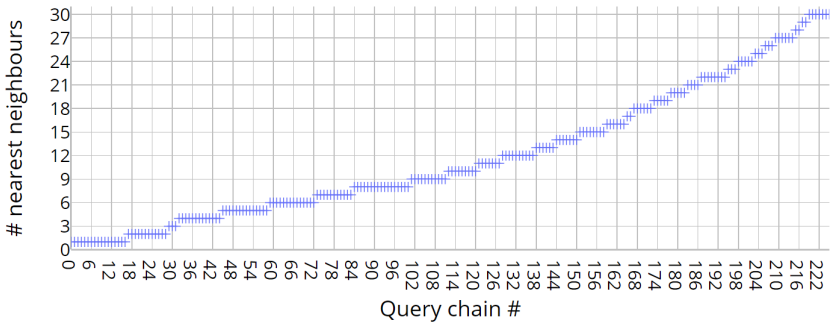


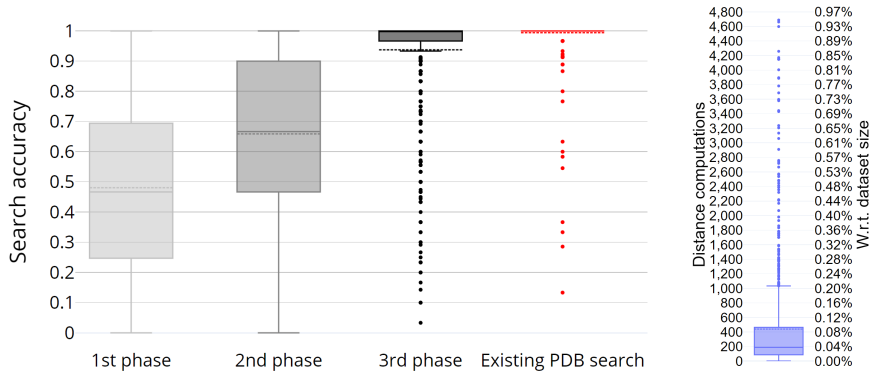
Fig. 7: Number of nearest neighbours within distance 0.5 to each query chain q. 218 out of 1,000 tested q do not have 30 nearest neighbours up to distance 0.5.

ground truth. The searching quality is described by the *accuracy*, i.e., the ratio of nearest neighbours from the ground-truth that is found. Fig. 7 clarifies that the similarity queries with the search radius 0.5 should provide non-empty answers for around 98 % of query chains despite an extreme distance density illustrated by Fig. 3. We point out that an empty answer provides useful information due to a strong relation of the distance function to the protein chain structures, as discussed in Sec. 2.4.

Box-plots in Fig. 8a depict the searching accuracy over particular query chains. The first and second phases search with a median accuracy 0.467 and 0.667, respectively. Both have a huge variance – the differences between the first and the third quartiles are 0.44 and 0.43, respectively. The third phase evaluates 700 out of 1,000 queries precisely, so it has the median accuracy 1. The average accuracy is 0.937 due to the worst query chains – averages are depicted by the dashed line for each box-plot in Fig. 8.

We evaluate 30NN queries with the radius 0.5 also by the existing PDBe search engine, but we use the setting which guarantees to find all nearest neighbour up to distance 0.3. This setting provides much faster search than the precise one, and it is still of a slightly better accuracy than our inherently approximate search with the radius 0.5 (see Fig. 8a). The speed comparisons of the search engines with these settings are thus quite fair.

Fig. 8b relates to the third phase of the query execution which uses the PPP-codes to select 5,000 candidate chains $CandSet(q)$, and filters [13] them with the large sketches. Fig. 8b illustrates that just 190 out of 5,000 candidate chains remain per median q after this filtering. The first and third quartiles are 87 and 465, respectively, the minimum number of remained chains is 2 and the maximum is 4,688. These numbers thus correspond to the only distance $d(q, o)$ evaluations conducted apart from 512 query-to-pivots distance evaluations to return the query answers with median accuracy 1 and average accuracy 0.937. Please see that this highest number of distance computations, 4,688, clarifies our



(a) Searching accuracy (b) Dist. comps.

Fig. 8: Searching accuracy and number of distance computations after the secondary filtering with large sketches in the 3rd phase

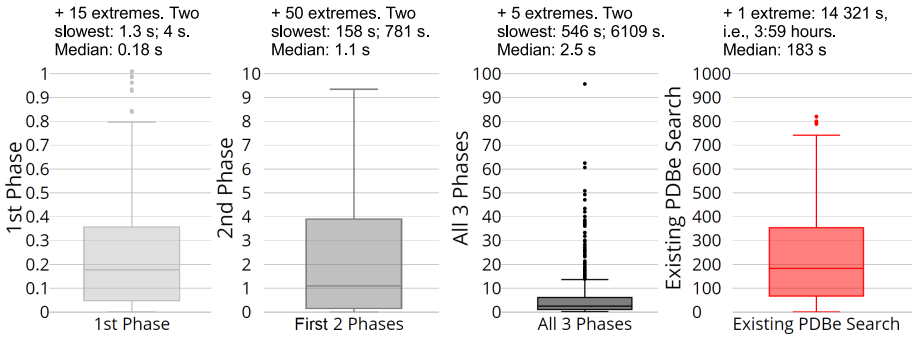


Fig. 9: Query execution times in seconds

choice of the candidate set size, i.e., 5,000 selected by the PPP-codes – we have observed query chains that utilises the vast majority of such candidate set.

Fig. 9 depicts the searching times in seconds[2]. Box-plots again describe the distribution of values over particular query chains q. The scale of the times is multiplied by 10 for each box-plot, and there are a few outliers that do not fit the scale and are described above each box-plot. All 3 phases provide their answers within 5 seconds for almost 70 % of query chains (see the third box-plot), but there is an extreme tail clearly explained by the previous analysis. 1 % of the slowest query executions (i.e., 10 queries) requires more than 49 seconds, each. The median searching time of all 3 phases is 2.5 s. The fourth box-plot depicts

[2] No caching is used here except a re-using the distances evaluated in the previous phases of the same query execution.

the searching times of the current PDBe search engine. The median is 183 s, i.e., 73 times slower.

The suitability of our approach is confirmed by an ability of the first and second phases to deliver results of a lower accuracy but much faster. The most difficult query chain out of 1,000 examined is evaluated with the following accuracy and times:

	1st phase	1st and 2nd phase	All 3 phases	PDBe engine
Accuracy	0.46	0.66	0.93	1
Time	4 s	13 min	1:40 hours	3:59 hours

5 Conclusions

We described our experience with the similarity search in extreme metric space which strongly suffers from the variance of objects size and the similarity function complexity. The times needed to evaluate the pairwise *protein chain* similarity vary by 6 orders of magnitude from 1 ms to more than 43 minutes. The number of similarity comparisons thus must be minimised. Providing users with intermediate query results of increasing quality effectively mitigates user inconvenience, and we evaluate queries in 3 consecutive phases. Each phase introduces the minimum overhead to the following phases, and the first query results are always delivered in a couple of seconds. Since the similarity query execution in 495,085 protein chains evaluates just hundreds of distance computations for a majority of query chains, we achieve median searching time 2.5 s which is 73 times faster than the result of the engine employed in the "Protein Data Bank in Europe". As the future work, we would like to develop a distributed search engine for real-life usage [14, 3].

References

1. Amato, G., Savino, P.: Approximate similarity search in metric spaces using inverted files. In: 3rd International ICST Conference on Scalable Information Systems, INFOSCALE 2008, Vico Equense, Italy, 2008. p. 28. ICST / ACM (2008)
2. Armstrong, D.R., Berrisford, J.M., Conroy, M.J., Gutmanas, A., Anyango, S., Choudhary, P., Clark, A.R., Dana, J.M., Deshpande, M., Dunlop, R., Gane, P., Gáborová, R., Gupta, D., Haslam, P., Koča, J., Mak, L., Mir, S., Mukhopadhyay, A., Nadzirin, N., Nair, S., Paysan-Lafosse, T., Pravda, L., Sehnal, D., Salih, O., Smart, O., Tolchard, J., Varadi, M., Svobodová-Vařeková, R., Zaki, H., Kleywegt, G.J., Velankar, S.: PDBe: improved findability of macromolecular structure data in the PDB. Nucleic Acids Research **48**(D1), D335–D343 (11 2019)
3. Batko, M., Novak, D., Falchi, F., Zezula, P.: Scalability comparison of peer-to-peer similarity search structures. Future Gener. Comput. Syst. **24**(8), 834–848 (2008)
4. Berman, H.M., Westbrook, J., Feng, Z., et al.: The protein data bank. Nucleic Acids Res. **28**(1), 235–242 (2000)
5. Bernhauer, D., Skopal, T.: Analysing indexability of intrinsically high-dimensional data using trigen. In: Similarity Search and Applications. pp. 261–269. Springer International Publishing, Cham (2020)

6. Chávez, E., Figueroa, K., Navarro, G.: Effective proximity retrieval by ordering permutations. IEEE Trans. Pattern Anal. Mach. Intell. **30**(9), 1647–1658 (2008)
7. Connor, R.C.H., Dearle, A., Mic, V., Zezula, P.: On the application of convex transforms to metric search. Pattern Recognit. Lett. **138**, 563–570 (2020)
8. Deng, L., Zhong, G., Liu, C., et al.: MADOKA: an ultra-fast approach for large-scale protein structure similarity searching. BMC Bioinformatics **20**(662) (2019)
9. Kearsley, S.K.: On the orthogonal transformation used for structural comparisons. Acta Crystallogr. A **A45**, 208–210 (1989)
10. Krissinel, E., Henrick, K.: Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions. Acta Crystallographica Section D: Biological Crystallography **60**(12), 2256–2268 (2004)
11. Krissinel, E.: Enhanced fold recognition using efficient short fragment clustering. Journal of molecular biochemistry **1**(2), 76 (2012)
12. Mic, V., Novak, D., Zezula, P.: Designing sketches for similarity filtering. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW). pp. 655–662 (Dec 2016)
13. Mic, V., Novak, D., Zezula, P.: Binary sketches for secondary filtering. ACM Transaction on Information Systems **37**(1), 1:1–1:28 (Dec 2018)
14. Novak, D., Batko, M., Zezula, P.: Large-scale similarity data management with distributed metric index. Inf. Process. Manag. **48**(5), 855–872 (2012)
15. Novak, D., Zezula, P.: Performance study of independent anchor spaces for similarity searching. The Computer Journal **57**(11), 1741 (2014)
16. Novak, D., Zezula, P.: PPP-codes for large-scale similarity searching. Trans. Large-Scale Data- and Knowledge-Centered Systems **24**, 61–87 (2016)
17. Skopal, T.: Unified framework for fast exact and approximate search in dissimilarity spaces. ACM Trans. Database Syst. **32**(4), 29 (2007)
18. Velankar, S., Best, C., Beuth, B., Boutselakis, C.H., Cobley, N., Sousa Da Silva, A.W., Dimitropoulos, D., Golovin, A., Hirshberg, M., John, M., Krissinel, E.B., Newman, R., Oldfield, T., Pajon, A., Penkett, C.J., Pineda-Castillo, J., Sahni, G., Sen, S., Slowley, R., Suarez-Uruena, A., Swaminathan, J., van Ginkel, G., Vranken, W.F., Henrick, K., Kleywegt, G.J.: PDBe: Protein Data Bank in Europe. Nucleic Acids Research **38**(suppl_1), D308–D317 (10 2009)
19. Winn, M.D., CC, C.C.B., Cowtan, K.D., et al.: Overview of the CCP4 suite and current developments. Acta Crystallogr. **D67**, 235—242 (2011)
20. Yang, A., Honig, B.: An integrated approach to the analysis and modeling of protein sequences and structures. i. protein structural alignment and a quantitative measure for protein structural distance. J Mol Biol, **301**, 665—678 (2000)
21. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search – The Metric Space Approach, Advances in Database Systems, vol. 32. Springer (2006)