Computational Assessment of Single-Molecule Protein Sequencing

Yao Yao

MSc Bioinformatics track, The Delft Bioinformatics Lab, Department of Intelligent Systems, Delft University of Technology, Delft, The Netherlands.

ThesisDefence: 2 July 2014 9 AM, Snijderszaal room (LB 01.010) EWI

Supervisors: Dick de Ridder, Margreet Docter, Chirlmin Joo, Marcel Reinders

ABSTRACT

Motivation: A single-molecule protein sequencer, which labels only 2 out of 20 amino acids and uses single-molecule TIRF microscopy to measure the order of these fingerprints, opens the door to identify proteins with high fidelity using only a small quantity of sample. From the fingerprint, a key challenge is to detect which protein was measured.

Results: We present a first tool that efficiently retrieves the protein sequences by just comparing the fingerprints, even in the presence of a high error rate. A clustering method is first employed to reduce the redundancy of the database. Given a fingerprint, our algorithm employs an efficient filtering strategy to identify potential matches and a dynamic programming to verify the matches found. These matches are then mapped back to the original fingerprint database to get the final proteins.

We analyzed the detection behavior on simulated data and investigated how the use of additional information may improve the performance. In addition, we tested whether the fingerprint information is sufficient to solve other problems, such as distinguishing whether a human cell sample contains bacterial or viral proteins.

Availability: This fingerprints detection tool FPD has been written in C++. Source code is available at http://homepage.tudelft.nl/g4b23

Contact: yaoyao0221@gmail.com

1 INTRODUCTION

Detection and quantification of protein expression levels is of importance to understand the essential molecular processes within a cell. For example, some proteins, which are alternatively spliced or post-translationally modified, are thought to be the key for some diseases. Knowing the structures of such proteins could help us understand the diseases. Recently, two international teams produced the first drafts of human proteome (Kim *et al.*, 2014; Wilhelm *et al.*, 2014). Both teams discovered a number of novel proteins and were able to identify proteins encoded by 84% to 92% of all the genes in the human genome predicted to encode proteins. However, many proteins are only present in low concentrations, which cannot be measured using current protein sequencing techniques: Edman degradation (Edman and Begg, 1967) and mass spectrometry (Berg *et al.*, 2002). Both rely on a large and highly concentrated sample of the protein to determine the amino acid order correctly (Steen and Mann, 2004; Berg *et al.*, 2002) (see Supplementary Material Section 1.1 about these existing techniques). Thus, sequencing at the single-molecule level is needed to detect proteins present in minute quantities.

Chirlmin Joo lab is developing a single-molecule finger-printing protein sequencer, which labels only 2 out of 20 amino acids, cystines (C) and lysines (K), and uses single-molecule TIRF microscopy to detect these fingerprints. Each of the labeled amino acid produces a specific spectral signal as it is sequenced. The signals measured at each position on the slides are imaged and are then converted into CK bases (see Supplementary Material Section 1.2 for more detailed description). This novel technique opens the door to identify proteins with high fidelity using only a small sample. Because of some technical barriers we encountered during the process, such as inefficient labeling and fast translocation, there is no formal data right now.

From the fingerprint, the challenge is to retrieve the actual protein based on the CK fingerprint detected, even in the presence of a high measurement error rate. Although numerous applications comparing protein sequences based on the 20 different amino acids (Altschul *et al.*, 1990; Thompson *et al.*, 1997; Needleman and Wunsch, 1970), none of these are equipped to use only few amino acids. When represented by only 2 amino acids, some proteins are not distinguishable, e.g. some proteins that are evolutionarily related. Accordingly, it is of importance to provide solutions for these indistinguishable fingerprints. Finally, comparing a fingerprint to a full protein database is too slow since the database can become very large (more than 56 millions sequences in UniProt (Magrane and Consortium, 2011)). Hence, we need to improve the time efficiency when searching against a large database.

The clinical use of this protein sequencing approach could provide accurate diagnosis of diseases. Hence, in this paper, we use the human proteome database. Here, we present a tool that retrieves proteins form this database by searching the fingerprints, even in the presence of a high error rate. First, a clustering method is employed to cluster similar proteins in the database with respect to their amino acid sequences to reduce the redundancy of the fingerprint database. An efficient filtering strategy is used to identify potential matches, which significantly improves the average running time of our algorithm, followed by a dynamic programming method, which verifies those potential candidates to find matching proteins. Finally, a post-processing step retrieves proteins with identical fingerprints with those matches in the original database.

We analyzed the performance of our method on simulated data and investigated how additional information, if available, influences performance. In addition, we tested whether the fingerprint approach can solve other problems, such as detecting whether a human cell sample contains bacteria or virus proteins.

2 METHODS

The inputs to our method are a reference database, a query fingerprint and an *error level* bound. The alphabet set is Σ ={'C', 'K'}, since we only comparing the fingerprints. Let Q denotes the query with length L_Q, $R_x \in \mathbf{R}$ denotes the xth reference in the database $\mathbf{R}$ with length $L_R^x \in \mathbf{L_R}$. The error level α is defined as the ratio of the number of errors (k) between two fingerprints to the length of the query L_Q. The *distance* $D(R_x, Q)$ between a reference R_x and query Q is the minimal steps to transform Q into R_x. It is used to calculate the dissimilarity between them, which is also referred to as the number of errors between two fingerprints. Formally, given Q, $\mathbf{R}$, and α, the problem is to find all $R_x \in \mathbf{R}$ for which $D(R_x, Q)$ is smaller than k, where L_Q is the length of the query L_Q and k equals to $\alpha \times L_Q$.

Given the inputs, the algorithm takes four steps to search for the actual matches as illustrated in Fig. 1. First, *a)* a clustering step is used to cluster similar proteins in the original database with respect to their amino acid sequences, hence reduce the redundancy of the fingerprints. It is done before searching the query against the database. In the following paper, $\mathbf{R}$ is specifically used to represent the after clustering database. Then *b)* a combined filtration strategy is applied to identify potential references in $\mathbf{R}$; and *c)* all remaining references are examined for possible matches; finally, *d)* a post-processing step retrieves proteins with identical fingerprints with those matches in the original database. These steps will be discussed in more detail in the section below.

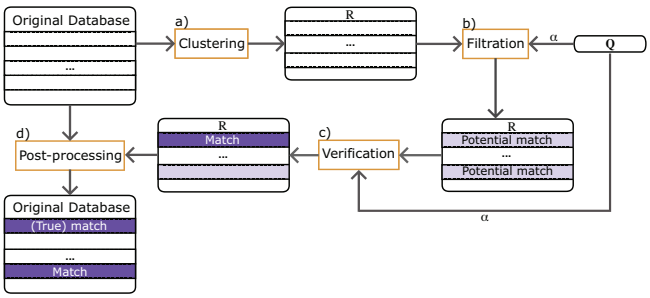


Fig. 1: Overview of the fingerprint retrieval algorithm. The algorithm takes as input a protein fingerprint database $\mathbf{R}$, a query Q and an error level threshold α.

2.1 Preprocessing: grouping related sequences using clustering

Intuitively, there is a significant loss of information due to the representation of 20 amino acids protein sequences using only 2 amino acids. For example, only 89.8% of proteins are uniquely represented by CK fingerprints (see Supplementary Material Section 5.6 for other possible 2 amino acids representations). The remaining proteins thus are not uniquely distinguishable by their fingerprints.

When two proteins have a significant degree of sequence similarity, they are likely to share an evolutionary origin. Our assumption is that two evolutionarily related proteins are more likely to have similar fingerprints (see Supplementary Material Section 2.1). Thus, by clustering proteins in the original amino acid database, we may capture some of the protein family information of those indistinguishable fingerprints and also reduce the redundancy of the fingerprints. Representative fingerprints of each cluster are retained after clustering, which are used in the subsequent three phases.

2.1.1 Greedy neighbor clustering method The algorithm starts by pairwise alignment of all full sequences using a Needleman-Wunsch (N-W) dynamic programming method. For each pairwise alignment, we compute the *sequence distance (SD)* as the number of errors (substitutions, insertions and deletions) in the alignment, and *sequence percentage identity (SPI)* is then defined as the ratio between the number of exactly matching residues and the length of the alignment. In the algorithm, the SPI between two full sequences x and y is calculated as follows:

$$SPI_{(x,y)} = 1 - \frac{SD_{(x,y)}}{L_{(x,y)}}, \tag{1}$$

where the $L_{(x,y)}$ is the length of the alignment between sequence x and y.

If the SPI exceeds a certain threshold, the two full sequences that are compared are considered to be neighbors. In this way, a neighbourhood relationship of all input sequences is determined. This matrix is used in a greedy neighbor clustering process. The process starts with a first sequence as a cluster member and puts any other sequence into that cluster if the sequence is a neighbor of at least one sequence already in the cluster. This procedure is repeated on all remaining sequences until all sequences are clustered. Next, sequences representing each cluster are extracted. Unlike conventional methods, in which only the longest amino acid sequence is used, we keep all different fingerprints in a cluster as representatives. As a result, we could acquire some of the protein family information from the clustering without losing information on the variation within a cluster.

2.1.2 Linear SPI threshold model We adapt the threshold for short fingerprints for two reasons. *1)* For some short lengths ($l < 9$) there are more fingerprint occurrences than the maximum distinguishable possibilities 2^l, where l represent the length. For instance, there are 136 fingerprints of length 5 in the database (shown in Fig. 3), while there are only $2^5 = 32$ different possibilities. Additionally, *2)* For short amino acid sequences, a higher SPI threshold means allowing a very low number of SD, which more often becomes zero. In such case these sequences are not clustered with similar sequences other than itself. Thus we propose to treat sequences differently according to their fingerprint length, namely by modelling the threshold as a linear function of their fingerprint length. A lower threshold for shorter fingerprints can cluster similar sequences together and reduce the redundancy, and a higher threshold for longer sequences helps to prevent unique fingerprint from being clustered.

From the protein length distribution, we learn that when the fingerprint is longer than 10 (about one-fourth of the average fingerprint length in Fig. 3), the ratio of the total number of sequences over the number of possible fingerprints is nearly zero. For those sequences, a higher SPI threshold helps to preserve the unique fingerprints. A lower threshold is more suitable for $1 \leq l < 10$ since it clusters some proteins together and reduces the redundancy for these short lengths. As a result, we model the threshold as a linear function of the fingerprint length until one-fourth of the average fingerprint length of the database:

$$\theta_l = \begin{cases} T_{low} + \frac{T_{high} - T_{low}}{L-1}(l-1) & , 1 \leq l < L; \\ T_{high} & , l \geq L. \end{cases} \tag{2}$$

where θ_l is the threshold corresponding to length l, T_{low} and T_{high} are two user specified parameters, and L is one-fourth of the average fingerprint length.

2.2 Filtration: eliminating uninteresting sequences

The optimal algorithm for retrieving fingerprints from a database is dynamic programing. Unfortunately, it is quite slow on ordinary computers, so many heuristic and hardware alternatives have been developed, such as

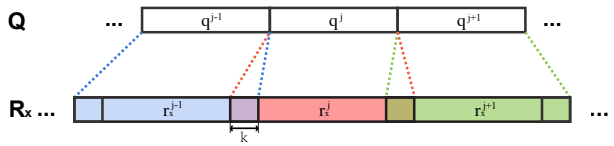


Fig. 2: Consecutive pieces q^{j-1}, q^j and $q^{j+1}$of query Q and their corresponding pieces r_x^{j-1}, r_x^j and r_x^j in reference R_x. For each query piece, it is compared within a limited range in the reference, which is length k larger on left and right.

FASTA (Pearson and Lipman, 1988), BLAST (Altschul *et al.*, 1990) and methods using hardware technologies (Farrar, 2007; Szalkowski *et al.*, 2008; Manavski and Valle, 2008). These methods have significantly reduced the running time, however, at the cost of reduced sensitivity or very expensive hardware.

In order to reduce the running time without affect the sensitivity of the algorithm, filtration is used to remove those references that definitely can not match the query fingerprint with distance smaller than or equal to k.

Filtration exploit the fact that it is much easier to tell that a reference fingerprint does *not* match a query fingerprint than to tell it matches. Typically, it uses a simple and highly efficient filter criterion to analyze the reference sequences. Only small number of R_xs that have a reasonable level of similarity to Q will be kept for further analysis.

2.2.1 q-gram partial exact matching This filtration method combines two algorithms, partial exact matching and q-grams counting. The query fingerprint Q is divided into $(k+1)$ pieces $q^0, q^1, ..., q^k$, where k equals to $\alpha \times L_Q$, so there must be at least one piece that appears exactly in a reference R_x (Wu and Manber, 1992). If, for R_x, no piece of Q appears exactly, it is discarded (for more detail, see Supplementary Material Section 3.3).

The q-gram counting method is a faster filtration method, which compares the q-grams of two fingerprints for filtration. A *q-gram* (Ukkonen, 1992) in the alphabet set Σ is any string in Σ^q, where Σ^q is the set of all possible strings of length q over Σ. For example, the 2-grams of sequences $CCCCKK$ are CC, CK and KK. The q-gram distance is defined as the sum of the absolute difference between each q-gram occurrence. If the q-gram distance is smaller than or equal to $2qk$, R_x is marked as a potential match (Ukkonen, 1992); otherwise, R_x can be filtered out (see Supplementary Material Section 3.4 for more detailed description).

We combined the partial exact matching and q-gram counting to decide whether there exists at least one piece in Q that appears exactly in a piece of R_x, a different distance function between two pieces of R_x and Q, r_x^j and q^j, based on their q-grams is defined below to design a more efficient filtration approach.

$$D_{qpm}(r_x^j, q^j) = \sum_{\nu \in \Sigma^q} \max(G(q^j)[\nu] - G(r_x^j)[\nu], 0), \tag{3}$$

where ν is a q-gram and $G(r_x^j)[\nu]$ and $G(q^j)[\nu]$ denote the total number of times ν occurs in r_x^j and q^j, respectively.

For each piece q^j in query, the corresponding piece r_x^j contains the same positions in reference with additional k positions on the two sides of r_x^j as shows in Fig. 2. It is sufficient to to comparing the r_x^j in reference with the q^j in query to determine whether the piece q^j appears in the reference R_x, since k errors cannot alter more than k positions. Since a query piece is searched in a limit range in a reference, it can discard more entries in the reference database compare to the partial exact matching method, where the q^j is compared with the whole reference.

The distance between a piece q^j in query Q and the corresponding piece r_x^j in R_x is computed to determine if the R_x is a potential match. Given a query Q and a reference R_x. For each q^j and its corresponding r_x^j, we checked whether any q-grams occur more often in q^j than in r_x^j. If not, the $D_{qpm}(r_x^j, q^j)$ is zero. If this happens for at least one piece q^j, then R_x is a potential candidate; otherwise, R_x is not a potential candidate.

Combining partial exact matching and q-grams counting methods, allows us to filter more effectively when q is small. However, since string comparison is used to distinguish different q-grams, when q gets bigger, the time needed becomes longer. In our implementation, we took advantage of the fact that comparing two numbers, by comparing individual bits using any of AND, XOR, 2's complements, etc, which takes at most a couple of instruction cycles, is much faster than comparing two strings representing the same numbers. We transformed the fingerprints into decimal numbers using a binary-to-decimal hashing function. Given fingerprint $x = x_1x_2x_3...x_n$, the corresponding decimal number is denoted by $dec(x)$:

$$dec(x) = 2^n + \sum_{i=1}^{n} 2^{n-i} B_i, \quad \text{where} \quad B_i = \begin{cases} 0 & \text{for } x_i = C \\ 1 & \text{for } x_i = K \end{cases}. \tag{4}$$

A single bit 1 is added at the left side to ensure a one-to-one correspondence between the resulting decimal number and a fingerprint sequence. By using this hashing method, we can efficiently determine different q-grams and count these occurrences.

2.2.2 Combining strategy The filtration method described above has its strengths and weaknesses. Although it works well and more efficient for most of the cases (see Table 2), it is slower comparing to a length filtering, which quickly find the potential matches by checking whether the length difference between a reference and query $|L_R^x - LQ| \leq k$. For example, let the query Q $CCKKK$ be compared to a reference R_x $CKKKKK$ with an error level of 20% (meaning k is 1) using the 2-gram partial exact matching method. The query is divided into $k+1 = 2$ pieces, $q^1 = CC$, $q^2 = KKK$, and the corresponding pieces in the reference are $r_x^1 = CKK$ and $r_x^2 = KKKKK$. Since $D_{qpm}(r_x^2, q^2) = 0$, then q^2 appears exactly in r_x^2, thus, R_x will not be discarded. However, the reference clearly is not a match, since it takes at least 2 steps (a substitution and a deletion) to transform R_x into Q. By employing a frequency distance filtering method, which discards a reference R_x by checking whether the maximum difference between symbol occurrences is bigger than k (in Supplementary Material Section 3.2) . Since the frequency distance $D_{fd}(R_x, Q) = 2$, which is bigger than 1, then the reference R_x can be filtered out easily. For this particular example, frequency distance works but the 2-gram partial exact matching does not. Thus, we cannot say which filtration method is superior to the others, but combining them can be more effective.

The basic idea of our combination strategy is to use different filtration approaches together for different error levels. A simple length filtering and frequency distance filtering are faster than a q-gram partial exact matching method. Hence, when the error level is smaller than 5%, length filtering and q-gram partial exact matching are employed. When the error level is larger than 5%, we add frequency distance filtration method. This makes our filtration quite efficient so that a large number of references are eliminated.

2.3 Verification: finding matches

In this phase, the remaining potential matches are examined by an N-W dynamic programming approach considering the set of possible error types. In our analysis, there are four types of errors might occur: deletion, insertion, mismatch an amino acid with another one (substitution), and swapping (transposition). Matching an amino acid at a position can also be seen as a substitution with the same amino acid (for more detail, see Supplementary Material Section 4.1).

2.3.1 Computation of $D_{i,j}$ The dynamic programming algorithm is designed to provide the optimal alignment between two sequences, i.e. an alignment with long regions of identical amino acid pairs and very few mismatches and gaps. As the sequences become more dissimilar, more

mismatched amino acid pairs and gaps should appear. To find the optimal alignment, a dynamic programming matrix D needs to be calculated. Each element $D_{i,j}$ represents the maximum score of aligning the substrings $Q[1...i]$ and $R_x[1...j]$. Let c denote the costs of the four operations. The base cases, $D_{0,j}$ and $D_{i,0}$, are defined as $(c_{del} \times j)$ and $(c_{ins} \times i)$ for all $1 \leq j \leq L_R^x$ (length of R_x) and $1 \leq i \leq L_Q$ (length of Q) respectively. Then considering the four possibilities, the $D_{i,j}$ is updated using the following recursive relation,

$$D_{i,j} = \max \begin{cases} D_{i-1,j-1} + c_{sub} \\ D_{i-1,j} + c_{ins} \\ D_{i,j-1} + c_{del} \\ D_{i-2,j-2} + c_{swap} \end{cases} \quad (5)$$

The cost for each operation are set differently based on the estimation of how likely each error could happen in our measurements. Because currently deletions caused by low labelling efficiency are the dominating errors, followed by insertions, transpositions and substitutions, we choose a relatively low penalty (negative) for deletions and higher penalties for transpositions and mismatches. For the matching positions, the costs of them are positives (see Supplementary Material Section 4.2). Note that the penalties can be adapted at any time.

2.3.2 Matches and ture matches By memorising the solutions to the subproblems for $1 \leq j \leq L_{R_x}$ and $1 \leq i \leq L_Q$ stored in the dynamic matrix, we can recursively compute the maximum score of aligning R_x and Q. Therefore we find the score of the optimal alignment of the two sequences starting from the maximum value in the last row or last column. Maintain a 'shadow matrix' of traceback pointers in the recursion, so that we remember which case was used to calculate every cell $D_{i,j}$. From the detailed alignment, the numbers of errors for different types as well as the total number of errors can be calculated. The distance between the query and the reference $D(R_x, Q)$ is the total number of errors (see Supplementary Material Section 4.3 for an example). If this distance is smaller than k, the reference sequence R_x is considered as a *match*. Otherwise, it is not a match of the query sequence with the error level α.

By using a post-processing phase, not only the matches that are preserved after clustering are retrieved, but also the matches in the original database which were clustered with those representative fingerprints. This is done by mapping the matches found to their clusters, and finding the exact matches within the same cluster.

A *true match* is found when the match is the exact query protein. If a match has the same fingerprint but a different amino acid sequence, it is not considered to be a true match. In our analysis, this is determined by checking the protein accession codes.

2.3.3 Spacing model Thus far, only fingerprints acquired from the measurements are used in the filtration phase. Ideally, additional information can be deduced from the measurements, the spacings between two read-outs. The spacings are non-labeled amino acids between two labeled ones, which show a different pattern in the measurement. Here we explore ways of taking the spacing information into account. There are two ways to do so: by considering whether or not spacings occur between two read-outs, or by considering the length of spacings between two read-outs.

1. *Occurrence of spacing:* If there is a spacing, it means there are some non-labeled amino acid between two read-outs. Since the measurement speed is unknown, gathering information on the occurrences of spacings is easier than estimating the length of spacings. If two read-outs are adjacent to each other, no spacing occurs. 81.8% of sequences have at least two read-outs next to each other, and the average number of such cases is 4.4 per sequence. This observation shows that including the occurrences of spacing, we should be able to improve the performance for these sequences.
2. *Length of spacing:* In this case, the number of non-labeled amino acids in the amino acid sequence is measured. From the sequencing signals, we can not easily determine the start or the end of proteins in the time trace if they do not correspond to a labelled amino acid. Thus, the starting and ending non-labeled amino acids are not included when we construct the fingerprint with length of spacing. Using the length of spacing, we should be able to improve the performance on most of the sequences even further.

The penalties used to incorporate spacing information in the verification step are described in detail in Supplementary Material Section 4.2.

Two distances between query and reference are calculated to examine whether a reference sequence is a match. One is the distance between the with spacing information fingerprints, which gives the total number of errors between them. Other one is the distance between the CK fingerprints, which gives the number of errors occur at C and K positions. If both distances are smaller than the $k^{'} = (\alpha \times L_Q^{'})$ and $k = \alpha \times L_Q$ respectively, where the $L_Q^{'}$ is the length of the with spacing information query fingerprint, L_Q is the length of the CK fingerprint and $k^{'}$ represents the number of errors allowed between two fingerprints with spacing information, then the reference sequence R_x is a match.

3 RESULTS

In this section, we present the performance of our method. We first describe the characteristics of the database used in the analysis, and then compare results of our clustering and filtration methods to other methods. Furthermore, we test the efficacy of our sequence detection method with simulated human sequences and other databases.

3.1 Database

We downloaded the reviewed human complete proteome (reviewed only) from Uniprot release 2014.04 (Magrane and Consortium, 2011) and used it to test our algorithm. It contains 20,264 different proteins; the average amino acid sequence length is 559.34 and the average fingerprint length is 44.87. The length distribution of the amino acid sequences and fingerprints are shown in Fig. 3. The ratio between the number of fingerprints of length l and the number of possible fingerprints, $\frac{N_l}{2^l}$, is also shown. Clearly, for sequences shorter than 9, $\frac{N_l}{2^l} > 1$, meaning it will be impossible to distinguish some of those fingerprints. Since Ks occur 2.49 times more frequent than Cs in the database, this is the fact that redundancy for fingerprints contains more Ks occurs even for long sequences. Although Cs occur relatively rare, it is still likely to have duplicated fingerprints contains more Cs for long sequences. For example, when $l = 22$, there are several indistinguishable fingerprints (see Supplementary Material Section 5.1). 14 proteins have no Cs or Ks at all, those proteins are removed for future analysis since they cannot be measured. The percentage of unique fingerprints in the original database is 89.8% after remove those proteins.

3.2 Clustering effectively reduce fingerprint redundancy

In our algorithm *SMGN*, we set T_{low} to 50% and T_{high} to 90% (see Supplementary Material Section 2.2 for the reasoning on the choice of parameters). To examine the performance of our clustering method, we compared it to two available applications, CD-HIT (Huang *et al.*, 2010) and BLASTClust (Altschul *et al.*, 1997). We

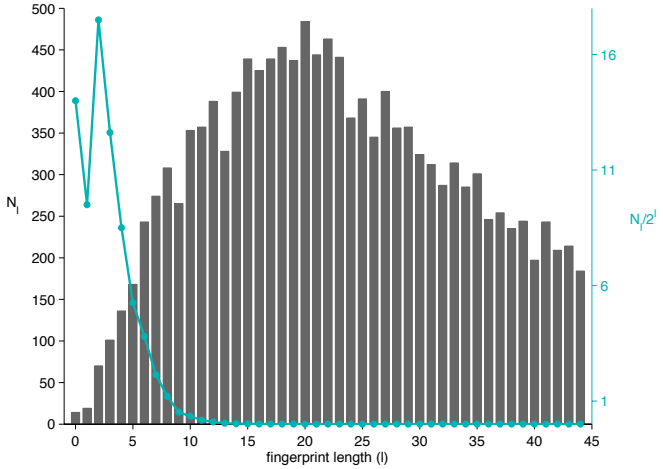


Fig. 3: The fingerprint length distribution of human complete proteome and the ratio between the number of fingerprints N_l and 2^l different fingerprints of length l.

evaluate the performance of these methods based on two aspects, *uniqueness* and *loss ratio*. They are defined as follows:

$$Uniqueness = \sum_{l \in \mathbf{L_R}} \frac{N_u^l}{N^l},$$
$$Loss\_ratio = 1 - \frac{Nd_R^C}{Nd_R}, \tag{6}$$

where N_u^l is the number of uniquely presented fingerprints of length l, N^l is number of fingerprints of length l, and Nd_R^C and Nd_R are the number of different fingerprints in the after clustering database and original ones respectively. The higher the uniqueness and the lower the loss ratio the better.

CD-HIT (Huang *et al.*, 2010) uses a greedy algorithm to cluster sequences and remove redundancy. Comparing with the greedy neighbor clustering method we used, this method puts a sequence into a cluster only if it is a neighbor to the longest amino acid sequence in the cluster. To compare with our method, the cut-off threshold is set to 90% using a local sequence percentage identity and the minimum alignment coverage for both sequences to 90%. A greedy method *SM-CDHIT* which used a linear threshold model, and different representatives is also implemented in order to compare with CD-HIT. BLASTClust (Altschul *et al.*, 1997) uses a greedy neighbor clustering method with local alignment methods to clustering sequences. The two parameters, alignment coverage and local sequence percentage identity of this method are set to be 90% and 90% for comparison. The database used here is 5000 random entries in the human complete proteome database (hm5000).

Table 1 shows the uniqueness and loss ratio of the four methods. Obviously, our clustering method SMGN can get the best uniqueness without losing any CK information. And the two clustering methods using the linear SPI threshold also outperform CD-HIT and BLASTClust that used only a single threshold based on the fact that the uniquenesses are higher than the two existing tools. A Venn diagram shows the overlap of the representatives of these four methods in Supplementary Material Fig. 10 to illustrate the degree of similarity of the sets of representatives obtained by these different programs.

Table 1. Comparison of uniqueness and loss ratio of the database after using the four clustering algorithms with the original database.

Method	No. of clusters	N^l	$Uniqueness$	$Loss\_ratio$
hm5000	×	5000	89.1%	×
CD-HIT	4827	4827	91.3%	1.9%
BLASTClust	4819	4819	91.2%	2.0%
SM-CDHIT	4829	4918	91.3%	0
SMGN	4804	4915	91.4%	0

3.3 Filtration can speed up detection of low error levels

The efficiency of filtering algorithms is very sensitive to the error level. Most filters work very well at low error levels but quickly become less efficient with increasing error levels. This is related to the amount of R_xs that filters are able to discard (Navarro, 2001). When evaluating filtering algorithms, it is important not only to consider their *time efficiency* (running time per query t) but also their tolerance for errors. Hence, the *filtration efficiency* η, which is the total number of matches found divided by the total number of potential matches retained by the filtration algorithm is examined as well.

We evaluated our combined filtration method 6GPMFDL on the hm5000 database and compared the results with those obtained by other filtration methods, including a simple length filtering OnlyL, frequency distance (Kahveci and Singh, 2001) with length FDL, partial exact matching (Wu and Manber, 1992) with length PML, 5-gram counting (Ukkonen, 1992) with length 5GCL and our 6-gram partial exact matching method 6GPML. We simulated data with different error level between 0% to 20% to test all the methods. A parameter q controlling the size of q-grams is needed for several methods. The choice of q should be based on the filtration efficiency and the time need to process it. Basically, different q-grams will have different efficiencies. In the error-free case, the performance of various q of our filtration method is listed in Supplementary Material Table 7. When q is 6, we will be able to obtain the highest filtration efficiency without compromising speed. For q-gram counting method, 5 is assigned to q (see Supplementary Material Table 6). Below, we used the above mentioned qs for the evaluation.

The experimental results show that our combined method 6GPMFDL filters out more references than most other methods. The second best approach, our method 6GPML has the best filtration efficiency when the error level is smaller than or equal to 5%. The frequency distance FDL is better when the error level is bigger than 5%.

The efficiency of all filtration algorithm quickly decreases when the error level increases. As mentioned before, if α is small, filtration algorithms can easily filter out the uninteresting references, and hence they finish in less than a millisecond. However, when α is large, the running time performance of our algorithm is better than that of most other filters.

Table 2. Comparison of filtration and time efficiency[a] on different error levels. In each entry η/t, where η denotes filtration efficiency and t denotes the running time of the algorithm per query in milliseconds.

Methods	0%	2%	5%	10%	20%
OnlyL	6%/1.92	3%/3.39	3%/19.27	3%/34.41	5%/63.72
FDL	40%/0.52	14%/2.44	9%/5.93	*7%/14.09*	**10%/39.58**
PML	73%/1.92	20%/8.30	4%/21.63	3%/35.73	5%/68.29
5GCL	98%/0.53	42%/2.00	8%/7.65	3%/29.44	5%/68.35
6GPML	*98%/0.61*	*56%/2.19*	*18%/6.33*	3%/34.44	5%/78.89
6GPMFDL	**98%/0.40**	**70%/1.25**	**34%/2.95**	**7%/14.66**	*8%/48.50*

[a] millisecond per query.

Table 3. The average detection sensitivities at different error levels on the simulated data.

Error level α	Average detection sensitivity S
0%	1.00
5%	1.00
10%	0.97
15%	0.96
20%	0.93
30%	0.90
40%	0.87

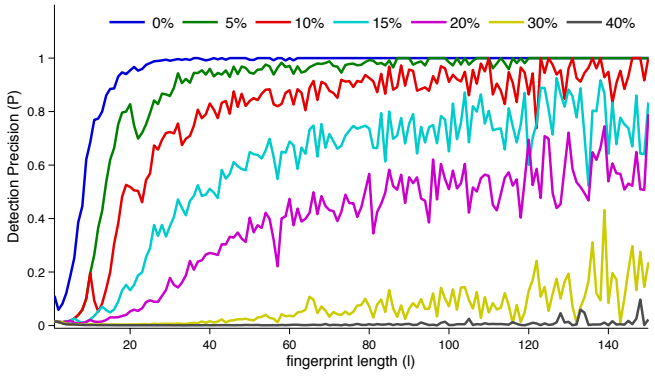


Fig. 4: Detection precision as a function of the fingerprint length for various error levels on the simulated data.

3.4 Proteins can be detected well at low error levels

3.4.1 CK fingerprints alone Here we examine the performance of using our algorithm to find matches based on fingerprints, considering various error levels, from error-free to 40%. We define *detection precision (P)* as one divided by the number of matches found by our algorithm, if the original query protein is retrieved; otherwise, it is zero. A unique match is found when the detection precision is one, i.e. only the same sequence as the given query sequence is retrieved. Similarly, the *unique match percentage (U)*, is defined as the percentage of queries yielding a unique match. If the detection precision is 1 for all queries, then the unique match percentage is maximized. Finally, *detection sensitivity (S)* is defined as one if the original query protein sequence is retrieved, i.e. when there is a true match; otherwise, it is zero.

In order to test the efficacy of our method, we simulated some fingerprints queries with properties similar to those expected from the single-molecule finger-printing technology. We used the human complete proteome database, and iteratively introduced up to 40% errors into each of the fingerprint. Up to 70% deletions, 20% insertions and 10% transpositions are introduced at random. Then the erroneous queries are searched in the reference database. Fig. 4 and Table 3 shows the performance on the simulated data with different error levels.

When the fingerprint length l ranges from 1 to 8 for $\alpha = 0\%$ (no errors), the average detection precision is smaller than 0.5 and never reaches 1, meaning we cannot find unique matches for fingerprints shorter than 9 symbols even in the error-free case. Thus, caused by the fact that there are too many short length fingerprints compared to the number of possible different fingerprints (2^l) we can distinguish. This is the main reason why the unique match percentage is less than 90% in Fig. 5. However, for some long sequences, such as l is 40, the detection precision is also smaller than 1, in this case, it is because K occurs 2.49 more often than C as we described before in Section 3.1.

If a higher error level is set for the simulation, more errors are introduced to the queries and more reference sequences will be retrieved for each query. Fig. 4 also shows the change of detection precision for various error levels. Clearly, it drops when the error level increases for each fingerprint length. Shorter sequences are strongly affected by the increase in error level. Table 3 shows whether we are able to find the true match using the simulated data. As error levels increase, we increasingly are unable to find the true match back, and so the detection sensitivities decrease. The reasons are described in detail in Supplementary Material Section 5.3.

As we already pointed out, we cannot find unique matches for sequences shorter than 9 even in the error-free case. In order to find out whether these small fingerprint proteins share a domain or a biological process, we did an annotation analysis on the sequences shorter than or equal to 12 (the precision for length between 9 and 12 is still very low), we found several annotations that are enriched in these sequences, such as *hormone activity*, *Ribonucleoprotein LSM domain*, and *S100/Calbindin-D9k, conserved site* (see Supplementary Material Section 5.5 for more detail). Sequences with these annotations are more likely to be non-specifically distinguished.

As illustrated in Fig. 5, for fingerprints alone, we can find a unique match for half of the queries when the error level is less than 10%. When the error level goes up to 30%, we are unable to find unique matches for any of the queries.

3.4.2 With additional spacing information Using the CK fingerprints alone, we are able to uniquely distinguish approximately 90% of the sequences in the error-free case. This percentage decreases when the error level goes up. We therefore examined the possibilities of including spacing information, both the occurrence of spacing and the actual spacing length.

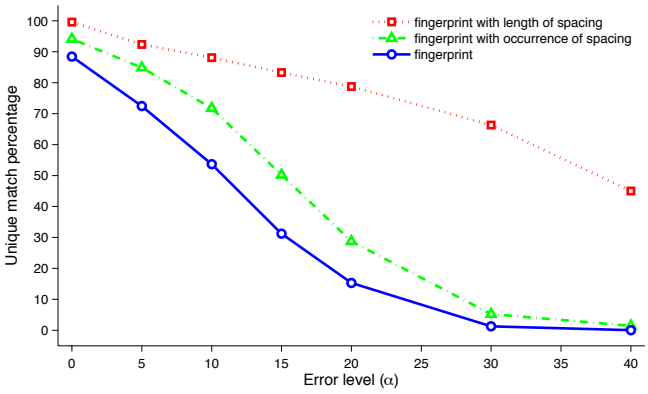


Fig. 5: Unique match percentage for various error levels on the simulated data.

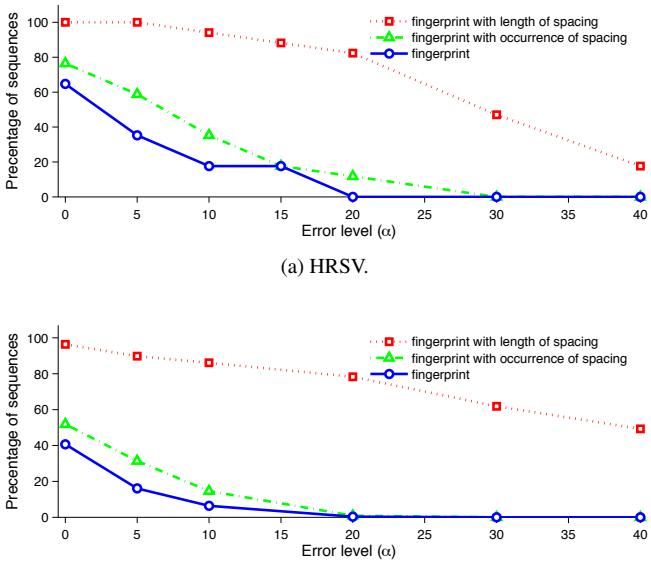
Fig. 5 compares the unique match percentages with and without spacing information. At each error level, the performance increases when we include spacing information. Even by using only the occurrence of spacing information, the unique match percentage increases for error level smaller than 30%. If the length of spacing is included as well, the unique match percentage increases dramatically. For the error-free case, it goes up to 99.6%, an improvement at about 10% compared to the original CK fingerprint performance.

In summary, using only the fingerprints, the performance becomes extremely poor for long fingerprints ($l \geq 30$) even when the error level is at only 15%. Shorter fingerprints, especially those ≤ 8, are very hard to detect at all. Including spacing information in some way improves performance, it is a necessity for shorter fingerprints. If we could measure a third amino acid, the detection precision will be improved even further (for detail, see Supplementary Material Section 5.4).

3.5 Single-molecule protein sequencing is useful in a variety of applications

Until now, we have explored the performance of single-molecule protein sequencing for identifying proteins in human samples. However, the system may have different uses. Here explore two possible applications: *1)* detection of bacteria/virus infection, and *2)* detection of cancer-related proteins.

3.5.1 Distinguish bacterial/viral proteins from human One possible application is to determine whether a prepared human cell sample contains bacterial or viral proteins using the fingerprint information. To examine this, we searched for proteins in this *human respiratory syncytial virus (HRSV)* and *Tuberculosis (TB)* protein datasets in the human proteome, and checked how many of these proteins are not retrieved, indicating they could be uniquely identified. We chose these two dataset causes for their potential clinical relevance. HRSV infection occur in children may lead to severe illness if diagnosed too late (Glezen *et al.*, 1986). Thus, we need to determine whether a child is infected with HRSV as quickly as possible. TB is a common and deadly infection;



(a) HRSV.

(b) TB.

Fig. 6: Detection performance *a)* HRSV and *b)* TB proteins in the human proteome at different error level thresholds.

it is definitively diagnosed by identifying *M. tuberculosis* using histopathology, which is a slow culture process taking 2 to 6 weeks (Pai *et al.*, 2008). As a matter of fact, treatment is often begun before a definitive diagnosis is confirmed (National Collaborating Centre for Chronic Conditions, 2006).

UniProt database contains 21 HRSV proteins (Magrane and Consortium, 2011), four of them have fingerprints shorter than 8 and one of them has a fingerprint length of 206. TB has many more proteins (6327), and 47.0% of which have fingerprint length of 8 or less. We searched each bacterial/viral protein in the human database using our algorithm. If the number of true matches is zero, it means the query protein is absent from the human database. The percentage of bacterial/viral proteins absent in the human proteome is computed and shown in Fig. 6.

We conclude that fingerprint can be used to determine whether a human protein sample contains bacterial/viral proteins when the error level is less than 20%, considering all proteins in a sample are measured. When the error level is higher than 20%, we find matching human protein sequences for all bacterial/viral proteins, which makes it impossible to use only fingerprints to detect them. As before, the allowed error level increases when spacing information is included.

3.5.2 Detecting cancer-related proteins Another use scenario for single-molecule protein sequencing is to diagnose cancer in human. By detecting or quantitating cancer-related proteins in the proteome. The cancer gene set is downloaded from COSMOC (Forbes *et al.*, 2011). The corresponding proteins are retrieved by cross-referencing the cancer gene set and the human proteome. 448 out of 522 unique cancer-related proteins are in the reviewed human complete proteome database obtained from UniProt (Magrane and Consortium, 2011). We used our algorithm to search these cancer

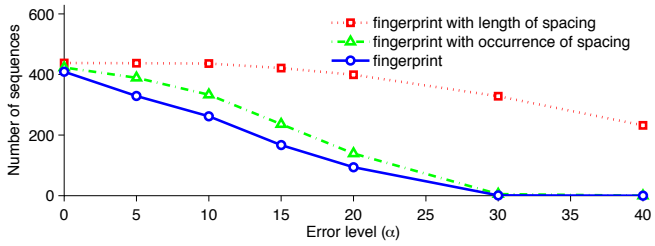


Fig. 7: Number of identifiable cancer proteins in human.

proteins in the human proteome. The number of uniquely retrieved proteins at various error levels is shown in Fig. 7.

Clearly, fingerprint is enough to detect cancer proteins in human sample at up to 30% of errors. When occurrence of spacing is included, performance is slightly better at the beginning, but no longer helps when the error level is higher than 30%. However, the use of the spacing length dramatically increases performance, even for $\alpha > 30\%$.

4 CONCLUSIONS AND FUTURE WORK

Here, we developed a first tool FPD to retrieve the actual protein sequences using the order of just two amino acids. Our algorithm employs a clustering method to extract sets of similar proteins and reduce redundancy of the fingerprints database, an efficient filtering strategy to identify potential matches and a dynamic programming to verify the matches found. These matches are then mapped back to the original fingerprint database to get the final proteins.

We demonstrated that by using the fingerprint information alone, we can accurately detect most of proteins, but it is hard to specifically identify with fingerprints less than 8 long. If errors are taken into account, performance quickly descends. Including additional information, such as a spacing or adding a third amino acid to the fingerprint, could improve performance. As a matter of fact, unless the length of spacing is measured or the error rate of the measurement is less than 15%, it's difficult to detect proteins with high precision using the single-molecule finger-printing technique. In addition, the fingerprint is sufficient to solve other problems, for example to detect bacterial/viral infections and to detect cancer-related proteins.

While the overall emphasis of the current paper was on sequence detection, we believe that several of our methods may have a broader application. For example, our clustering method can be used in other problems, which require to remove redundancy of the database, especially when the database consist of patterns extracted from an original database. The combined filtration strategy could also be useful for other string matching applications, where our method could be used to improve efficiency.

We have deferred several issues for future work. Currently, we do not have real data of the measurement, thus the performance analysis is done on the simulated data with properties similar to those expected from the single-molecule protein sequencing technology. We plan to run the algorithm with the real data in the future, and test using a better simulation when we learn more form the measurement.

Another worthwhile analysis is how much performance can be improved by including unknown phosphorylation sites of a protein. This may be useful, cause physically phosphorylation positions can be effectively labeled.

Although our clustering method outperforms two existing methods, it is slower. The main reason is the use of a slower dynamic programming pairwise sequence comparison. Since dynamic programming is also used in the verification, a faster method could be employed to future speed up our method without influence the accuracy. Implementing it through the use of SIMD instructions, which perform operations on multiple values in parallel (Wozniak, 1997; Rognes and Seeberg, 2000; Farrar, 2007), could be the solution.

REFERENCES

Adelson-Velskii, G. and Landis, E. M. (1962). An information organization algorithm. In *Doklady Akademia Nauk SSSR*, volume 146, pages 263–266.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**(3), 403–410.

Altschul, S. F., Madden, T. L., Schffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, **25**(17), 3389–3402.

Baker, T. A. and Sauer, R. T. (2012). ClpXP, an ATP-powered unfolding and protein-degradation machine. *Biochimica et Biophysica Acta (BBA)-Molecular Cell Research*, **1823**(1), 15–28.

Berg, J. M., Tymoczko, J. L., and Stryer, L. (2002). Amino acid sequences can be determined by automated edman degradation. In *Biochemistry, 5th edition*. WH Freeman.

Edman, P. and Begg, G. (1967). A protein sequenator. *European Journal of Biochemistry*, **1**(1), 80–91.

Farrar, M. (2007). Striped Smith-Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, **23**(2), 156–161.

Forbes, S. A., Bindal, N., Bamford, S., Cole, C., Kok, C. Y., Beare, D., Jia, M., Shepherd, R., Leung, K., Menzies, A., Teague, J. W., Campbell, P. J., Stratton, M. R., and Futreal, P. A. (2011). COSMIC: mining complete cancer genomes in the catalogue of somatic mutations in cancer. *Nucleic Acids Research*, **39**(suppl 1), D945–D950.

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

Glezen, W. P., Taber, L. H., Frank, A. L., and Kasel, J. A. (1986). Risk of primary infection and reinfection with respiratory syncytial virus. *American Journal of Diseases of Children*, **140**(6), 543–546.

Huang, D. W., Sherman, B. T., and Lempicki, R. A. (2009). Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Research*, **37**(1), 1–13.

Huang, Y., Niu, B., Gao, Y., Fu, L., and Li, W. (2010). CD-HIT suite: a web server for clustering and comparing biological sequences. *Bioinformatics*, **26**(5), 680–682.

Kahveci, T. and Singh, A. K. (2001). Efficient index structures for string databases. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 351–360, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kim, M.-S., Pinto, S. M., Getnet, D., Nirujogi, R. S., Manda, S. S., Chaerkady, R., Madugundu, A. K., Kelkar, D. S., Isserlin, R., Jain, S., *et al.* (2014). A draft map of the human proteome. *Nature*, **509**(7502), 575–581.

Lee, S., Lee, J., and Hohng, S. (2010). Single-molecule three-color FRET with both negligible spectral overlap and long observation time. *PLoS One*, **5**(8), e12270.

Magrane, M. and Consortium, U. (2011). UniProt Knowledgebase: a hub of integrated protein data. *Database : the journal of biological databases and curation*, **2011**, bar009.

Manavski, S. A. and Valle, G. (2008). CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. *BMC Bioinformatics*, **9**(Suppl 2), S10.

Miseta, A. and Csutora, P. (2000). Relationship between the occurrence of cysteine in proteins and the complexity of organisms. *Molecular Biology and Evolution*, **17**(8),

1232–1239.

National Collaborating Centre for Chronic Conditions (2006). *Tuberculosis: clinical diagnosis and management of tuberculosis, and measures for its prevention and control*. Royal College of Physicians (UK).

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys (CSUR)*, **33**(1), 31–88.

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

Pai, M., Ramsay, A., and O'brien, R. (2008). Evidence-based tuberculosis diagnosis. *PLoS Medicine*, **5**(7), e156.

Pearson, W. R. and Lipman, D. J. (1988). Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, **85**(8), 2444–2448.

Rognes, T. and Seeberg, E. (2000). Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics*, **16**(8), 699–706.

Sedgewick, R. (1978). Implementing quicksort programs. *Communications of the ACM*, **21**(10), 847–857.

Steen, H. and Mann, M. (2004). The ABC's (and XYZ's) of peptide sequencing. *Nature Reviews Molecular Cell Biology*, **5**(9), 699–711.

Stevens, B., Chen, C., Farrell, I., Zhang, H., Kaur, J., Broitman, S. L., Smilansky, Z., Cooperman, B. S., and Goldman, Y. E. (2012). FRET-based identification of mRNAs undergoing translation. *PloS one*, **7**(5), e38344.

Szalkowski, A., Ledergerber, C., Krähenbühl, P., and Dessimoz, C. (2008). SWPS3–fast multi-threaded vectorized Smith-Waterman for IBM Cell/BE and $\times$86/SSE2. *BMC Research Notes*, **1**(1), 107.

Thompson, J. D., Gibson, T. J., Plewniak, F., Jeanmougin, F., and Higgins, D. G. (1997). The CLUSTAL_X windows interface: Flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research*, **25**(24), 4876–4882.

Trinquier, G. and Sanejouand, Y. H. (1998). Which effective property of amino acids is best preserved by the genetic code? *Protein Engineering*, **11**(3), 153–169.

Ukkonen, E. (1992). Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, **92**(1), 191 – 211.

Wilhelm, M., Schlegl, J., Hahne, H., Gholami, A. M., Lieberenz, M., Savitski, M. M., Ziegler, E., Butzmann, L., Gessulat, S., Marx, H., *et al.* (2014). Mass-spectrometry-based draft of the human proteome. *Nature*, **509**(7502), 582–587.

Wozniak, A. (1997). Using video-oriented instructions to speed up sequence comparison. *Computer Applications in the Biosciences: CABIOS*, **13**(2), 145–150.

Wu, S. and Manber, U. (1992). Fast text searching: allowing errors. *Communications of the ACM*, **35**(10), 83–91.

SUPPLEMENTARY MATERIAL

1 SINGLE-MOLECULE PROTEIN SEQUENCING

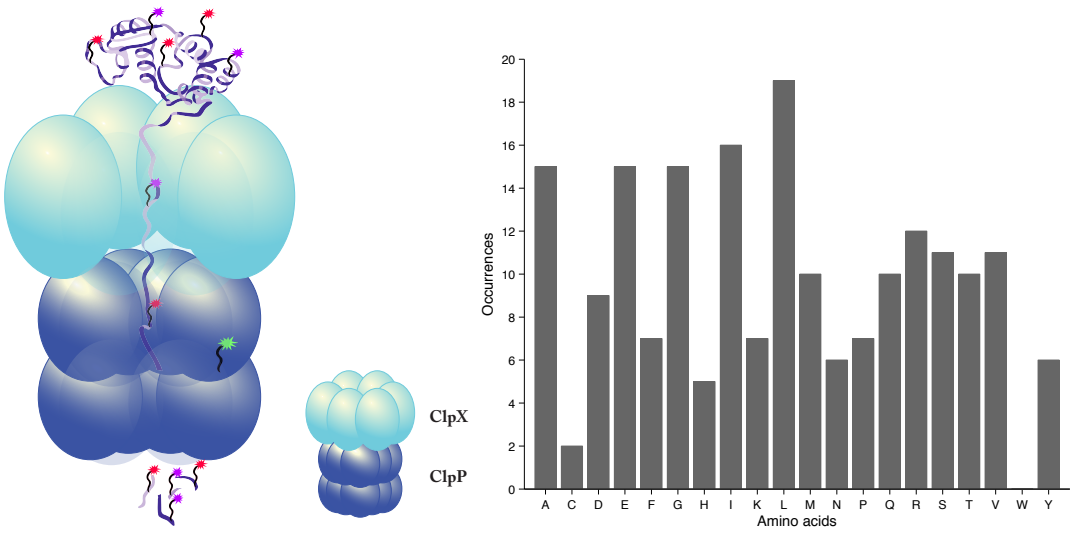
The single-molecule protein sequencing method is a laser-based technique that allows the fast sequencing of an individual protein molecule. It detects the fluorescent signal of labeled individual amino acids one by one. By using this technique, we can measure the sequence using only a single protein, particularly useful for sequencing proteins with a low expression level.

1.1 Existing protein sequencing methods

Right now, there are two main approaches to determine a protein's sequence without prior knowledge of the gene which translate to the protein: Edman degradation and mass spectrometry. Each of these has its limitations. *1)* Edman degradation, developed by Peer Edman, identifies a protein by removing amino acids one by one from the amino-terminus (Edman and Begg, 1967). However, if the peptide sequence is longer than 50 residues (Berg *et al.*, 2002) or the N-terminus is modified or hidden within the tertiary structure, this method will fail to identify the protein accurately and completely (Steen and Mann, 2004). *2)* Electrospray ionization mass spectrometry was developed in the late 1980s, allowing large molecules like polypeptides to be processed and measured. The protein is first degraded by an endopeptidase (an enzyme that breaks peptide bonds of nonterminal amino acids) to produce protein fragments before measuring the mass-to-charge ratios (Berg *et al.*, 2002). As a result, this method can analyze only protein fragments. This is problematic since we need to assemble those small fragments in order to reconstruct the original sequence. We cannot be 100% certain that the reconstructed sequence is accurate using techniques at the present time. And like Edman degradation, it requires a large and highly concentrated sample of a protein.

1.2 Single-molecule finger-printing approach

A large and pure protein sample is required to accurately and completely sequence a protein using the current sequencing techniques (in Supplementary Material Section 1.1). Single-molecule finger-printing approach allows us to sequence the entire protein using only a small amount of protein samples. This is required, because *1)* some proteins are found in extremely small amounts in nature. For example, a human cell may only have few copies of a transcription factor. *2)* Likewise, know that some diseases are due to the occurrence of specific proteins. If we could easily sequence proteins, we could diagnose diseases by identifying the particular protein associated with the disease. *3)* Protein deficiency diseases occur because of the lack of a specific protein. By sequencing protein samples from healthy individual, and reproducing these proteins *in vivo* or *in vitro*, they can be used in therapies to cure such deficiency diseases. Here, the single-molecule fluorescence resonance energy transfer (smFRET) measurements which lead to protein fingerprint is described in more detail (see Fig. 8 (a)).



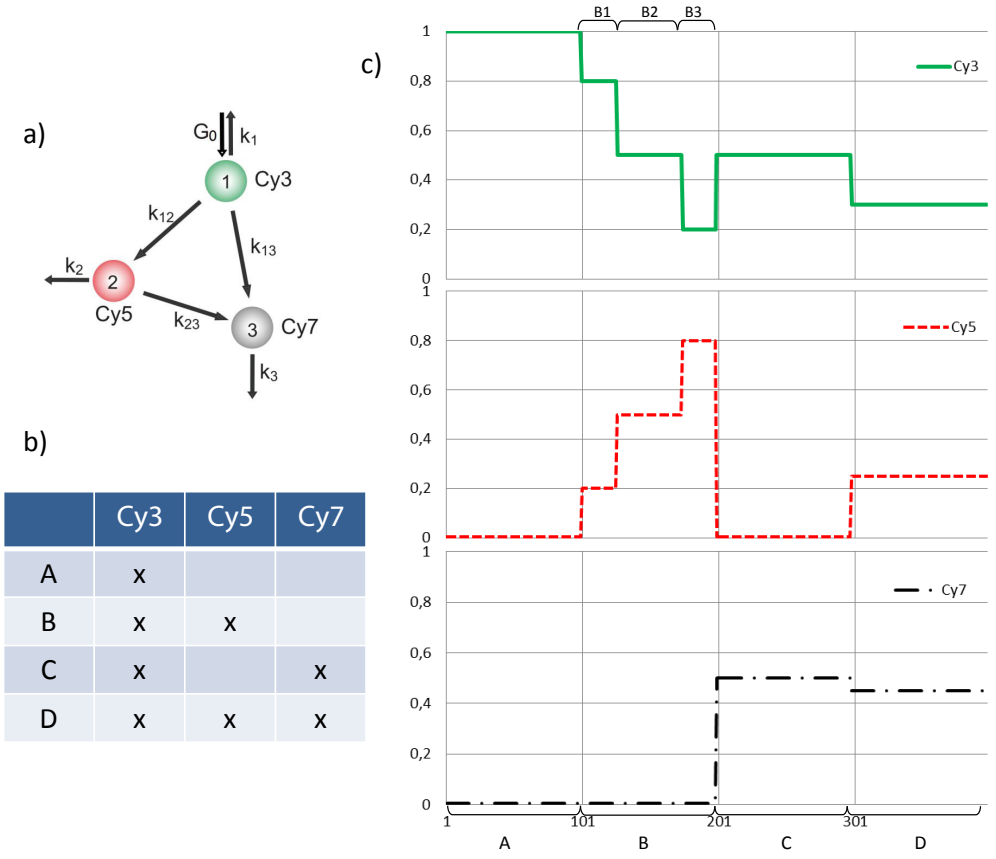
(a) ClpXP model.

(b) AA composition of ClpP.

Fig. 8: *a)* The model of how a protein goes through the ClpXP. The donor is placed at the top disk of the ClpP. The protein is labeled with two different dyes. *b)* Amino acid composition of the ClpP subunit. The length of a ClpP subunit is 207 amino acids.

First, the sequencing substrates need to be labeled with acceptors fluorophores. Labeling all the 20 different amino acids in a protein needs an equivalent number of dyes, which is rather expensive to achieve at the present time. It is also inefficient since not all of them can be labeled specifically. So labeling two amino acids, cysteine (C) and lysine (K), is used. The reason for choosing cysteine is that it is conserved in proteins and the only amino acid that has a thiol group (-SH). The occurrence of cysteine in various species ranges from 0.5% to 2.26% (Miseta and Csutora, 2000). Also, cysteine residues in homologous proteins are highly conserved according to the PAM 250 scoring matrix, which is used to score aligned protein sequences to determine the similarity. Lysine is rather common; the frequency of occurrence of lysine is 5.8% in the nonredundant OWL protein database (Trinquier and Sanejouand, 1998). Since maleimide dyes react with a thiol group and NHS (N-hydroxysuccinimide) ester dyes are commonly used for fluorescent labeling of amine groups (-NH_2) in proteins, thus, the thiol group of cysteine is labeled with maleimide dyes and the amine group of lysine is conjugated with NHS ester days.

The labeled protein sequence has to pass through an instrument that can unfold it and scan it accurately, in order to read the labeled C and K residues. ClpXP consists of two proteins, a ClpX and a peptidase called ClpP (Baker and Sauer, 2012), is chosen. ClpX unfolds the protein substrate before ClpP degrades the substrate into small peptide fragments. A donor dye (Cy3) needs to be placed in a specific location in ClpXP in order to react with labeled cysteine and lysine residues of a protein. We choose to label a cysteine in a ClpP subunit, because the maleimide dyes react with a thiol group of cysteines, and cysteines occur less than the other amino acids (except for tryptophan (W), which is not present in ClpP; see Fig. 8 (b) for the occurrence of 20 amino acids in a ClpP subunit). Since ClpP is symmetric as illustrated in Fig. 8 (a), the donor can be near the bottom of ClpP chamber or at the top. Here, we place the donor at the top disk of the ClpP.



	Cy3	Cy5	Cy7
A	x		
B	x	x	
C	x		x
D	x	x	x

Fig. 9: FRET signal between three dyes. *a)* Interaction map among three dyes (Lee *et al.*, 2010). Only Cy3 is directly excited by the light source G_0; k_i, decay rate of the i^{th} dye in the absence of FRET interaction and k_{ij}, the rate of FRET from the i^{th} dye to the j^{th} dye. *b)* Four different kind of occurrences among three dyes. In each situations, the excited dyes are marked with 'x's. *c)* The FRET signal without noise. The time period is divided into 4 parts. In situation A, Cy3 is excited by the light source and no acceptors are within the FRET distances. Intensities shown in part B and part C correspond to a Cy3-Cy5 pair and a Cy3-Cy7 pair respectively. Due to the interaction among the dyes, all three dyes have some intensity value in the last part. We assume the distance between dyes are constant in situation C and D, but in situation B, the distance between Cy3 and Cy5 is decreasing from B1 to B3.

After placing the donor in a feasible place, we can measure the FRET signal between the donor (Cy3) and the acceptors (Cy5 and Cy7) to distinguish cysteines and lysines. The sensitivity of FRET depends on the distance between donor and acceptor pairs, FRET is most sensitive when the distance between two dyes is about 5.4 nm for a Cy3-Cy5 pair and 3.8 nm for a Cy3-Cy7 pair (Lee *et al.*, 2010). Fig. 9 (a) shows the interaction diagram among the three dyes. From this diagram, we can deduce four different situations based on the occurrence of three dyes within the FRET working range and the ideal signal with respect to the four situations (see Fig. 9 (b) and (c)).

To acquire the fingerprint, we need to image the process using fluorescence microscopy. In order to measure individual proteins over an extended time period and to discard the background noise at the same time, we immobilize ClpXP proteins on a quartz slide and illuminate under total internal reflection fluorescence microscopy (TIRF). The ClpP is tagged with biotin conjugated with a neutravidin, the quartz slide is also marked with biotin. Since neutravidin is a tetramer with a strong affinity for biotin, the ClpXP will stick on the slide and be immobilized. The sequencing process will start when the proteins are introduced into the immobilized ClpXP. Image processing and quantitation will finally result in a fingerprint sequence.

2 CLUSTERING ANALYSIS

2.1 Evolutionarily related proteins are more likely have similar fingerprints

To verify our assumption that two evolutionarily related proteins are more likely to have similar fingerprints, we checked several different set of proteins which we know for sure that are evolutionarily related, each of the set consists sequences that have the same gene name or similar protein names. The following Table 4 shows the number of sequences of each set and the average pairwise amino acid and fingerprints similarities. It is clear that proteins with higher amino acid similarities also share a significantly higher fingerprint similarity. The average fingerprint similarity between all pairs of proteins in the database is 46.3%, which is at least higher than or equal to the average fingerprint similarity between pairs of non-related proteins, thus is the upper bound of the fingerprint similarity between not evolutionarily related proteins. Since the upper bound is much lower comparing to the listed fingerprint similarities for related proteins, thus it is true that two evolutionarily related proteins would have a high fingerprint similarity.

Table 4. Amino acids and fingerprints similarities of sets of proteins which share an evolutionary origin.

Gene/Protein name	No. of sequences	Amino acids similarity	fingerprint similarity
GN: HLA-A	21	93.9%	88.9%
GN: HLA-DRB1	13	93.2%	88.6%
GN: HLA-B	35	92.8%	87.5%
PN: Ankyrin repeat domain-containing protein 20...	4	97.1%	94.9%
PN: Arf-GAP with GTPase, ANK repeat and PH domain-containing protein...	11	72.3%	82.4%
PN: 14-3-3 protein...	7	75.9%	81.1%

2.2 Choice of threshold parameters of clustering method

Here, we determined which SPI threshold is the optimal choice to reduce the redundancy of the fingerprints. The protein sequences are clustered based on their amino acid sequences and analyzed on their fingerprints. Running on the whole database is too slow, so we tested on a smaller database containing 5000 random sequences (hm5000).

Because although there are many individual exceptions, it is believed that when two proteins share 50% or higher sequence identity, their backbones differ by less than 1 $\mathring{A}$ root-mean-square (RMS) deviation (Gerstein and Levitt, 1998). Thus the protein sequences share at least 50% of similarities are considered as evolutionary-related proteins and works as the T_{low} in our method. We compared the within and between class fingerprints similarities, and also checked the number of clusters, number of sequences and uniqueness after clustering to choice a better maximum SPI threshold T_{high}. We can see from Table 5 that when T_{high} decreases, the number of clusters drops as well as the number of sequences. In the mean time, the after clustering uniquenesses are always bigger than before clustering and increase while T_{high} decreases. For the fingerprints, the within class similarity is always smaller than the corresponding maximum SPI threshold T_{high}. When T_{high} drops, the within class similarity decreases and the between class similarity increases. Thus we choose T_{high} is 90% by

considering the influences of within class similarity, between class similarity as well as the after clustering uniqueness. Note that since all different fingerprints are kept as representatives, the loss ratio of our method is always zero.

Table 5. The performance of the clustering method for different SPI thresholds. The between class similarities are analyzed on all clustered members, and not only on representative sequence.

T_{low}-T_{high}	Uniqueness before	No. of clusters	No. of sequences	Within class CK similarity	Between class CK similarity	Uniqueness after
50% - 95%		4851	4919	92.5%	37.161%	91.3%
50% - 90%	89.1%	4804	4915	84.6%	37.158%	91.4%
50% - 85%		4772	4914	81.1%	37.155%	91.4%

2.3 Representatives of the four clustering methods

To known which representatives are chosen by the clustering method, we show a Venn diagram in Fig. 10 to illustrate the finding that the sets of representatives obtained by different programs fed with the same input database and maximum SPI threshold have a considerable degree of similarity in size and content. It is also clear that using our criteria on choosing the representatives will keep more sequences which should not be considered as redundancy because they are distinct in terms of fingerprints.

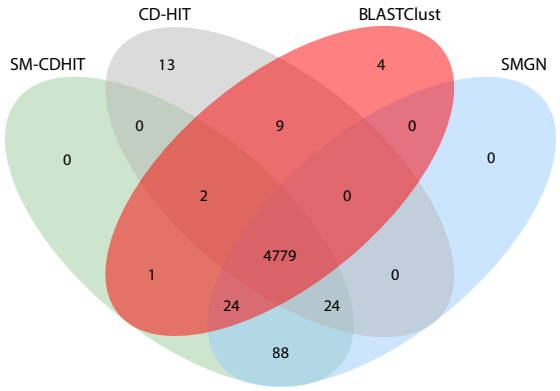


Fig. 10: Venn diagram. The overlap of four different databases after clustering, each obtained with a different program, based on the same input database (hm5000) and same maximum SPI threshold (90%).

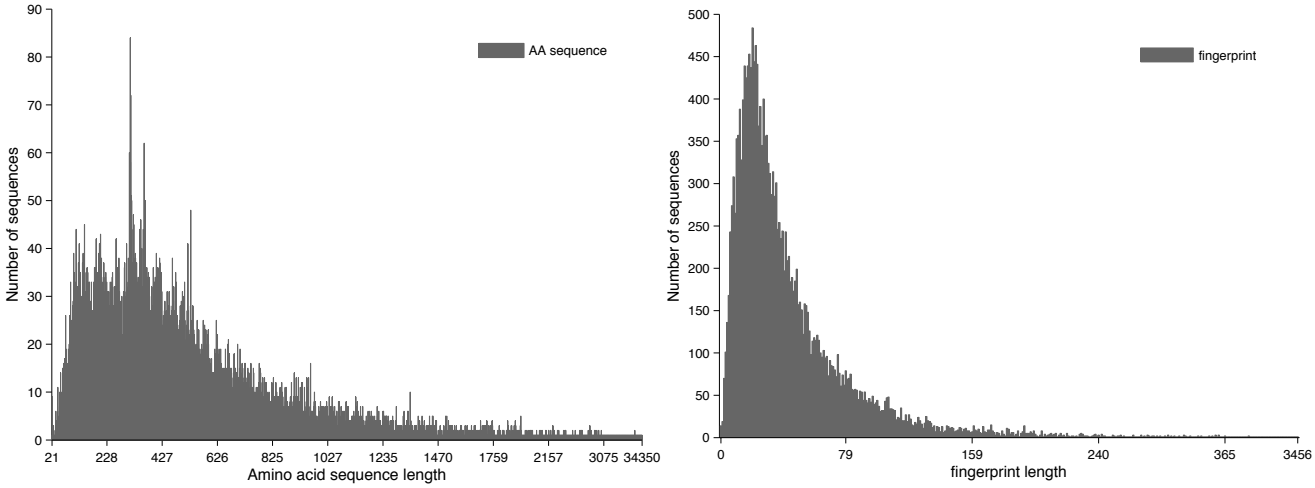
2.4 Using a single pairwise sequence distance SD threshold instead of a linear SPI threshold model to control the similarity

The amino acid length distribution of human proteome ranges from 21 to 34350 (Fig. 11) (a). The average length is 559.02 amino acids. As we have shown previously, a very high similarity between two amino acid sequences results in a poor clustering, since it fails to put related proteins together. On the contrary, a lower similarity might put non-related proteins in a cluster. Hence, it is impossible to set a single SD bound works well for very short amino acid sequence and long ones at the same time because of the extremely wide range of the amino acid sequence length. Thus, a single SD bound does not satisfactory for our purpose.

3 FILTRATION METHODS AND ANALYSIS

3.1 Filtration based on length infromation

Given query Q, reference R_x and error level α, R_x is a potential match when the absolute difference between the length is smaller than or equal to k, where k is $\alpha \times L_Q$.



(a) Amino acid length.

(b) Fingerprint length.

Fig. 11: The *a)* amino acid length and *b)* fingerprint distribution of human complete proteome.

An Adelson-Velskii and Landis' tree (AVL) is a height-balanced binary search tree (Adelson-Velskii and Landis, 1962). Since the database is large and searched frequently, to quickly locate data without having to search through the entire database we build an AVL tree structure on the lengths of fingerprints in the database. The process consists of two steps: a sorting step and building step. In the sorting step, a quicksort algorithm (Sedgewick, 1978) is used to order the entries in the database by their fingerprint lengths. In the building step, an AVL tree is constructed on the fingerprint lengths, where the node represents fingerprint length and the positions of the first and last entries of that length in the database are stored for each node. Afterwards, we can quickly locate potential matches by searching the smallest and largest potential length in the AVL tree.

Building a tree structure on the sequence length is better than using a lookup table, since it saves more memory based on the observation of the fingerprint length distribution in Fig. 11 (b). The fingerprint length ranges from 0 to 3456, but only 366 of these numbers are taken by at least one sequences in the database. If we would use a lookup table, only 10.6% of them would be used to save the locations. Hence we build a balanced search tree on the length which uses less spaces.

3.2 Filtration based on frequency distance

The idea of *frequency distance* is that if two sequences are similar, then the frequency, i.e. the number of occurrence of the alphabet symbols in two sequences should also be similar (Kahveci and Singh, 2001).

Given two fingerprints Q and R_x over an alphabet Σ. The *frequency vector* $f(x)$ is defined as $f(x) = f(x^1)...f(x^i)...f(x^\sigma)$, where $f(x^j)$ is the count of the jth symbol of Σ in sequence x. The frequency distance, $D_{fd}(R_x, Q)$, between reference R_x and query Q is defined as the minimum number differences between R_x and Q. Let $posDistance$ denotes the number of extra symbols for Q comparing to R_x, and $negDistance$ the number of missing symbols. Since each possible operation (deletion, insertion, substitution and transposition) changes the value of $posDistance$ and $negDistance$ by at most one, the larger one of the two values equals to $D_{fd}(R_x, Q)$. Formally,

$$\begin{aligned} D_{fd}(R,Q) &= max(posDistance, negDistance), \\ \text{where} \qquad posDistance &= \sum_{f(Q^j)>f(R_x^j)} f(Q^j) - f(R_x^j) \\ \text{and} \qquad negDistance &= \sum_{f(Q^j)<f(R_x^j)} f(R_x^j) - f(Q^j). \end{aligned} \tag{7}$$

The frequency distance between two fingerprint sequences is a lower bound on their distance $D(R_x, Q)$ (Kahveci and Singh, 2001). Thus, if the frequency distance $D_{fd}(R_x, Q)$ is larger than k, $D(R_x, Q) > k$, hence the reference sequence R_x can be filtered out. Otherwise, R_x

becomes a potential match and is kept for verification. Note that for our fingerprint sequences, the $posDistance$ equals to $negDistance$, since the alphabet symbol set Σ is {C, K}.

3.3 Partial exact matching

The central idea of this partial exact matching approach is to search some substrings of a query sequence which appear exactly in a reference (Wu and Manber, 1992). Suppose that we have a query Q and we want to find a reference sequence R_x in the database with at most k errors. Then for reasonably small k, there must exist a substring in Q which exactly appears in R_x. This approach therefore performs an exact string matching first.

Different algorithms select different subsequences in the query to do the exact string matching. The original approach cuts the query sequence into ρ pieces, where ρ is $(k+1)$. In this way, at least one of the pieces must appear exactly in a reference, since k errors cannot alter $(k+1)$ pieces at the same time (Wu and Manber, 1992). Hence, the first step for this algorithm is to divide Q into $(k+1)$ non-overlapping pieces $q^0, q^1, ..., q^k$ that have approximately equal length. Then, for each reference R_x, it checks whether any of the divided pieces in Q appear in R_x exactly. If none of the pieces appear exactly, then the checking step ignores this reference R_x. Otherwise, R_x is marked as a potential match.

Let k denotes the number of errors and L_R^x the length of the reference sequence R_x. This method needs to compare $(\rho^2 \frac{L_R^x}{L_Q})$ pairs of subsequences to determine whether there is a piece of Q appears exactly in R_x. Thus, it is time-consuming. In our implementation, we take advantage of the binary-to-decimal hashing function described above, and choose different subsequences based on the query length and k to quickly determine whether a reference is a potential match or not. Since the hashing method is slower for long fingerprints, instead of dividing the query into $(k+1)$ pieces, we iteratively find ρ for the query sequence. Each iteration consists of examining whether the length of each piece is smaller than a certain value and doubling ρ if at least one piece is longer. It repeats until all pieces are shorter than the value. In practice, we set this value to 14, because the hashing function works less effectively if the length of the (sub)sequence is larger than 14. Since we divide the query into ρ pieces, with $\rho \geq (k+1)$, the number of pieces that must appear exactly in a reference is $(\rho - k)$. If the number of exactly matching pieces is larger than $(\rho - k)$, then the reference is considered as a potential match. Otherwise, the reference is discarded by the filtration method.

3.4 q-gram counting filtering

A *q-gram* (Ukkonen, 1992) in the alphabet set Σ is any string in Σ^q, where Σ^q is the set of all possible strings of length q over Σ. In our case Σ is {C, K}. For $q = 1$, the sequence $CCKKKCC$ breaks down into 4 Cs and 3 Ks and sequence $CKCKK$ has 2 Cs and 3 Ks. The q-gram counting approach uses the q-grams of two sequences for filtration, and it takes the order of CK into account. For example, the above two sequences share the same number of Ks but the difference between the number of Cs in two sequences is 2. Thus, we must need at least two operations to make the numbers of Cs and Ks equal, for example, by deleting two Cs in the first sequence.

Given two sequences, reference R_x and query Q, we define the distance between two sequences as the total number of differences of the occurrences of q-grams as follows:

$$D_q(R_x, Q) = \sum_{\nu \in \Sigma^q} |G(R_x)[\nu] - G(Q)[\nu]|, \tag{8}$$

where ν is a q-gram and $G(R_x)[\nu]$ and $G(Q)[\nu]$ denote the total number of times ν occurs in R_x and Q, respectively. The relation between q-gram distance and the number of errors is given in (Ukkonen, 1992). Formally,

$$\frac{D_q(R_x, Q)}{2q} \leq D(R_x, Q). \tag{9}$$

It indicates that the q-gram distance divided by 2 times of q is a lower bound of the number of errors, hence can be used as a filter. When $\frac{D_q(R_x,Q)}{2q} \geq k$, the number of errors between Q and R_x is also larger than k, and therefore the reference sequence R_x can be filtered out. On the contrary, if $\frac{D_q(R_x,Q)}{2q} \leq k$, then it is possible that $D(R_x, Q) \leq k$, and R_x is marked as a potential match.

In our implementation, we took advantage of the binary-to-decimal hashing function to efficiently count the occurrences of each q-gram.

The choice of q should be based on the filtration efficiency and the time needed to process it. In the error-free case, the performance of various q is listed in Table 6. It is clear that when q increases, the filtration efficiency increases at first, and then decreases at some point. In terms of speed, the running time decreases at first, then increases when q is larger than 5. Considering the influences of filtration efficiency and time needed to perform this method, we choose q to be 5.

This filtration method sums the differences of all q-grams between two fingerprint sequences. When $q = 1$, this method counted the absolute differences between occurrence of symbols which is different from frequency distance method, which is the maximum of these differences.

Table 6. Comparison of performances using different q for q-gram counting filtration method. Note that tests were performed on the human complete proteome database.

q	filtration efficiency η	time t (ms)
1	0.242	2.33
2	0.598	2.07
3	0.866	1.88
4	0.937	1.98
5	0.958	2.02
6	0.963	2.55
7	0.958	2.55
8	0.948	2.91
9	0.935	3.56
10	0.924	4.59

3.5 The choice of q for q-grams partial exact matching

The choice of q should be based on the filtration efficiency and the time need to process it. In the error-free case, the performance of various q is listed in Table 7. When q is 6, we will be able to obtain the highest filtration efficiency without compromising to the speed.

Table 7. Comparison of performances of different q for q-gram partial exact matching filtration method. Note that it is run on the human complete proteome database.

q	filtration efficiency η	time t (ms)
1	0.237	1.74
2	0.592	1.34
3	0.862	1.33
4	0.937	1.60
5	0.958	1.80
6	0.963	2.15
7	0.958	2.18
8	0.948	2.68

3.6 Examples of filtration methods

To help better understand those different filtration methods, here we show examples of applying different filtration methods: length filtering, frequency distance, q-gram counting, partial exact matching, and q-gram partial exact matching. Let the query $Q\ CCKKK$ be compared to a reference $R_x\ CKKKKK$ with an error level of 20%. k equals to 1, since $k = \alpha \times L_Q$.

1. Using the length filtering, we check whether the length difference $|L_R^x - L_Q| \leq k$. If the length difference satisfies the criterion, R_x is a potential match. For this example, L_R^x is 6 and L_Q is 5, thus R_x will not be discarded by the length filtering.
2. The frequency distance $D_{fd}(R_x, Q)$ is 2, which is the maximum of $posDistance$ and $negDistance$, where $posDistance = 1$ and $negDistance = 2$. R_x will be filtered because $D_{fd}(R_x, Q) > k$.
3. By using a 2-gram counting filtration method, the reference R_x will be kept, since $D_q(R_x, Q) = 3$, hence $\frac{D_q(R_x,Q)}{2q} = \frac{3}{4}$, smaller than k.

4. Partial exact matching method first divides the query fingerprints into 2 pieces, thus query pieces $q^i = CC$, $q^2 = KKK$. Since $q^2 = KKK$ appears exactly in R_x, R_x will not be discarded.
5. Using a 2-gram partial exact matching method, the query fingerprint is divided into 2 pieces, $q^1 = CC, q^2 = KKK$, the corresponding pieces in reference are $r_x^1 = CKK$ and $r_x^2 = KKKKK$. Since $D_{qpm}(r_x^2, q^2) = 0$, meaning q^2 appears exactly in r_x^2, R_x will not be discarded.

For this particular example, only frequency distance can eliminate the uninteresting reference R_x. Normally, q-gram partial exact matching works better, follows by q-gram counting, partial exact matching, frequency distance and length filtering (shown in Table 2).

4 VERIFICATION ANALYSIS

4.1 The set of operation possibilities

Based on the analysis of the placement of donor dye, we concluded that four different errors can occur in the measurements: substitution, insertion, deletion, and transposition (swapping).

1. Insertions and Deletions are operations that change the length of the fingerprint. Intuitively, deletion removes a symbol and insertion adds a symbol. Deletion in one sequence is equivalent to insertion in the other. In our case, all operations are done in query, since references are verified by UniProt (Magrane and Consortium, 2011). For notational convenience, we denote inserted symbols with a special gap character "-", understanding that "-" corresponds to either a deletion or an insertion of a specific alphabet symbol. By our analysis, because of the low labelling efficiency and the weaker of the sequencing signal (leading to dark bases which are indistinguishable from the sequencing signal), a large number of amino acids would not be read out during the sequencing process, leading to deletion errors in query sequence. Moreover, since dyes are specifically conjugated to a certain group of an amino acid, few other amino acids will be incorrectly labeled, meaning few insertion errors in query sequence.
2. Substitution occurs when a symbol is replaced by another symbol at the same position in the sequence (mismatching). It is also known as the mutation or replacement operator. Since it is not likely a C or K wrongfully labeled with the other dye, mismatching errors are unlikely to occur in the measurements.
3. Transposition is an operation that swaps the position of two sequential symbols in a sequence. We restrict ourselves to only transposing sequential letters and allowing letters to be transposed only once. When the two labeled residues are close enough, it may be difficult to distinguish the order of these residues from the sequencing signal. Thus, it is possible to have transposition errors (see Fig. 12 for how often it might occur for various fingerprint length).

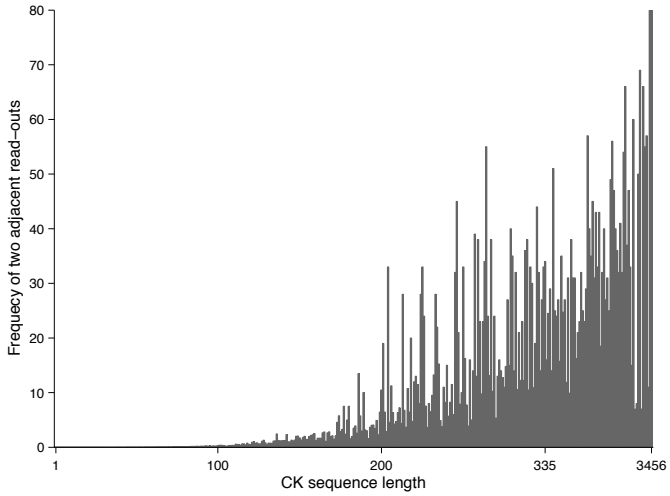


Fig. 12: The frequency of two read-outs that are adjacent to each other for various fingerprint length.

4.2 Costs for the four operations

As we discussed above, four types of errors might occur in our measurements. Thus, in the dynamic programing algorithm, these four operations are allowed. Note that a match of symbols between two sequences can be seen as a substitution operation, in which a symbol is "replaced" by itself. The number of matching positions between two sequences is of importance to determine whether a fingerprint is a match or not. Let c denotes the cost of a possible operations, each cost is described in detail below and the penalties set for each operation is shown in Table 8.

1. Deletion (adding a gap in query): It costs c_{del} to add a gap character after $Q[i]$ to match $R_x[j]$. The rest of the alignment score comes from aligning $Q[1...i]$ and $R_x[1...j-1]$. Therefore the maximum score ending with a deletion is $D_{i,j-1} + c_{del}$, where c_{del} is the cost of a deletion in the query sequence.
2. Insertion (adding a gap in reference): This is equivalent to adding a gap character after $R_x[j]$ to match $Q[i]$, which costs c_{ins}. The rest of the alignment score comes from aligning $Q[1...i-1]$ and $R_x[1...j]$. Hence the maximum score ending with an insertion is $D_{i-1,j} + c_{ins}$.
3. Substitution: It costs c_{sub} to substitute $Q[i]$ for $R_x[j]$. The rest of the alignment cost originates from aligning $Q[1...i-1]$ and $R_x[1...j-1]$. Therefore the maximum score ending with a substitution is $D_{i-1,j-1} + c_{sub}$. When $Q[i] = R_x[j]$ (a match), c_{sub} is positive, otherwise it is negative.
4. Transposition: We swap $Q[i-1]$ with $Q[i]$ and then substitute them for $R_x[j-1]$ and $R_x[j]$ respectively. These three operations cost c_{swap}. The rest of the alignment score comes from aligning $Q[1...i-2]$ and $R_x[1...j-2]$, whose maximum value is $D_{i-2,j-2}$. Therefore, the maximum score ending with a swap is $D_{i-2,j-2} + c_{trans}$.

Table 8. The penalties used in verification phase.

	' C '	' K '	' X '	' os '	' ls '
' C '	50				
' K '	$-50/-45$	50			
' X '	$-50/-45$	$-50/-45$	50		
' os '	$-8/-30$	$-8/-30$	$-8/-30$	20	
' ls '	$-1/-20$	$-1/-20$	$-1/-20$	$\times$	2

Let 'os' represent an occurrence of spacings, and 'ls' represent a spacing. 'X' represent a third amino acid. a/b in some cells, where a is the substitution penalty and b is the transposition penalty. The costs for deletion and insertion are $c_{del} = -2$ and $c_{ins} = -5$, respectively.

4.3 Dynamic programming example

Comparing two sequences based on the CK fingerprints alone and those with spacing information. Table 9 shows the inputs, and Fig. 13 shows the resulting pairwise alignments. It shows that if only fingerprints are used, the reference sequence is recognized as a true match. However, if spacing information is added, the reference sequence is discarded since the distance between them does not satisfy the criteria $(\alpha \times L_Q^{'})$ although $\alpha \times L_Q$ is satisfied, where $L_Q^{'}$ is the length of fingerprint with spacing information and L_Q is the length of CK fingerprint alone.

Table 9. The inputs of a example detection. Error level α is 30%

	fingerprint	fingerprint with occurrence of spacing	fingerprint with length of spacing
Query:	KCCKK	KoCoCoKK	KooooooCooooCKK
Reference:	KCCKKCK	KoCoCoKKoCoKo	KooooooCooooCKKoooCoooK

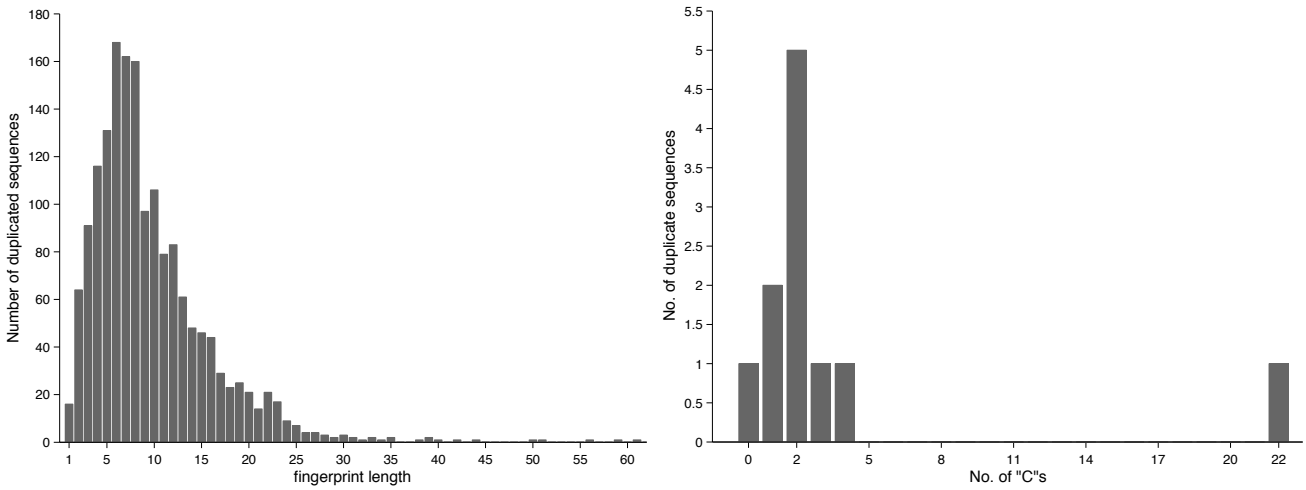
```
KCCKKCK   7
|||||
|||||
KCCKK--   7

  Matches: 5, Mismatches: 0, Transpositions: 0, Gaps in Query: 2, Gaps in Reference: 0, Total error: 2
==========================================================================


KoCoCoKKoCoKo   13
||||||||
||||||||
KoCoCoKK-----   13

  Matches: 8, Mismatches: 0, Transpositions: 0, Gaps in Query: 5, Gaps in Reference: 0, Total error: 5, Total CK error: 2
==========================================================================


K-ooooooCoooooCKKoooCooooK   26
| ||||||||||||||
| ||||||||||||||
KooooooooCooooCKK---------   26

  Matches: 16, Mismatches: 0, Transpositions: 0, Gaps in Query: 9, Gaps in Reference: 1, Total error: 10, Total CK error: 2
==========================================================================
```

Fig. 13: An example result of the verification.

5 RESULTS ANALYSIS

5.1 Fingerprint duplication

A non-redundant fingerprint database is one where each single entry isn't repeated. The $uniqueness$ is one and the $loss\_ratio$ is zero for a non-redundant database. This database is constructed to calculate the number of duplicates. For each entry in the non-redundant database, the number of exact matching fingerprints in the original database is extracted. Then the number of duplicates for a particular entry is defined as the number of exact matching fingerprints minus one.



(a) Various fingerprint length.

(b) Fingerprint length of 22.

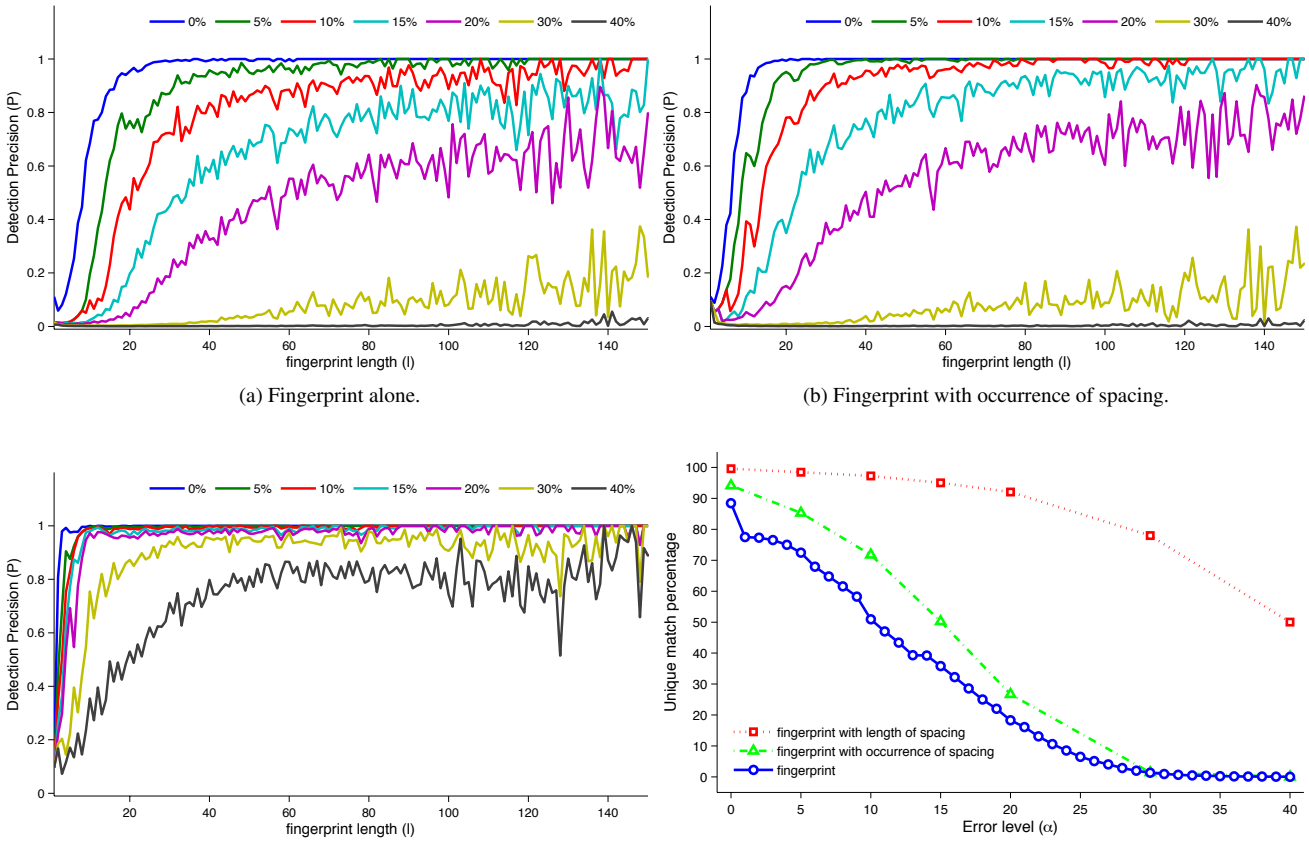
Fig. 14: Number of duplicates of *a)* various fingerprint lengths and *b)* fingerprint length 22.

From Fig. 14 (a), we can see that duplication not only occurs for short fingerprints, but even when the fingerprint length is relatively long. For example, we examined for fingerprints of length 22 (Fig. 14 (b)). It is clear that if a fingerprint consists of all most all Ks or Cs, it is more likely to have duplicate sequences.

5.2 Error-free query sequences

We consider the queries are error-free in this analysis, thus the true matches are retrieve every time, meaning the detection sensitivity is optimal. Detection precision for various fingerprint lengths are shown in Fig. 15 (a)-(c). It shows that when fingerprint length increases, the detection precision increases; and the detection precision drops when the error level increases for each fingerprint length. When more spacing information is included, the higher the detection precision for all fingerprint length. Although only a slight improvement can be seen when occurrence of spacing is included comparing to use the fingerprint alone. When the length of spacing is included, the improvement is significant.

As illustrated in Fig. 15 (d), the unique match percentage decreases while the error level increases. We can find a unique match for half of the queries when the error level is less than 10%. When the error level goes up to 30%, we will not be able to find unique matches for most of the queries by using the fingerprint alone. At each error level, the performance increases when we include more spacing information. Even by using fingerprint whith the occurrence of spacing, the unique match percentage increases while the error level is smaller than 30%. If the length of spacing is included as well, the unique match percentage increases drastically; for the error-free case, it goes up to 99.6%, an improvement at about 10% compared to use fingerprint alone.



(a) Fingerprint alone.

(b) Fingerprint with occurrence of spacing.

(c) Fingerprint with length of spacing.

(d) Unique match percentage.

Fig. 15: Detection precision as a function of the fingerprint length and the unique match percentage for various error levels.

5.3 Performance on simulated data

The detection precision on the simulated data is already shown earlier. Here, we examine the detection sensitivity when spacing information is included. The average of the detection sensitivities for three different measurements are shown in Table 10. When error level increases, the detection sensitivity decreases. Given an error level, the average detection sensitivity drops when more spacing information is included. The

reasons of why the detection sensitivity drops are: *1)* the dynamic programing algorithm favors deletions and insertions over substitutions and transpositions, where the latter two are considered as two deletions and/or insertions. Thus, the number of errors between them are bigger, which leads to misidentification. Another reason could be *2)* the length of the true match falls outside of the search range $(1-\alpha) \times L_Q \leq l \leq (1+\alpha) \times L_Q$. It might happen when too many deletions or insertions are simulated.

Table 10. The average detection sensitivities of different error levels on simulated data.

Error level α	Fingerprint alone	Fingerprint with occurrence of spacing	Fingerprint with length of spacing
0%	1.00	1.00	1.00
5%	1.00	0.95	0.91
10%	0.97	0.92	0.90
15%	0.96	0.91	0.87
20%	0.93	0.89	0.84
30%	0.90	0.86	0.81
40%	0.87	0.85	0.80

5.4 A third amino acid

Even though we can improve the performance by using spacing information, still some of the sequences are hard to uniquely identify. If we can label a third type of amino acid, we may be able to improve the detection precision especially for very short fingerprints, such as of length one and zero. Here, we analyse how much detection precision improves when include a third amino acid.

Fig. 16 presents the unique match percentage changes when a third amino acid is measured for the error-free case. All of them are significantly higher than using only fingerprints and when occurrence of spacing is included. Compared to using fingerprint with length of spacing, several third amino acids have a higher unique match percentage. These are among the amino acids that occur more frequently in the database. We can acquire the highest unique match percentage (99.9%) for the error-free case among the possible choices of a third amino acid, if the third labeled amino acid is serine (S). And the lowest unique match percentage we can get is 97.3% when tryptophan (W) is used as the third amino acid.

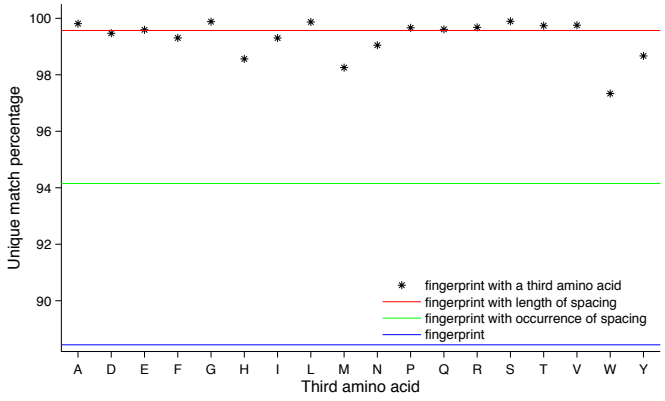


Fig. 16: Unique match percentages for possible third amino acids in error-free case. The red, green and blue horizontal lines correspond to the unique match percentage in error-free case using fingerprint with length of spacing, fingerprint with occurrence of spacing and fingerprint alone.

5.5 Annotation analysis for short fingerprints $L \leq 12$

We found that the detection precision is worse when the fingerprints are short. Thus, we more detailed examined the distribution of detection precision for various fingerprint lengths. From Fig. 17, it is clear that when the fingerprints have lengths up to 8, most of the detection precisions are smaller than one. For fingerprints with length range from 9 to 12, the total number of cases where detection precisions are smaller than one is larger than or equal to the number of cases where detection precisions equal to one. When fingerprint length is larger than 12, most of the cases have detection precisions equal to one, and few of them with detection precisions smaller than one. Thus, we considered fingerprints with length shorter than or equal to 12 as short fingerprints.

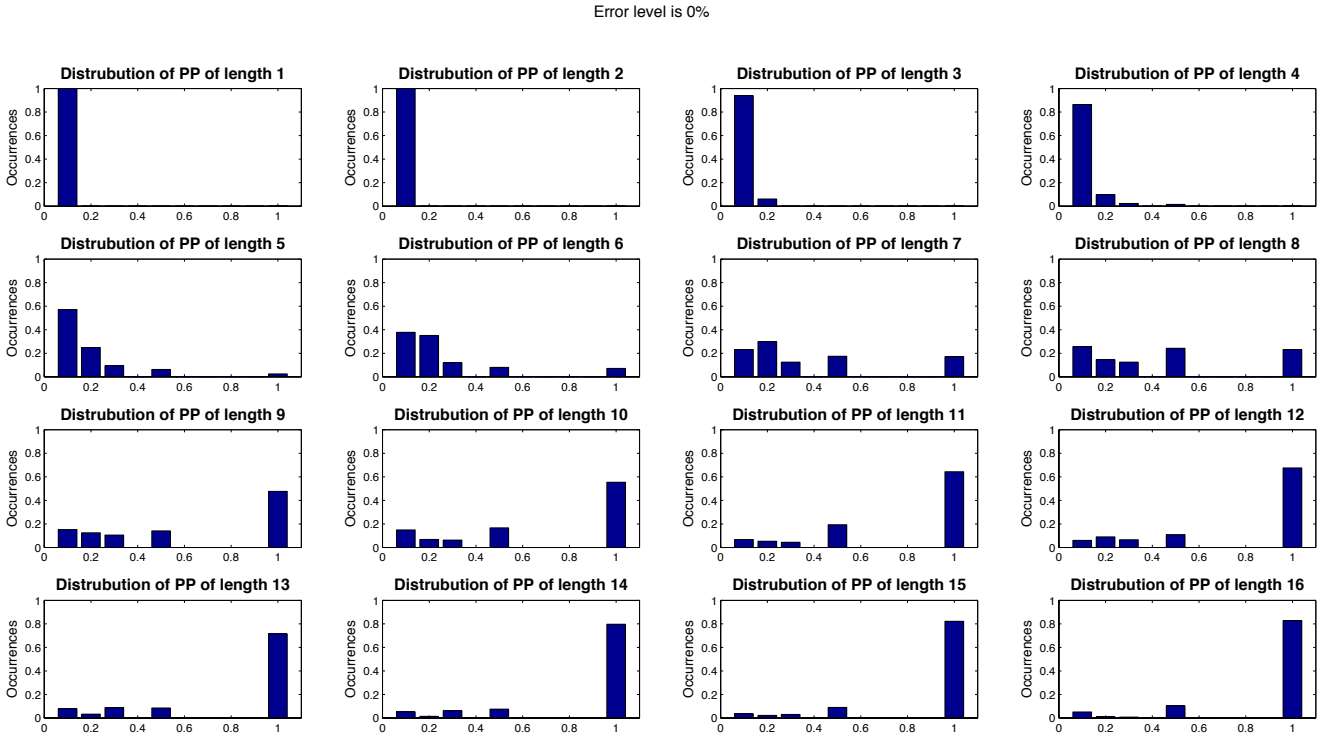


Fig. 17: The detection precision distribution for fingerprint length range from 1 to 16.

Duplicated sequences are not distinguishable by our algorithm. This may not be a problem in application when searching for proteins share the same function. We therefore asked whether such proteins share some other properties, such as protein domain and/or gene ontology. We performed an annotation analysis using DAVID (Huang *et al.*, 2009) on the fingerprints of length shorter than or equal to 12, and retrieved a list of annotations that are enriched in these sequences. We set the searching criteria to occur more than 3 times and have an EASE score $\leq$ 0.05 in two different annotation categories, gene ontology and protein domain. A term with an EASE score ≤ 0.05 usually is considered as strongly enriched (Huang *et al.*, 2009). Some shorter sequences, even though they have distinct amino acid sequences, share the same gene ontology annotations and/or protein domains.

Here we only shown several different annotations enriched in all lengths ≤ 12 and plotted the fingerprint length distribution of proteins having the corresponding annotation in Fig. 18. For each annotation, the fingerprint length distribution of proteins with the given annotation is overlaid on the fingerprint length distribution of the human proteome database. We specified the bin size to 12, so the first bin contains a length range between 1 to 12 and so on. Annotations, such as *Ribonucleoprotein LSM domain* and *Ribonucleoprotein LSM domain, eukaryotic/archaea-type* are restricted to short fingerprint lengths, although the other annotations are distributed over a larger range of fingerprint length. For example, the proteins with *Calcium-binding EF-hand* or *organelle envelope* annotations distributed over all human complete proteome lengths. Thus, although we retrieved these annotations using DAVID, they are not of interest for us. Most of these annotations show a higher percentage of proteins length ranges from 1 to 12 (the first two turquoise bin) comparing to the whole human proteome database (the first two grey bin).

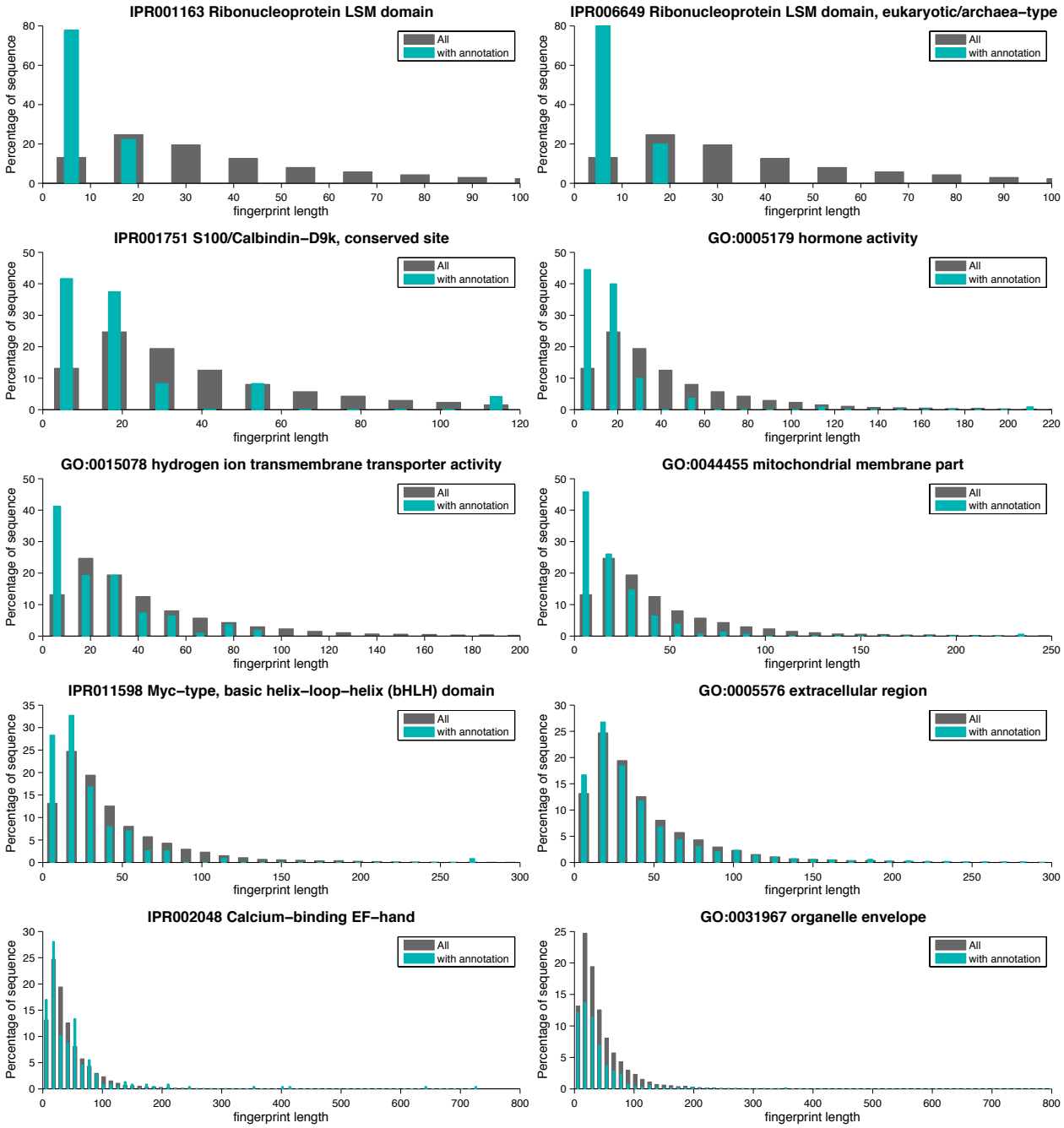
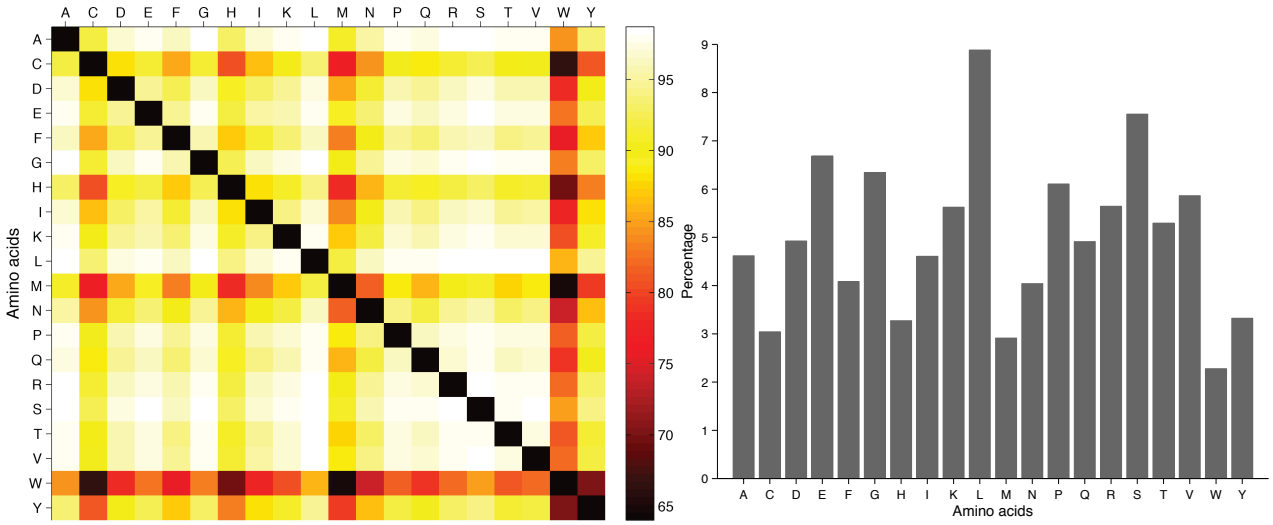


Fig. 18: The length distribution of gene annotations and fingerprints.

5.6 Choice of 2 amino acid combinations

To explore how would other 2 amino acid combinations influence the detection performance, here we analyse the uniqueness of all the possible choice of 2 amino acids combinations.

As illustrated in Fig. 19 (a), the uniqueness varies for different combinations. The higher the uniqueness the better choice of this combination. When the two amino acids occur more often, the uniqueness is relatively high. The combination of the two most frequent amino acids (see Fig. 19 (b)), leucine (L) and serine (S), shows the highest percentage of unique fingerprints (98.7%), and the combination



(a) Duplication ratio.

(b) AA composition.

Fig. 19: *a)* Uniqueness of all 2 amino acid combinations and *b)* amino acid composition for human complete proteome database.

of methionine (M) and tryptophan (W) has the lowest uniqueness (64.6%). The current choice of labeling, the combination of C and K, shows 89.8% of unique fingerprints, which is larger than the average percentage (87.3%). Stevens *et al.* presented the feasibility of labeling phenylalanine (F) and valine (V) to identify mRNAs translations (Stevens *et al.*, 2012), where the uniqueness of the combination of F and V is 94.8%, better than the choice of C and K.