Long spaced seeds for finding similarities between biological sequences

LUCIAN ILIE[1]
Department of Computer Science
University of Western Ontario
N6A 5B7, London, Ontario, CANADA
e-mail: ilie@csd.uwo.ca

SILVANA ILIE[2]
Centre for Mathematical Sciences
Numerical Analysis, Lund University
Box 118, SE-221 00 Lund, SWEDEN
e-mail: silvana@maths.lth.se

***Abstract*—Homology search finds similar segments between two biological sequences, such as DNA or protein sequences. A significant fraction of the computing power in the world is devoted to finding similarities between biological sequences. The introduction of optimal spaced seeds in [Ma et al., Bioinformatics 18 (2002) 440–445] has increased both the sensitivity and the speed of homology search and it has been adopted by many alignment programs such as BLAST. In spite of significant amount of work, there are no algorithms able to compute long good seeds. We present a different approach here by introducing a new measure that has two desired properties: (i) it is highly correlated with sensitivity of spaced seeds and (ii) it is easily computable. Using this measure we give algorithms that compute better seeds than all previous ones. The fact that sensitivity is not required is essential as it enables us to compute very long good seeds, far beyond the size for which sensitivity can be computed.**

***Index Terms*—homology search, spaced seed, sensitivity, PatternHunter, BLAST**

I. INTRODUCTION

HOMOLOGY search finds similar segments between two biological sequences, such as DNA or protein sequences. A significant fraction of the computing power in the world is dedicated to performing such tasks. The increase in genomic data is quickly outgrowing computer advances and hence better mathematical solutions are required. As the classical dynamic programming techniques of [19], [25] became overwhelmed by the task, popular programs such as FASTA [15] and BLAST [1] used heuristic algorithms. BLAST used a filtration technique in which positions with short consecutive matches, or *hits*, were identified first and then extended into local alignments. Speed was traded for sensitivity since longer initial matches missed many local alignments, hence decreasing sensitivity, whereas shorter initial matches produced too many hits, thus decreasing speed.

A breakthrough came with PatternHunter [17] where the hits were no longer required to consist of consecutive matches. Precisely, PatternHunter looks for runs of 18 consecutive nucleotides in each sequence such that only those specified by 1's in the string `111*1**1*1**11*111` are required to match. Such a string is called a *spaced seed* and the number of 1's in it is its *weight*. Using this notion, BLAST required a hit according to a *consecutive* seed such as `11111111111`.

The filtration principle has been used before in approximate string matching [10], [22], [5] but the important novelty of PatternHunter was the use of optimal spaced seeds, that is, spaced seeds that have optimal sensitivity. (The sensitivity of a seed is the probability that it hits a random region of a given length.) Impressively, the approach of PatternHunter increases both the speed and sensitivity. Since then the idea has been adopted by the new versions of BLAST – MegaBLAST, BLASTZ – and other software programs [3], [21], [12].

Quite a few papers have been written about spaced seeds, evaluating the advantages of spaced seeds over consecutive ones [11], [4], [6], [16], showing that the relevant computational problems are NP-hard [14], [16], giving exact (exponential) algorithms for computing sensitivity [14], [11], [4], [6], [7], [4], polynomial time approximation schemes [16] or heuristic algorithms [14], [7], [23], adapting the seeds for more specific biological tasks [3], [21], [13], or building models to understand the mechanism that makes spaced seeds powerful [4], [26], [23].

In this paper we present a different approach. Previous work gave algorithms to compute the sensitivity exactly, which is NP-hard [16], or to approximate it. We introduce a combinatorial measure of complexity which attempts to replace sensitivity in the sense that it is highly correlated with it but easily computable. Our measure is based on string overlaps, hereto called *overlap complexity*. Seeds with low overlap complexity have high, or even optimal, sensitivity. Optimal seeds with weight up to 18 can be computed by considering a number of seeds with low overlap complexity and pick the best. With a much simpler and faster algorithm we obtain better seeds than those of [23] for weights between 19 and 24.

For higher weights, no good seeds are known. Note that computing long good seeds is of practical importance since some software tools, such as MegaBLAST [28] and SSAHA [20], use long seeds. Great difference in speed allows our algorithm to compute seeds long enough for all purposes one can imagine – we provide a list of seeds of selected weights up to 64. It is interesting that the sensitivity of these seeds cannot be computed by any of the existing algorithms and likely it will never be since the problem is NP-hard.

[1]Research partially supported by NSERC and CNRS.
[2]Supported by a postdoctoral fellowship from the Natural Science and Engineering Research Council of Canada.

II. Spaced seeds

A *spaced seed* is any[3] string consisting of `1`'s and `*`'s; `1` stands for a 'match' and `*` for a 'don't care' position. For a seed s, the *length* of s, denoted by ℓ, is the total number of characters in s and the *weight* of s, denoted by w, is the number of 1's in s. For instance, the length and weight of PatternHunter's seed `111*1**1*1**11*111` are 18 and 11, respectively.

Given two DNA sequences and a seed s, we say that s simultaneously matches (hits) the two sequences at given positions if each `1` in s corresponds to a match between the corresponding nucleotides in the two sequences; see Fig. 1 for an example using PatternHunter's seed.

The above process can be reformulated as follows. Assume there are two DNA sequences S_1 and S_2 such that the events that they are identical at any given position are jointly independent and each event is of probability p, called the *similarity* level. The sequence of equalities/inequalities between the two DNA sequences translates then into a sequence R of 1's and 0's, corresponding to matches and mismatches, that appear with probability p and $1-p$, respectively. Therefore, given an (infinite) Bernoulli random sequence R and a seed s, we say that s *hits* R (ending) at position k if aligning the end of s with position k of R causes all `1`'s in s to align with 1's in R; see Fig. 1.

The *sensitivity* of a seed s is the probability that s hits R at or before position n. Note that the sensitivity depends on both the similarity level p and the length of the homologous region n.

III. Seed overlaps

The hits of a seed can obviously overlap but overlapping hits will detect a single local alignment. Therefore, the sensitivity of a seed is inverse proportional with the number of overlapping hits, since the expected number of hits is the same for all seeds. Therefore, good seeds should have a low number of overlapping hits. The definite proof that (nonuniformly) spaced seeds (defined formally in the section) are better than consecutive seeds, due to [16], involves estimating the expected number of nonoverlapping hits. However, computing this number in general is as difficult as computing sensitivity. Therefore, we look here for simpler ways to detect low numbers of overlapping hits.

The discussion leading to our definition of overlap complexity is not given here because of limited space. We say only that there are three natural candidates, depicted in Fig. 2; the thick lines are the seeds, overlapping in various ways, whereas the dashed boxes show the part that is considered for counting: (i) the overlaps alone, (ii) the participant strings, and (iii) a fixed-size window containing the overlapping strings as prefixes.

Keeping in mind that a `*` aligned against a `1` will have to take value 1 with probability p and two `*`'s aligned against each other will take the same value, 0 or 1, with total probability $p^2+(1-p)^2$, longer overlaps will get penalized more by the candidate (i), which is wrong, whereas the candidate (ii) tends to penalize the short overlaps too much. An in-between choice is our third candidate. It is interesting to note that the value of p does not seem to be relevant as far as the obtained ranking is concerned. This is due to the fact that, even if the optimal seed may change with p, the differences in sensitivity are very small among these top seeds (compared, of course, for the same similarity level), a fact heavily exploited later. We shall therefore fix the value of p as 0.5.

The *overlap complexity* of a seed s is formally defined as

$$\mathsf{OC}(s) = \sum_{i=1}^{\ell-1} 2^{\sigma[i]} \ ,$$

where $\sigma[i]$ gives the number of pairs of `1`'s aligned together between s and its shift by i positions. (Equivalently, this means the number of pairs of `1`'s at distance i in s.) As an example, for PatternHunter's seed we have $\sigma = (5,5,5,4,4,3,3,4,3,2,3,3,3,2,3,2,1)$ and its overlap complexity is $\sum_{i=1}^{\ell-1} 2^{\sigma[i]} = 214$.

Note that, for any seed s, $\mathsf{OC}(s) = \mathsf{OC}(s^r)$ (s^r is the reversal of s) which is a property to be expected since the sensitivity of s and s^r is the same. Throughout our experiments we consider only one of s and s^r.

IV. Consecutive and uniformly spaced seeds

The spaced seed of PatternHunter clearly outperformed the former consecutive seeds but whether spaced seeds were always better than consecutive ones required investigation. In fact, it is not true for all spaced seeds: *uniformly spaced seed*, i.e., seeds of the form

$$u_{k,r} = {*}^r(1{*}^k)^w {*}^{\ell-r-w(k+1)} \ ,$$

for $k \geq 0$, are not better. Note that consecutive seeds are a particular case of this definition for $k = 0$. For the remaining ones, indication of their superiority has been given in [11], [6] but a rigorous proof had to wait until [16].

It can be proved that uniformly spaced seeds (consecutive ones included) are the worst with respect to overlap complexity. (Here "worst" means of highest overlap complexity.) Moreover, they are the only ones with this property. The formal proof is omitted. We shall only mention that, for fixed length and weight, all these seeds have the same overlap complexity, namely

$$\mathsf{OC}(u_{k,r}) = 2^w + \ell - w - 2 \ .$$

V. Correlation between overlap complexity and sensitivity

We present in this section the numerical data which experimentally show very good correlation between overlap complexity and sensitivity.

First, we consider the good seeds computed by Choi, Zeng, and Zhang in [7]. While we do not have the space to show all data, we mention only that all top seeds in the sensitivity ranking are at the top, or very close, of the ranking induced by overlap complexity. In all cases, at least one sensitivity-optimal seed tops the overlap ranking.

[3] From biological point of view only strings starting and ending with `1` are spaced seeds. The ones we shall ultimately compute satisfy this condition.

DNA sequence	A	G	G	C	A	C	T	G	T	A	T	G	T	A	T	A	T	C
DNA sequence	A	G	G	C	A	A	T	G	C	A	T	T	T	A	A	A	T	C
matches/mismatches	=	=	=	=	=	≠	=	=	≠	=	=	≠	=	=	≠	=	=	=
Bernoulli sequence	1	1	1	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1
spaced seed	1	1	1	*	1	*	*	1	*	1	*	*	1	1	*	1	1	1

Fig. 1. An example of a hit using PatternHunter's seed.

Fig. 2. Three possibilities for counting overlaps.

The opposite comparison, even more important, is shown in Table I. We give only the mean statistics. The second and third columns contain the mean and standard deviation, resp., of the difference between optimal sensitivity, $\mathcal{S}_{\text{optimal}}$ (taken from [7]), and the sensitivity of the seed with best overlap complexity for the same weight and length, $\mathcal{S}_{\text{best\_overlap}}$. (In fact, almost all differences are zero and the remaining ones are negligible, which proves a remarkable correlation between the two measures.) The data are computed for weight range 9 to 18. The optimal sensitivity for higher weights is unknown. The last two columns will be discussed later.

VI. Finding good spaced seeds using overlap complexity

The exact algorithms for computing sensitivity are all exponential, see [11], [4], [6], which is expected since the problem is NP-hard [16]. Some though are better than others. The one of [6] runs in time $\mathcal{O}(n\ell 2^{2(\ell-w)})$, which makes it the slowest. The other two have running times $\mathcal{O}(n\ell^2 2^{\ell-w})$ for [11] and a bit less, $\mathcal{O}(nw2^{\ell-w})$, for [4]. Therefore, finding optimal seeds by trying all of a given weight (and length) and selecting the best is computationally very expensive. In fact, it has been shown by [14] to be NP-hard for an arbitrary distribution. On the other hand, [16] explains why finding an optimal seed in an uniform distribution is probably not NP-hard. We refer to [16] for details.

Choi, Zheng and Zhang [7] used their $\mathcal{O}(n\ell 2^{2(\ell-w)})$-algorithm for sensitivity to find optimal seeds for weights up to 18. The total complexity of the exact algorithm is then $\mathcal{O}(\binom{\ell}{w} n\ell 2^{2(\ell-w)})$. They noticed that the sensitivity for homologous region of length $n = 2\ell$ is a good indicator for the target sensitivity for $n = 64$ (length 64 for homologous region has been recommended by [17]). Therefore, they have a heuristic algorithm of complexity $\mathcal{O}(\binom{\ell}{w} \ell^2 2^{2(\ell-w)})$.

A much faster heuristic algorithm has been proposed by Preparata et al. [23], where heuristics derived from a complicated analysis of a probability leakage model are used. The complexity of their algorithm is not explicitly given but, from the description of the tests performed, it seems to be something like $\mathcal{O}(2^w)$ in addition to computing the exact sensitivity for a few seeds.

The heuristic algorithm we derived from our overlap complexity is very simple: compute the seed with the lowest overlap complexity. With a heuristic using the fact that the best seeds must start and end with a number of `1`'s, the time complexity is roughly the same as the one of [23]. However, our seeds are always better. For each seed computed in [23], we found only one seed with minimum overlap complexity with the same length and weight (using the heuristic with the number of `1`'s at the ends) and computed its sensitivity for comparison. Table II shows the results for homologous regions of length 128, as used in [23]. (Our seeds' sensitivities are better also for length 64.)

Note that our algorithm described above is better than the heuristic algorithm of [7] since, being much faster, we can afford computing the sensitivity for a number of seeds with low overlap complexity and the optimum is found in all cases of [7]. In addition, our algorithm is much simpler than all the other ones.

VII. Long good seeds

So far we have seen that simple algorithms based on overlap complexity can produce remarkably good seeds. They are better than those produced by previous algorithms. However, the computation of the seeds with lowest overlap complexity requires investigation of all seeds of given weight and length. Even when using the heuristic with the `1`'s at the ends their number remains exponential. To compute much longer good seeds we need something much better. The simplicity of our approach will allow an extremely efficient heuristic algorithm for approximating overlap complexity which will be the one we are looking for. It runs fast and produces excellent results.

We shall say that `1` *flipped* is `*` and vice versa. For a seed s and a set of positions $i_1, i_2, \ldots, i_k$, we denote by $\mathsf{flip}(s, i_1, i_2, \ldots, i_k)$ the seed obtained from s by flipping the letters in positions $i_1, i_2, \ldots, i_k$. For instance, we have $\mathsf{flip}(\texttt{1*11*11***1}, 3, 6, 8) = \texttt{1**1**11**1}$. With this notation, the algorithm, called $\textsc{swap}(w, \ell)$, is described in Fig. 3.

Thus, the algorithm essentially swaps a `1` with a `*` as long as the overlap complexity can be improved and then swaps two `1`'s with two `*`'s with the same goal. Each swap greedily

TABLE I
Heuristic versus optimal sensitivity for weights 9 to 18 and length of homologous region 64.

similarity	$\mathcal{S}_{\text{optimal}} - \mathcal{S}_{\text{best\_overlap}}$		$\mathcal{S}_{\text{optimal}} - \mathcal{S}_{\text{heuristic\_overlap}}$	
	mean	standard deviation	mean	standard deviation
65%	0.000024	0.000043	0.000152	0.000217
70%	0.000033	0.000098	0.000454	0.000492
75%	0.000029	0.000061	0.000812	0.000810
80%	0.000089	0.000148	0.001153	0.001077
85%	0.000383	0.000809	0.001309	0.001449
90%	0.000270	0.000644	0.000707	0.001137

TABLE II
Comparing our seeds with the ones of [23] for homologous region of length 128. For each weight, the seed of [23] is given first and then our seed.

weight	good seeds (our seeds are second for each weight)	sensitivity under a similarity level 70%	80%	90%
18	`1111*11*1*111***11**1**11111`	0.1213710	0.634614	0.995108
	`111*111**1*1*11**1*11**11111`	0.1226640	0.639962	0.995514
19	`11111**11*1*1**111**1*1*11111`	0.0874472	0.555891	0.990715
	`111*11*1*11**11*1*1*11**11111`	0.0879040	0.558625	0.991186
20	`11111**11***111*1**1*1*11*11111`	0.0625721	0.482963	0.984702
	`1111*1*1*11**111**1*11**11*1111`	0.0629695	0.485533	0.985240
21	`11111*1**111*1*111**1**1**111111`	0.0439876	0.407405	0.973435
	`11111**11*1*1*11**11*1*11**11111`	0.0446784	0.414399	0.975833
22	`11111*1**11*11*1**111**1*1**111111`	0.0312319	0.347302	0.961702
	`1111*11**111**1*11*1*1*1*11**11111`	0.0315421	0.350986	0.963462
23	`111111*1*1**1**11*11***11*11*111111`	0.0216683	0.285175	0.939958
	`11111**11*1*1*1*11*1**111**11*11111`	0.0222235	0.294443	0.947310
24	`111111**1*1**11***111**11*1*11*111111`	0.0153424	0.240568	0.921973
	`1111*1*11**111**11*11**1*11*1*1*11111`	0.0155691	0.245343	0.927552

SWAP(w, ℓ)
- given: the weight w and length ℓ
- returns: a seed s with weight w, length ℓ and low overlap complexity

1. $s := \mathtt{1}^w\mathtt{*}^{\ell-w}$
2. **while** there are $i < j$ with $\{s[i], s[j]\} = \{\mathtt{1}, \mathtt{*}\}$ and $\mathsf{OC}(\mathsf{flip}(s, i, j)) < \mathsf{OC}(s)$ **do**
3. choose a (i, j) that reduces $\mathsf{OC}(\cdot)$ the most
4. $s := \mathsf{flip}(s, i, j)$
5. **while** there are $i_1 < i_2 < i_3 < i_4$ with $\{s[i_1], s[i_2], s[i_3], s[i_4]\} = \{\mathtt{1}, \mathtt{1}, \mathtt{*}, \mathtt{*}\}$ (multiset) and $\mathsf{OC}(\mathsf{flip}(s, i_1, i_2, i_3, i_4)) < \mathsf{OC}(s)$ **do**
6. choose a (i_1, i_2, i_3, i_4) that reduces $\mathsf{OC}(\cdot)$ the most
7. $s := \mathsf{flip}(s, i_1, i_2, i_3, i_4)$
8. **return**(s)

Fig. 3. The SWAP algorithm.

chooses the largest decrease in overlap complexity. We could of course swap more positions but then the complexity of a single swap, though polynomial, increases too much.

The algorithm is extremely simple but the results are remarkably good. As an example, PatternHunter's seed is computed immediately using 6 single swaps (step 4) and 2 double swaps (step 7). In Table I, the last two columns give the mean and standard deviation, resp., of the difference between optimal sensitivity, $\mathcal{S}_{\text{optimal}}$ (taken from [7]), and the sensitivity of the seed computed by SWAP, $\mathcal{S}_{\text{heuristic\_overlap}}$. All differences are again negligible, which proves a remarkable efficiency of our heuristic algorithm. (The data are computed for weight range 9 to 18.)

Perhaps more surprising is the fact that using the much simpler SWAP, we obtain better seeds in all cases than those of [23]. The differences are not big but it is the consistency with which it obtains good seeds that makes us believe that it should continue to produce good seeds even if we venture into the unknown territory of large weights, where sensitivity cannot be computed at all.

A problem that needs to be solved is predicting the best length. So far we worked with a fixed length (and computed the best overlap complexity) or chose the best in an interval of lengths (for a fixed weight), based on sensitivity (this is the case of SWAP for weights between 18 and 24). Since sensitivity is no longer there to help, we shall interpolate the optimal lengths given by SWAP for weights up to 24. The line given by the least squares method, see, e.g., [9], gives a good approximation as can be seen in the plots drawn with MAPLE software ([18], [8]) in Fig. 4 where the precise equations are also shown. (We note that some of the lengths used as basis for interpolation may be slightly higher, since sensitivity could not be computed for a large enough interval. This seems also consistent with the interpolation we used.) All lengths are computed or predicted for length of the homologous region 128.

Using SWAP, we computed good seeds for all weights between 25 and 64, similarity levels 70%, 80%, and 90%, and length of homologous region 128. The implementation of SWAP is straightforward. The NTL library, [24], was employed to deal with the large numbers of overlaps. A selection of these seeds is shown in Table III.

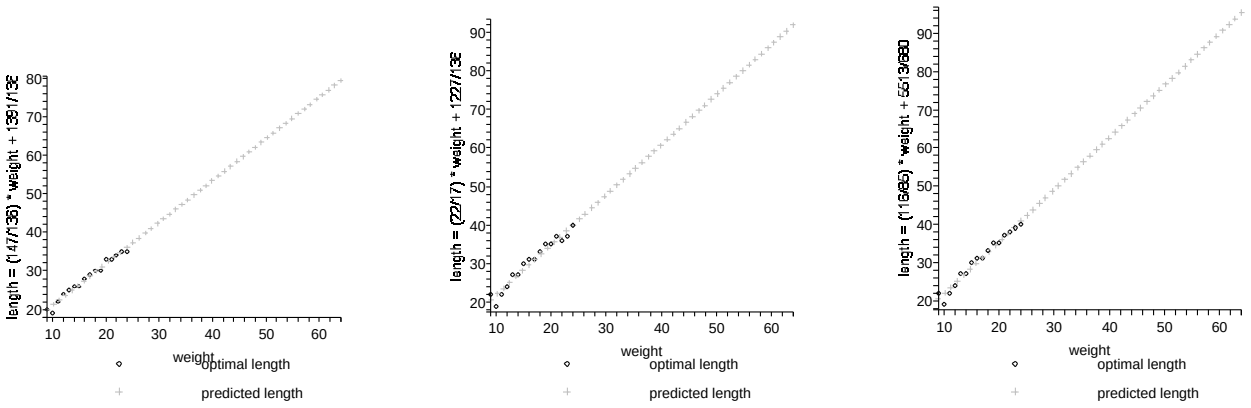

Fig. 4. Predicting the length by least squares method. From left to right, the similarity is 70%, 80%, and 90%, respectively.

TABLE III
A selection of long good seeds computed by SWAP for length of homologous region 128. We have computed good seeds for all weights between 25 and 64 but they are not shown here due to limited space.

weight	similarity	good seeds	length
25	70%	1111*11*1*1*111**111*1*11**11*11*1111	37
	80%	1111*11*1*1*11**11*1*1***111**11*1**11111	41
	90%	1111*1*1**111**11*1*11**1**11*1*1*11**1111	42
26	70%	1111*11*1*111**11*1*1*1111**11*11*1111	38
	80%	111111**11*1*1*1*1**1*1111**1*11**11**1111	42
	90%	11111*1**1*11**111*1*1**11**11*1*1*11**1111	43
27	70%	11111*11*1*111*11**111*1*11*1*11**11111	39
	80%	111111**111**1*11*1*1*1*11**11**1*11*1111	43
	90%	111111**1**111**11*1*1*1*11**1*11**11*1*1111	44
28	70%	11111*1*111*111**111*1*11*1*11*11**11111	40
	80%	111111**11*1**1*111**11**1*1*1*111**11*1*1111	45
	90%	11111*11*11***11*1*1*1**111***11*11**1*1*11111	46
29	70%	11111*1*111**111*1*11*11*1*111*11**111111	41
	80%	1111*1*11**11*111***1*11*11*1*1*11**111**11111	46
	90%	1111*1*11**11*11***111*11*1*11**1*1*1*11**11111	47
30	70%	11111*111*11*11*1*11**1111*1*1*111**111111	42
	80%	1111*11*1*11*1*1*1*1*11**1111**11*11***11*11111	47
	90%	111111***111***11*11*1*1**11*11**11*1*1*1*11*1111	49
35	70%	111111*11*111**1111*11*1*111*1*111**111*11*11111	48
	80%	1111*11*1*11*111**1*1*111*1**11*111***111**1*111*11111	54
	90%	111111*1*1*11*11**111***111*11**111*1**1*11**11*1*11111	55
40	70%	111111*111*1*1111*11*11*1111*111**1111*1*111*11*11111	53
	80%	11111*1*111**111**1111*1*1*11*1**11*111**11*11*11*1*1*111111	60
	90%	11111*1*1*1111***111**1*11*11**111*11*11**1*1*111**11*1*111111	62
46	70%	1111111*1111**11111*111*11*1*11111*111*11*1111*111*1*111111	59
	80%	111111*1*11*111**11*1*1111**1*111*1*11**11111**1*1*111**111*11*11111	68
	90%	111111**111**111*11*11*1*1*1*111**1*111**11*11*11*1*11**1111**1*111111	70
52	70%	1111111**1111*1111*1*1111*111*11111*11*11111*111*111*11*1111*11111	66
	80%	1111111*1*11*11**1111*11*1*1*11*111*11**1111*1**1111**111*1*111*1*11**111111	76
	90%	11111*111**11**1111***1111**11*1111**11*111*1*1*11*11*1*1*11*1*111*1**1*1111111	79
58	70%	111111*11111*111*111*11111*11*11111*11*1111*1111*1*1111*11111*111*111111	72
	80%	1111111***1111*1*111**11*111**111*1*11*1111*1*111*1*111*11*11**111*11*11*11*1*111111	84
	90%	111111*1**111*11*1*1111***1111**11*111*1*11*11**111**1*111*1*111*1*11**111*11*1*1111111	87
64	70%	111111*1111*11111*11111*1*111111*11*1111*1111*11111*11*111*11111*111*111*111111	79
	80%	111111*1*11*1111**1111*11*1*1111**111*11*1111***1111*111*1*1*111*11*1*111*11**11*11*1111111	91
	90%	1111111*1*111**1*1111*1*111**111**1111*1*11*11*11*111*1**1111*1**11*1111**11**11*111*1*1*111111	95

VIII. Conclusion and further research

We have introduced a new measure, overlap complexity, which is experimentally very well correlated with sensitivity. Our heuristic algorithms for computing overlap complexity enabled us to compute very long seeds about which we have good reasons to believe they have high sensitivity. Their sensitivity cannot be computed by the existing algorithms and maybe it will never be found since computing sensitivity is NP-hard. Therefore, different ways for inferring high sensitivity are needed.

While our experimental results are excellent, the theory to support them needs development. Problems include proving guarantees for the correlation between overlap complexity and sensitivity, finding bounds on the approximation ratio of our heuristic algorithm, and computing its running time. On the one hand, these theoretical questions are probably difficult to solve and they are not essential for the practical aspect of our study. On the other hand, they may bring new ideas to further improve our approach.

A closely related problem not investigated here is computing good multiple seeds [4], [14]. We expect our algorithm to work

as well in that case but it remains to be investigated.

REFERENCES

[1] S.F. Altschul et al., Basic local alignment search tool, *J. Mol. Biol.* **215** (1990) 403 – 410.

[2] S.F. Altschul et al., Gapped Blast and Psi-Blast: a new generation of protein database search programs, *Nucleic Acids Res.* **25** (1997) 3389 – 3402.

[3] B. Brejova, D. Brown, and T. Vinar, Optimal spaced seeds for homologous coding regions. *J. Bioinf. and Comp. Biol.* **1** (2004) 595 – 610.

[4] J. Buhler, U. Keich, and Y. Sun, Designing seeds for similarity search in genomic DNA, in: *Proc. of RECOMB'03*, ACM Press, 2003, 67 – 75.

[5] S. Burkhardt and J. Kärkkäinen, Better filtering with gapped q-grams *Proc. of CPM'01*, LNCS **2089**, Springer, 2001, 73 – 85.

[6] K.P. Choi, and L. Zhang, Sensitivity analysis and efficient method for identifying optimal spaced seeds, *J. Comput. Sys. Sci.* **68** (2004) 22 – 40.

[7] K.P. Choi, F. Zeng, and L. Zhang, Good Spaced Seeds for Homology Search, *Bioinformatics* **20** (2004) 1053 – 1059.

[8] R.M. Corless, *Essential Maple 7. An Introduction for Scientific Programmers*, Springer-Verlag, New York, 2002.

[9] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins Univ. Press, Baltimore, 1996.

[10] R. Karp and M.O. Rabin, Efficient randomized pattern-matching algorithms, *IBM J. Res. Develop.* **31** (1987) 249 – 260.

[11] U. Keich, M. Li, B. Ma, and J. Tromp, On spaced seeds for similarity search, *Discrete Appl. Math.* **3** (2004) 253 – 263.

[12] D. Kisman, M. Li, B. Ma, and L. Wang, tPatternHunter: Gapped, fast and sensitive translated homology search, *Bioinformatics* **21** (2005) 542 - 544.

[13] G. Kucherov, L. Noe, and Y. Ponty, Estimating seed sensitivity on homogeneous alignments, in: *Proc. IEEE 4th Symp. on Bioinformatics and Bioengineering*, Taiwan, 2004, 387 – 394.

[14] M. Li, B. Ma, D. Kisman, and J. Tromp, Pattern-HunterII: highly sensitive and fast homology search, *J. Bioinformatics and Comput. Biol.* **2** (2004) 417 – 440.

[15] D.J. Lipman and W.R. Pearson, Rapid and sensitive protein similarity searches, *Science* **227** (1985) 1435 – 1441.

[16] M. Li, B. Ma, and L. Zhang, Superiority and complexity of spaced seeds, in *Proc. of SODA'06*, SIAM, 2006, 444 – 453.

[17] B. Ma, J. Tromp, and M. Li, PatternHunter: faster and more sensitive homology search, *Bioinformatics* **18** (2002) 440 – 445.

[18] Maplesoft's MAPLE, `http://www.maplesoft.com/`.

[19] S.B. Needleman and C.D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, *J. Mol. Biol.* **48** (1970) 443 – 453.

[20] Z. Ning, A.J. Cox, and J.C. Mullikin, SSAHA: A fast search method for large DNA databases, *Genome Res.* **11** (2001) 1725 – 1729.

[21] L. Noé and G. Kucherov, Yass: enhancing the sensitivity of DNA similarity search, *Nucleic Acids Res.* **33** (2005).

[22] P. Pevzner and M.S. Waterman, Multiple filtration and approximate pattern matching, *Algorithmica* **13** (1995) 135 –154.

[23] F.P. Preparata, L. Zhang, and K.P. Choi, Quick, practical selection of effective seeds for homology search, *J. Comput. Biol.* **12** (2005) 1137 – 1152.

[24] V. Shoup's NTL: A Library for doing Number Theory, web site: `http://shoup.net/ntl/`.

[25] T.F. Smith and M.S. Waterman, Identification of common molecular subsequences, *J. Mol. Biol.* **147** (1981) 195 – 197.

[26] Y. Sun and J. Buhler, Designing multiple simultaneous seeds for DNA similarity search, in: *Proc. of RECOMB'04*, ACM Press, 2004, 76 – 85.

[27] J. Xu, D. Brown, M. Li, and B. Ma, Optimizing multiple spaced seeds for homology search, in: *Proc. of CPM'04*, Lectutre Notes in Comput. Sci. **3109**, Springer, 2004, 47 – 58.

[28] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller, A greedy algorithm for aligning DNA sequences, *J. Comput. Biol.* **7** (2000) 203 – 214.