Efficient Filtration for Similarity Search with Spaced k-mer Neighbors

Weiming Li*, Bin Ma†, and Kaizhong Zhang*
*Department of Computer Science, University of Western Ontario,
London, ON, Canada N6A 5B7,
Email:{wli5,kzhang}@csd.uwo.ca
†School of Computer Science, University of Waterloo,
Waterloo, ON, Canada N2L 3G1,
Email:binma@uwaterloo.ca

***Abstract*—In DNA and protein sequence similarity search, seeding (or filtration) has been widely used to trade search sensitivity with search speed. In this paper, a new seeding method, called spaced k-mer neighbors, is introduced to provide a more efficient tradeoff between the speed and sensitivity in protein similarity search. The new method pre-selects a set of spaced k-mers as neighbors, and uses the neighbors to detect hits between the query and database sequences. An efficient heuristic algorithm is proposed for the neighbor selection. We demonstrate that the method can improve the tradeoff efficiency over existing seeding methods.**

***Keywords*-spaced seeds, homology search, similarity search.**

I. Introduction

Sequence similarity search is vitally important for the study of DNA and proteins in modern molecular biology, and has been actively researched in bioinformatics. Due to the large size of the DNA and protein sequence data, efficiency is a top-priority in the development of similarity searching algorithms. A commonly used technique is the tradeoff between speed and sensitivity by using filtration. A filtration method quickly identifies the potential similarity regions between the query and the database sequences, then only these similarity candidates are further examined by a more accurate (and usually slower) algorithm.

Earlier development of filtration was exemplified by the BLAST program, which uses a consecutive k-mer match as a seed to filter out the potential similarities between two DNA sequences [1]. Since the finding of exact k-mer matches is relatively simple, this seeding idea greatly improved the search speed of BLAST. However, because not all DNA similarities have a long k-mer match, some similarities are lost due to this filtration, resulting in reduced sensitivity. The parameter k, can be used to tradeoff between speed and sensitivity.

For a better tradeoff, the PatternHunter [2] program first proposed the optimized spaced seed algorithm. A spaced seed is represented by a binary string such as 111*11*1, where the 1 means "required match" and the * means "don't care". The number of 1s in the seed is called the weight of the seed. Given a spaced seed, the algorithm will use the exact match at the required matching positions as the filtration criterion. To the surprise of many, the PatternHunter paper demonstrated that by optimizing the configuration of the positions of the 1 and * in the spaced seed, a weight-k spaced seed is significantly more sensitive than the weight-k consecutive seed used by BLAST. This spaced seed idea was later on implemented in the BLAST program for the searching of distant similarities.

Since PatternHunter, significant researches have been carried out to find more efficient ways to tradeoff search sensitivity and speed. In particular, the PatternHunter II paper [3] studied the multiple spaced seeds, where several spaced seeds with different shapes are used simultaneously to increase the sensitivity. The vector seed and multiple vector seeds methods [4], [5] modifies the spaced seed by allowing one or a few of the required positions to be mismatches. An earlier review of some of these developments can be found in [6]. Most of these developments in more efficient seeding are designed for DNA similarity search and regard a mismatch between two letters a negative evidence to the similarity. Recently, the spaced seed method has also been adopted for the efficient reads mapping for next generation genome sequencing analysis [7].

While for DNA sequences a mismatch between two bases is considered a negative evidence for the DNA homology, protein sequences are different. According to the BLOSUM [8] and PAM [9] matrices that measure the amino acid similarities, the mismatch between a pair of amino acids can be scored positively. So, the seeding methods used for DNA similarity search need to be adjusted to work on proteins. The BLASTp program extended the consecutive seed idea by using the approximate match of a pair of k-mers that have matching score greater than or equal to a given threshold as seeds. The k-mer pairs with higher-than-threshold matching scores are pre-calculated and indexed for efficient finding of those approximate matches. Particularly, the BLASTp program build a DFA to search for the positions of hits at the scanning stage. The multiple vector seeds method [4] combined BLASTp method with multiple spaced seeds. Another approach for protein similarity search in the literature is to classify the amino acids into several classes [10] or hierarchical classes [11] so that the spaced seeds designed for DNA sequences can work on amino acid classes.

In this paper we point out that the k-mer pairs selection by using the BLOSUM [8] score threshold is not optimal for protein homology search. The reasons for this sub-optimality are examined, and a new filtration method, namely spaced k-mer neighbors, is proposed.

II. Preliminaries and Notations

Given a pair of query and database sequences, a hit denotes a pair of positions, each from one of the two sequences. In a filtration method for similarity search, a *hit* indicates a possible similarity around the two positions that deserves further examination. A *seeding scheme* defines the criterion that two positions of the query and database sequences generate a hit. A spaced seed x is denoted by a binary string over alphabet $\{1, *\}$, where 1 indicates the required matching positions and $*$ for "don't cares". Given a spaced seed $x = x_0 \dots x_{l-1}$ and a sequence s, the *spaced k-mer* at position i of s is defined as the string $t_0 \dots t_{l-1}$ such that

$$t_j = \begin{cases} s[i+j] & \text{if } x[j] = 1 \\ * & \text{if } x[j] = * \end{cases}$$

For example, if the spaced seed is 11*1 and sequence is AMKMKK, then the spaced k-mer at position 0 is AM*M and at position 1 is MK*K. Thus, in a spaced seed method, two positions of the query and database sequences generate a hit if their spaced k-mers at those positions are identical. A *similarity matrix* defines the similarity score between two letters. For example, the BLOSUM matrix defines the similarity between each pair of amino acids. Given a similarity matrix M, the score between two sequences s_1 and s_2 with equal length l, denoted by $f_M(s_1, s_2)$, is calculated by $\sum_{j=1}^{l} M(s_1[j], s_2[j])$. A high-scoring segment pair (HSP), is a pair of sequences such that the $f_M(s_1, s_2)$ have score higher than a specified threshold. For the sake of presentation clarity, for any given scoring matrix M, we define $M[*, *] = 0$. The similarity score between two spaced k-mers t_1 and t_2, are denoted by $f_M(t_1, t_2)$, and computed as $\sum_{j=1}^{l} M(t_1[j], t_2[j])$.

For protein homology search, a sequence alignment is usually the concatenation of several HSPs. The general procedure for a homology search algorithm first utilizes a seeding method to detect hits. Then an extension procedure is used to check if there is an HSP around the hit. Only when the HSP is successfully detected, a Smith-Waterman algorithm is called to generate sequence alignment. The main optimization goal of the seeding method is then becoming the efficient detection of the HSPs.

The sensitivity of a seeding method, is the portion of HSPs that generate at least one hit. The selectivity of a seeding scheme, is the reciprocal of the probability that two random positions of a query and database sequences is a hit. If two variables x and y are linearly correlated to each other, we denote $x \propto y$.

III. The Dependencies that May Affect Sensitivity

If two seeding schemes have the same selectivity, they normally produce similar number of hits in the HSPs. The key reason that they may have very different sensitivity is the different distributions of the hits. Since each HSP requires only one hit for its detection, having more than one hit in an HSP is a waste. Consequently, if one distributes hits more evenly across different HSPs, it will detect more HSPs, or in another word, have a higher sensitivity.

This appeared to be the main reason that the spaced seed proposed in PatternHunter outperformed the consecutive seed used by BLAST. More specifically, consider that an HSP has a consecutive seed hit at position i. Thus the two k-mers $(a_i a_{i+1} \dots a_{i+k-1})$ and $(b_i b_{i+1} \dots b_{i+k-1})$ exactly match each other. Consider the two k-mers $(a_{i+1} \dots a_{i+k-1} a_{i+k})$ and $(b_{i+1} \dots b_{i+k-1} b_{i+k})$, they would have a high probability to match because this second hit requires only one additional letter match between a_{i+k} and b_{i+k}. Thus, when consecutive seeds are used, there is a tendency that once an HSP is hit, it is hit multiple times. However, for spaced seeds, this tendency is greatly reduced.

Note that the above argument still holds even if the seeding scheme only requires the two k-mers to match approximately. Specifically, when the BLOSUM matrix M is used for the protein sequences, BLASTp requires the matching score $f_M(a_i a_{i+1} \dots a_{i+k-1}, b_i b_{i+1} \dots b_{i+k-1})$ to be greater than a threshold for considering $(a_i a_{i+1} \dots a_{i+k-1})$ and $(b_i b_{i+1} \dots b_{i+k-1})$ as a hit. Even under this approximate match scenario, the hit at position i will still increase the hit probability at position $i+1$. This is the first dependency one needs to get rid of in order to increase the sensitivity. Based on this observation, the first proposed change over the BLASTp's seeding scheme is the following:

Change 1: Replace the k-mer high-scoring matches with spaced k-mer. This is similar to what was proposed in the multiple vector seeds paper [4].

The second dependency we examine is the dependencies between amino acids within a k-mer. For an amino acid a, let $P(a)$ be the probability that a occurs in the query and database sequences. For two amino acids a and b. $P(a, b)$ is the probability that they occur at the same position in the two sequences of an HSP. According to [8], the BLOSUM matrix M is such that

$$M[a, b] \propto \log \frac{P(a, b)}{P(a)P(b)}$$

Thus, for two k-mers a and b, if the amino acids at different positions of a k-mer are independent to each other, then

$$f_M(A,B) \propto \log \prod_{i=1}^{k} \frac{P(a[i],b[i])}{P(a[i])P(b[i])} = \log \frac{P(A,B)}{P(A)P(B)}$$

Here $P(A,B)$ is the probability that the k-mer pair (A,B) occurs in the HSPs, and $P(A)P(B)$ is the probability that the k-mer pair occurs randomly at two random positions of the query and database sequences. By using the high-scoring k-mer matches as hits, BLASTp essentially requires that the hitting k-mer pairs to have a high ratio between its occurring frequency in HSPs and in random sequence positions. As a result, the seeding scheme can maximize both the sensitivity and the selectivity simultaneously.

However, the above argument required the independence between the amino acids at different positions of a k-mer, which may not hold in reality. To illustrate this, the following experiment is carried out and the result is shown.

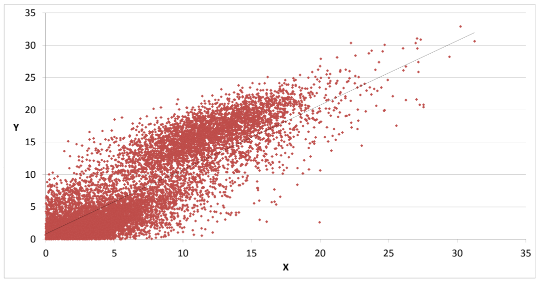

Figure 1. The two values of $2\log_2 \prod_{i=1}^{k} \frac{P(a[i],b[i])}{P(a[i])P(b[i])}$ (Y-axis) and $2\log_2 \frac{P(A,B)}{P(A)P(B)}$ (X-axis) may be different in reality.

On real HSPs between human and mouse proteins, for $k = 3$, Figure 1 shows the differences between $2\log_2 \prod_{i=1}^{k} \frac{P(a[i],b[i])}{P(a[i])P(b[i])}$ (Y-axis) and $2\log_2 \frac{P(A,B)}{P(A)P(B)}$ (X-axis), and Figure 2 shows the distribution of $2\log_2 \frac{P(A,B)}{P(A)P(B)}$ for 3-mer pairs with BLOSUM62 score greater than or equal to 11.

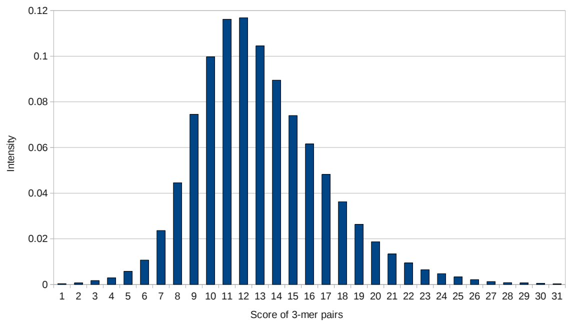

Figure 2. The distribution of values of $2\log_2 \frac{P(A,B)}{P(A)P(B)}$ for 3-mer pairs with BLOSUM62 score $>= 11$ used in BLAST

Because of these discrepancies, the simple use of BLOSUM scores as selection criteria for k-mer pairs is not anymore optimal. Instead, the real $P(A,B)/(P(A)P(B))$ should be used.

Change 2: Instead of using the BLOSUM score as hitting criterion, a set S of k-mer pairs are pre-selected. Based on the real $\frac{P(a,b)}{P(a)P(b)}$, the match of a pair of k-mers is a hit if and only if it belongs to S.

The choice of using a pre-selected set of k-mer pairs also enabled the removal of another type of dependencies between k-mer pairs. We illustrate the situation using consecutive seed first. Consider two k-mers pair $(a_1 \dots a_k) \sim (b_1 \dots b_k)$ and $(a_2 \dots a_k\ x) \sim (b_2 \dots b_k\ y)$. Each of these two pairs can detect their own set of HSPs. However, the two detected HSP sets intersect each other and share all the HSPs that were detectable by the $(k+1)$-mer pair $(a_1 \dots a_k\ x) \sim (b_1 \dots b_k\ y)$. Thus, having one of the two k-mer pairs in S reduces the benefit of including the other.

This phenomena is illustrated in the following experiment. Consider the three 3-mer pairs:

I: $(a_1a_2a_3) \sim (b_1b_2b_3)$, II: $(a_2a_3x) \sim (b_2b_3y)$, and II′: $(xa_2a_3) \sim (yb_2b_3)$

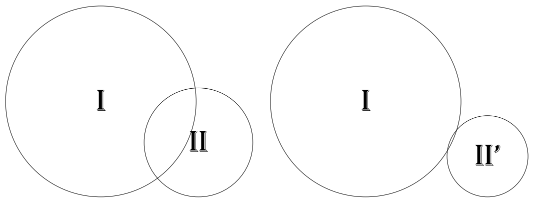

Figure 3. Comparison of I: (LSC)∼(LAC); II: (SCS)∼(ACA); II′: (SSC)∼(AAC). I and II′ share 3 HSP; I and II share 10517 HSPs while I can hit 70376 HSPs alone.

In Figure 3, the Venn Diagram of the HSPs hit by the three pairs indicate that I and II share a significant portion of HSPs.

Note that this dependency will be reduced by using spaced seed as two adjacent hits share a fewer number of positions. However, the further consideration of dependencies between (spaced) k-mer pairs will further reduce this dependency.

Change 3: The selection of k-mer pair set S should take into account of dependencies between different pairs.

IV. Spaced k-mer Neighbors Method

In this section we propose the spaced k-mer neighbors method for increasing the similarity search sensitivity. A simple algorithm to optimize the k-mer pairs is also proposed.

A. Selecting a Good Set of k-mer Neighbors

In this section we will present our algorithm for the selection of the spaced k-mer neighbors.

A good set of k-mer neighbors should hit the most of HSPs. Intuitively, if we have a set $\mathcal{H}$ of fine sampling of the real HSPs, then we can find a good set (according to some criteria) of k-mer neighbors to hit all of the HSPs in $\mathcal{H}$. Since each HSP can be viewed as a set of k-mers, the problem of selecting a good set of k-mer neighbors becomes a hitting set problem.

As for the criteria of selecting hitting k-mer neighbors set, a good set $\mathcal{S}$ of k-mer neighbors should have high sensitivity. This means that each individual $s = (A, B) \in S$ should hit many HSPs implying that $P(A, B)$ should be relatively high. At the same time, a good set $\mathcal{S}$ of k-mer neighbors should not produce too many random hits. This means that $P(A) * P(B)$ should be low. Therefore for an individual k-mer neighbor s, the ratio $P(A, B)/P(A) \cdot P(B)$ should be high. However, this will not solve the problem of k-mer pairs dependencies. Consider the situation that there is another $s_1 = (A_1, B_1)$ with a high $P(A_1, B_1)/P(A_1) \cdot P(B_1)$ ratio. Its detected HSPs have a large overlap with the detected HSPs of s. Including both s and s_1 will not increase much the ability to hit HSPs but will increase the probability of producing random hits. If there is a third k-mer pair $s_2 = (A_2, B_2)$ such that its detected HSPs cover the non-overlapping part of detected HSPs of s_1 and $P(A_2) \cdot P(B_2) < P(A_1) \cdot P(B_1)$, then including s and s_2 is a better choice.

Therefore, the criteria should be to find a hitting set $\mathcal{S}$ of k-mer neighbors that hits all HSP sets and minimizes $\sum_{s \in S} w(s)$, where $w(s) = P(A) \cdot P(B)$. With this criteria, random hits will be minimized and the amount of dependencies will be reduced.

The problem of selecting a good set of k-mer neighbors becomes a weighted minimum hitting set problem as follows:

Let $\mathcal{K}$ be the set of all the k-mers over a finite set alphabet Σ, such that:

$$\mathcal{K} = \{u | u \text{ is a } k\text{-mer}\}$$

In case of protein homology search, Σ is the twenty commonly used amino acids.

Let $\mathcal{U}$ be a finite set contains all of the spaced k-mer pairs, such as:

$$\mathcal{U} = \{(u, v) | \ \forall\, u, v \in \mathcal{K}\ \} \tag{1}$$

Consider an HSP $\mathcal{P}$ of two sequences of t_1 and t_2 with length l, then $\mathcal{P} \subseteq \mathcal{U}$ can be represented as a set of k-mer pairs as following:

$$\mathcal{P} = \left\{ (u, v) \middle| \begin{array}{l} u \in t_1[i \ldots i + k - 1] \\ v \in t_2[i \ldots i + k - 1] \\ \forall i = \{1, \ldots, l - k + 1\} \end{array} \right\} \tag{2}$$

And given a set $\mathcal{H}$ which contains a finite number of HSPs such as:

$$\mathcal{H} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\} \tag{3}$$

We are to find a k-mer neighbors set $\mathcal{S} \subseteq \mathcal{U}$ to hit each HSP in $\mathcal{H}$ at least once, or $\mathcal{S} \cap \mathcal{P}_i \neq \emptyset, \forall i = 1, \ldots, n$, such that the selectivity $\sum_{s \in S} w(s)$, where $s = (A, B)$ and $w(s) = P(A) \cdot P(B)$, is minimized.

The above described problem of finding a set $\mathcal{S}$ with respect to a weight function $w : \mathcal{U} \rightarrow \Re^+$ is the well-known Weighted Minimum Hitting Set (WMHS) problem. Since we are comparing against the BLASTp-like high scoring matches methods having selectivity M, we need to further constrain that $\mathcal{S}$ also has a selectivity less than or equal to M, i.e. $\sum_{s \in S} w(s) \leq M$. In reality, with an iterative incremental process of constructing $\mathcal{S}$, k-mer neighbors will be added to $\mathcal{S}$ until the weight reach M.

Finding an optimal set $\mathcal{S}$ is not feasible because WMHS is a known NP-hard problem [12]. Hence our proposed algorithm of finding a good set $\mathcal{S}$ is the result of adopting the popular greedy solution, which is reviewed in paper [13], of the WMHS problem.

The k-mer neighbors set $\mathcal{S}$ is greedily selected by using a training set $\mathcal{H}$ of HSPs. Let $F_{\mathcal{H}}(m_1, m_2)$ be the total counts of the k-mer pair (m_1, m_2) in $\mathcal{H}$, and $f(m)$ be the frequency of a k-mer m in a protein sequence. The selection algorithm greedily selects the k-mer pair that maximizes $\frac{F_{\mathcal{H}}(m_1, m_2)}{f(m_1) f(m_2)}$, and dynamically updates $F_{\mathcal{H}}(m_1, m_2)$ by removing the HSPs that are hit by the currently selected k-mer pairs.

The detailed algorithm is given in Algorithm 1. Notice that, $F_{\mathcal{H}}(m_1, m_2)$ is a counter of each k-mer pair occurs in a training set of HSPs, and $f(m)$ is fixed and calculated using all the protein sequences released in July 2011 from the NCBI Refseq [14] database. In the actual implementation of Algorithm 1, each HSP in $\mathcal{H}$ has been indexed with an unique id, and for each k-mer pair with counter $F_{\mathcal{H}}$, we need to maintain a list of ids of the contributing HSPs. Because of maintaining such list, in the update process, we could decrease the value of counter $F_{\mathcal{H}}$ and remove the corresponding HSPs directly from the list rather than scanning though $\mathcal{H}$ for it.

Algorithm 1 SelectNeighbor

```
Require: A set $\mathcal{H}$ of HSPs, a positive integer $k$, a spaced
    seed $s$, and a targeted selectivity $M$
Ensure: A set $\mathcal{S}$ of spaced $k$-mer neighbors
1: repeat
2:     $\mathcal{S} \leftarrow \phi$; $t \leftarrow 0$
3:     for all $k$-mer pair $(m_1, m_2)$ do
4:         count the frequency $\frac{F_{\mathcal{H}}(m_1,m_2)}{f(m_1)f(m_2)}$ occurring in $\mathcal{H}$
5:     end for
6:     Let $(m_1, m_2)$ be the pair that maximize $\frac{F_{\mathcal{H}}(m_1,m_2)}{f(m_1)f(m_2)}$
7:     $S \leftarrow S \cup \{(m_1, m_2)\}$ and $t \leftarrow t + f(m_1)f(m_2)$
8:     Remove all HSPs that are hit by $(m_1, m_2)$ from $\mathcal{H}$
9: until $\mathcal{H}$ is empty or $t \geq \frac{1}{M}$
```

Notice that SelectNeighbor can be implemented efficiently. When executed for the first time, steps 3-5 require the enumeration of every HSP in $\mathcal{H}$ to calculate $F_{\mathcal{H}}$. This takes $O(N)$ time and N is the total length of the HSPs. However, in the future repetition, $F_{\mathcal{H}}$ does not need a complete recalculation and can be updated dynamically. When an HSP is being removed from $\mathcal{H}$ in Step 8, we only need to reduce the counter $F_{\mathcal{H}}(m_1', m_2')$ by $\frac{1}{f(m_1', m_2')}$ for each pair of k-mers (m_1', m_2') occurring in $\mathcal{H}$. A priority queue data structure is used to store all the counters and efficiently find the optimal (m_1, m_2) in Step 6. Each update of a counter takes only $\log N$ time because the size of the queue is at most N. And there are at most N updates. Therefore, the total execution time is $O(N \log N)$ if the algorithm is implemented as described above.

V. Experimental Results

Four different seeding schemes were compared in our experiments. The first seeding scheme is BLASTp's default scheme. That is, with a consecutive seed 111, all 3-mer pairs that have BLOSUM62 score greater than or equal to 11 are regarded as neighbors. The second scoring scheme is similar to the first one, except that a spaced seed 11*1 is used. The third and the fourth schemes all use the SelectNeighbor algorithm to train the 3-mer neighbors from training HSPs. But they use a consecutive seed 111 and a spaced seed 11*1, respectively.

The protein sequences used in the study are the complete proteomes of human, mouse, drosophila (fruit fly), cow, and pig from the Uniprot [15] database. The sequences were downloaded from the Uniprot database on April 18, 2012. The five proteomes contain 65481, 46439, 17516, 26588, and 19572 proteins, respectively.

To compare the performances of different seeding methods, we restricted each method's selectivity to be approximately the same, and compared their sensitivity on the HSPs between a pair of the above-mentioned five proteomes. On SHARCNET, SSearch program [16], which is an efficient implementation of the Smith-Waterman algorithm, was used to generate the benchmark HSPs. Computing the local alignments results from SSearch program is actually very time consuming task because the Smith-Waterman algorithm runs in quadratic time and tens of thousands of proteins sequences are used. However, we were able to simultaneously run many SSearch programs by spreading it on different nodes across the SHARCNET clusters after efficiently splitting the task into smaller ones, we reduced our wait time for the same results by ten folds.

The sensitivity of each seeding method is measured by the percentage of the benchmark HSPs detected by the method. The selectivity of a set $\mathcal{S}$ of k-mer neighbors is calculated by $1/\sum_{(m_i,m_j)\in\mathcal{S}} P(m_i)P(m_j)$. This is the reciprocal of the probability that two random positions of a query and database sequences produce a hit.

For each such comparison, half of the protein sequences are randomly selected for parameter training for the seeding methods, and the remaining half are used for testing the

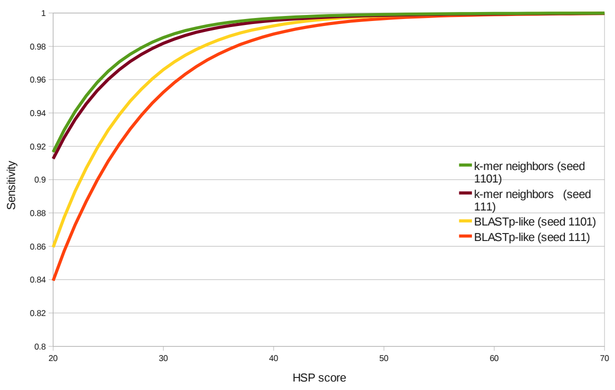

Figure 4. Sensitivity of each method in comparison on HSPs of human versus mouse with BLOSUM62 score $\geq X$.

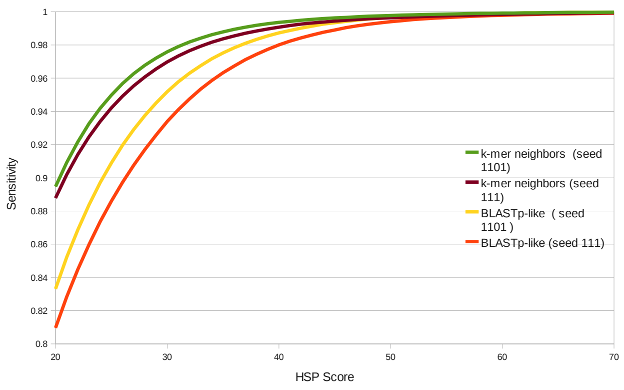

Figure 5. Sensitivity of each method in comparison on HSPs of human versus drosophila with BLOSUM62 score $\geq X$.

performance. Figure 4 shows the experiment we conducted on the complete proteome of human and mouse. There are 38982312 and 38891790 HSPs generated by SSearch from the training and testing proteins, respectively. The sensitivity curves clearly indicate that the spaced k-mer neighbors selected by Algorithm SelectNeighbor has a much better sensitivity. The same experiment with the human and drosophila produced similar result (Figure 5). There are 19807012 HSPs in the training data and 19908095 HSPs in the testing data from the protein sequences of human versus drsophila generated by SSearch.

In the above experiments, the training and testing are on different sets of proteins from the same organisms. We further studied the performance of spaced k-mer neighbors when the training and testing were on different pairs of organisms. The same set of k-mer neighbors trained using the training HSPs of human versus mouse was used. Figure 6 shows the results with the testing HSPs of human versus drosophila; and Figure 7 shows the testing with 995423 HSPs generated with 1957 randomly selected pig proteins versus 2659 randomly selected cow proteins. Both figures

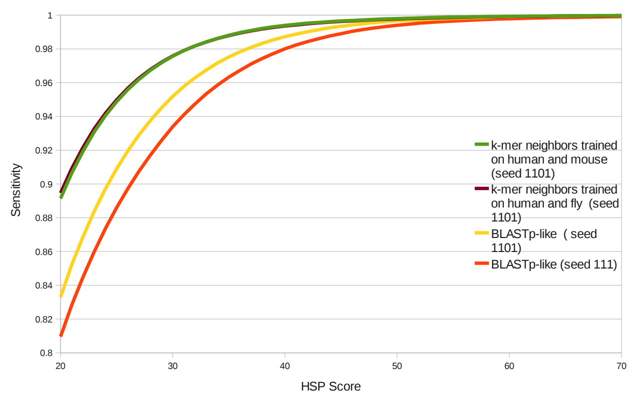

Figure 6. Sensitivity of each method in comparison on HSPs of human versus drosophila with BLOSUM62 score $\geq X$.

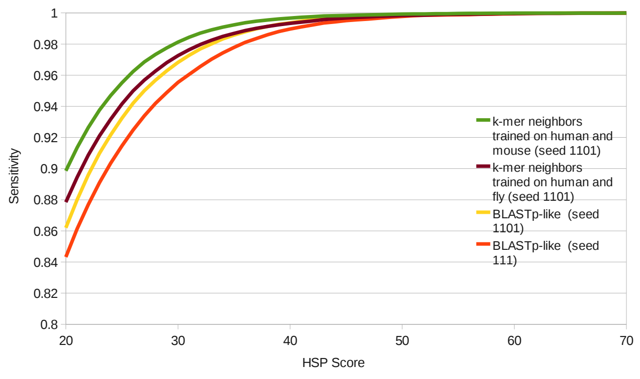

Figure 7. Sensitivity of each method in comparison on HSPs of pig versus cow with BLOSUM62 score $\geq X$.

suggest that the spaced k-mer neighbors still work very well even if the training data and testing data are from different organisms.

VI. Conclusion

A new spaced k-mer neighbor method is proposed for more efficient tradeoff between the sensitivity and selectivity in protein similarity search. An efficient heuristic algorithm is provided to pre-select the spaced k-mer neighbors from a large set of training HSPs. Experiments showed that the method can significantly increase the search sensitivity at the same selectivity.

Acknowledgement

This work was supported by the Natural Sciences and Engineering Research Councilof Canada (NSERC) and Sharc-Net.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, “Basic local alignment search tool,” *J. Mol. Biol.*, vol. 215, no. 3, pp. 403–410, 1990.

[2] B. Ma, J. Tromp, and M. Li, “PatternHunter: faster and more sensitive homology search,” *Bioinformatics*, vol. 18, no. 3, pp. 440–445, 2002.

[3] M. Li, B. Ma, D. Kisman, and J. Tromp, “PatternHunter II: Highly sensitive and fast homology search,” *J. Bioinf. and Comp. Biol.*, vol. 2(3), pp. 417–440, 2004.

[4] D. Brown, “Multiple vector seeds for protein alignment,” in *Proc. of 4th Workshop on Algorithms in Bioinformatics (WABI)*, 2004, pp. 170–181.

[5] B. Brejova, D. G. Brown, and T. Vinar, “Vector seeds: an extension to spaced seeds,” *Journal of Computer and System Sciences*, vol. 70, no. 3, pp. 364–380, 2005, early version appeared in WABI 2003.

[6] D. G. Brown, M. Li, and B. Ma, “A tutorial of recent developments in the seeding of local alignment,” *Journal of Bioinformatics and Computational Biology*, vol. 2, no. 4, pp. 819–842, 2004.

[7] H. Lin, Z. Zhang, M. Zhang, B. Ma, and M. Li, “Zoom! zillions of oligos mapped,” *Bioinformatics*, vol. 24, no. 21, pp. 2431–2437, 2008.

[8] S. Henikoff and J. G. Henikoff, “Amino acid substitution matrices from protein blocks,” *Proc. of the National Academy of Sciences of the United States of America*, vol. 89, no. 22, pp. 10 915–10 919, 1992.

[9] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt, “A model of evolutionary change in proteins,” *Atlas of Prot. Seq. and Struct.*, vol. 5, pp. 345–352, 1978.

[10] W. Li, B. Ma, and K. Zhang, “Amino acid classification and hash seeds for homology search,” *Bioinformatics and Computational Biology*, pp. 44–51, 2009.

[11] A. Gambin, S. Lasota, M. Startek, M. Sykulski, L. Noé, G. Kucherov *et al.*, “Subset seed extension to protein blast,” 2011.

[12] R. Karp, “Reducibility among combinatorial problems,” *50 Years of Integer Programming 1958-2008*, pp. 219–241, 2010.

[13] A. Cincotti, V. Cutello, and F. Pappalardo, “An ant-algorithm for the weighted minimum hitting set problem,” in *Swarm Intelligence Symposium, 2003. SIS’03. Proceedings of the 2003 IEEE.* IEEE, 2003, pp. 1–5.

[14] K. Pruitt, T. Tatusova, and D. Maglott, “Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins,” *Nucleic acids research*, vol. 35, no. suppl 1, pp. D61–D65, 2007.

[15] U. Consortium *et al.*, “Reorganizing the protein space at the universal protein resource (uniprot),” *Nucleic Acids Res*, vol. 40, pp. D71–D75, 2012.

[16] X. Huang, R. C. Hardison, and W. Miller, “A space-efficient algorithm for local similarities,” *CABIOS*, vol. 6, pp. 373–381, 1990.