BIOINFORMATICS

Vol. 00 no. 00 2007
pages 1-9

Sequence Analysis

Multiple spaced seeds for homology search

Lucian Ilie [1,*] and Silvana Ilie [2]

[1]Department of Computer Science, University of Western Ontario, N6A 5B7, London, Ontario, CANADA,
[2]Numerical Analysis, Centre for Mathematical Sciences, Lund University, Box 118, SE-221 00 Lund, SWEDEN

Received on XXXXXXXX; revised on XXXXXXXXXX; accepted on XXXXXXXXXXXX

Advance Access publication XXXXXXXXXXX

Associate Editor: XXXXXXXXXXXXX

ABSTRACT

Motivation: Homology search finds similar segments between two biological sequences, such as DNA or protein sequences. The introduction of optimal spaced seeds in PatternHunter, Ma et al. (2002), has increased both the sensitivity and the speed of homology search and it has been adopted by many alignment programs such as BLAST. With the further improvement provided by multiple spaced seeds in PatternHunterII, Li et al. (2004), Smith-Waterman sensitivity is approached at BLASTn speed. However, computing optimal multiple spaced seeds was proved to be NP-hard and current heuristic algorithms are all very slow (exponential).

Results: We give a simple algorithm which computes good multiple seeds in polynomial time. Due to a completely different approach, the difference with respect to the previous methods is dramatic. The multiple spaced seed of PatternHunterII, with 16 weight 11 seeds, Li et al. (2004), was computed in 12 days. It takes us 17 seconds to find a better one. Our approach changes the way of looking at multiple spaced seeds.

Contact: ilie@csd.uwo.ca

1 INTRODUCTION

Homology search finds similar segments between two biological sequences, such as DNA or protein sequences. A significant fraction of 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 Needleman and Wunsch (1970); Smith and Waterman (1981) became overwhelmed by the task, popular programs such as FASTA (Lipman and Pearson, 1985) and BLAST (Altschul et al., 1990) 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 short initial matches produced too many hits, thus decreasing speed.

A breakthrough came with PatternHunter (Ma et al., 2002) where the hits were no longer required to consist of consecutive matches. More 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 (Karp and Rabin, 1987; Pevzner and Waterman, 1995; Burkhardt and Kärkäinen, 2001) but the important novelty of PatternHunter was the use of optimal spaced seeds, that is, spaced seeds that have optimal sensitivity. Impressively, the approach of PatternHunter increases both the speed and sensitivity. The idea has been adopted since by the new versions of BLAST, MegaBLAST, BLASTZ, and other software programs (Brejova et al., 2004; Noé and Kucherov, 2005; Kisman et al., 2005).

As noticed in Ma et al. (2002), multiple spaced seeds—sets of seeds that hit whenever one of the components does so—are better, and with their introduction in PatternHunterII, Li et al. (2004), (Smith and Waterman, 1981) sensitivity is approached whereas the speed is that of BLASTn.

Quite a few papers have been written about spaced seeds, evaluating the advantages of spaced seeds over consecutive ones (Buhler et al., 2003; Keich et al., 2004; Choi and Zhang, 2004; Li et al., 2006), showing that the relevant computational problems are NP-hard (Li et al., 2004, 2006), giving exact (exponential) algorithms for computing sensitivity (Buhler et al., 2003; Li et al., 2004; Keich et al., 2004; Choi and Zhang, 2004; Choi et al., 2004), polynomial time approximation schemes (Li et al., 2006) or heuristic algorithms (Li et al., 2004; Choi et al., 2004; Yang. et al., 2004; Preparata et al., 2005; Ilie and Ilie, 2007; Kong, 2007), adapting the seeds for more specific biological tasks (Brejova et al., 2004; Kucherov et al., 2004; Sun and Buhler, 2004; Noé and Kucherov, 2005), or building models to understand the mechanism that makes spaced seeds powerful (Buhler et al., 2003; Sun and Buhler, 2004; Preparata et al., 2005).

Finding optimal (multiple) spaced seeds is NP-hard but even finding good ones is very difficult. Exhaustive search involves two exponential-time steps: (i) there are exponentially many seeds to be tried and (ii) computing the sensitivity of each takes exponential time as well. Several approaches (Buhler et al., 2003; Li et al., 2004; Keich et al., 2004) tried to deal with the latter exponential by approximating the sensitivity. For the former, the number of seeds to be considered has been reduced by various heuristics (Choi et al., 2004; Yang. et al., 2004; Preparata et al., 2005; Kong, 2007) but it remained exponential.

*To whom correspondence should be addressed.

The approach here is based on the overlaps between the hits of a multiple seed. A new measure, *overlap complexity*, is introduced and shown to be experimentally well correlated with sensitivity. Since the new measure is computable in (low) polynomial time, we shall use overlap complexity instead of sensitivity and this takes care of the exponential in (ii). A similar approach has been introduced in Ilie and Ilie (2007) for single seeds. Also, Yang. et al. (2004); Kong (2007) contain some other measures well correlated with sensitivity for multiple seeds. However, we take care also of the exponential at (i), that is, the exponential number of candidate seeds. We give a simple algorithm which improves quickly the overlap complexity of an initial multiple seed, thus providing a good multiple seed in polynomial time.

We provide some results showing the good correlation between overlap complexity and sensitivity for single seeds. Our polynomial-time algorithm produces single seeds of sensitivity very close to optimal. For the multiple seed case such comparison cannot be made since no optimal multiple seeds are known. We shall compare our multiple seeds with previous ones and show them to have better sensitivity while our algorithm is much faster. The most important test is to compare against the multiple seed implemented in PatternHunterII, which contains 16 weight 11 seeds. While it took Li et al. (2004) 12 days to compute this multiple seed, we obtain a better multiple seed in 17 seconds. The dramatic improvement is due to a completely different approach. As discussed in the last section, our approach allows looking at multiple seeds in a totally different way.

A number of problems remain to be investigated such as proving guarantees about the correlation between overlap complexity and sensitivity, approximation ratio and exact running time of our heuristic algorithm for approximating the overlap complexity. However, such problems may be mostly of theoretical interest as in practice our algorithms produce very good multiple seeds in very short time.

The paper is organized as follows. The next section formally introduces multiple spaced seed and all concepts needed later. Our new measure is introduced in Section 3. Section 4 shows good correlation between overlap complexity and sensitivity. Our polynomial-time algorithm for computing good multiple seeds is given in Section 5. In Section 6 we compute better seeds than all previous ones. We conclude with a brief discussion in Section 7. More seeds whose sensitivity is discussed in the text are provided in the Appendix.

The content of the paper can be read in several ways, according to the goal of the reader. First, we computed a number of multiple spaced seeds that are ready to be used. No understanding of our algorithm is necessary for that purpose. Second, our algorithm is simple and explained in detail for the reader interested in producing a more efficient implementation and/or modifying the algorithm in order to solve different problems, such as computing more specialized seeds. Finally, we provide explanation of the intuitive ideas behind our algorithm in order to provide the interested reader with in-depth understanding of our approach.

2 SPACED SEEDS

We start with some basic definitions. An alphabet is a finite nonempty set, denoted by A. The set of finite strings over A is denoted by A^*. For a string $x \in A^*$, the length of x is denoted by $|x|$. For $1 \leq i \leq |x|$, the ith letter of x is denoted by $x[i]$. If $u = xy$, for some $x, y \in A^*$, then x (y, resp.) is called a prefix (suffix, resp.) of u. For two strings u and v, an *overlap* between u and v is any string that is both a suffix of u and a prefix of v.

A *spaced seed* is any[1] string over the alphabet $\{1, *\}$; 1 stands for a 'match' and $*$ for a 'don't care' position. For a seed s, the *length* of s is $\ell = |s|$ and the *weight*, w, of s is the number of 1's in s. A *multiple spaced seed* S is any finite nonempty set of spaced seeds.

The quality of the spaced seeds is given by their sensitivity, which, intuitively, is a measure of their ability to detect similar segments between biological sequences; see Ma et al. (2002). This is done as follows. 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. Such a match is then extended using classical methods to a local alignment.

```
DNA seq. S1      A C G A G G C A C T G T A T G T A T A T C T A
DNA seq. S2      A G T A G G C A A T G C A T T T A A A T C T C
matches/mism.    = ≠ ≠ = = = = = ≠ = = ≠ = = ≠ = = ≠ = = = = ≠
Bernoulli seq. R 1 0 0 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 0
spaced seed s          1 1 1 * 1 * * 1 * 1 * * 1 1 * 1 1 1
```

Fig. 1. An example of a hit using PatternHunter's spaced seed. All 1's in the seed (the last row) must correspond to matches between the sequences. The spaced seed s hits the Bernoulli sequence R (ending) at the third position from the right.

However, in order to be able to compare spaced seeds and ultimately compute good ones, we need a precise mathematical setting. The above process will therefore be reformulated as follows, see Ma et al. (2002); Keich et al. (2004). 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 (corresponding to matches) and 0's (corresponding to 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.

We are now in the position to give a rigorous definition for sensitivity of a spaced seed. The *sensitivity* of a seed s is the probability that s hits R at or before position n; see Ma et al. (2002); Keich et al. (2004). Note that the sensitivity depends on both the similarity level p and the length of the random region n.

An intuitive explanation of the reason for which seeds have different sensitivities follows. Recall that the sensitivity of a seed is the probability of hitting a random region of a given length. For

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

two spaced seeds of the same weight, the expected number of hits is the same but their sensitivities need not be the same. This happens as the hits of one seed may appear more clustered. A good intuitive example is searching for the strings aaa and abc in a random text. For each occurrence of aaa, the chance of having another one sharing two letters with it is $1/26$ whereas starting afresh would require $(1/26)^3$. Therefore, the occurrences of abc are more evenly distributed and it is more likely to see first an abc in the text.

A multiple spaced seed hits a sequence R if and only if one of its seeds hits R. The sensitivity of a multiple spaced seed S is defined similarly, that is, the probability that at least one seed of S hits R at or before position n.

In the light of the tradeoff between search speed and sensitivity, it makes sense to consider only multiple seeds in which all seeds have the same weight (they may have different lengths).

3 OVERLAP COMPLEXITY

We introduce in this section our complexity measure, the overlap complexity, which will turn out to be well correlated with sensitivity but much easier to compute. Therefore, it will replace sensitivity in our computations. Before introducing it, we give some intuitive explanation why overlapping hits of a seed are undesirable.

The hits of a seed can overlap but overlapping hits will detect a single local alignment. An example showing such a situation is shown in Fig. 2.

```
hit of good seed             1 1 1 * 1 * * 1 * 1 * * 1 1 * 1 1 1
local alignment        0 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0
1st hit of bad seed          1 1 * 1 1 * 1 1 * 1 1 * 1 1 * 1
2nd hit of bad seed                1 1 * 1 1 * 1 1 * 1 1 * 1 1 * 1
3rd hit of bad seed                      1 1 * 1 1 * 1 1 * 1 1 * 1 1 * 1
```

Fig. 2. An example showing the intuition behind overlap complexity; a local alignment is detected by one hit of a good seed whereas a bad seed “wastes” three hits to detect the same alignment.

Therefore, the sensitivity of a seed is inversely proportional with the number of overlapping hits, since the expected number of hits is the same. Thus, good seeds should have a low number of overlapping hits. The definite proof that (non-uniformly) spaced seeds are better than consecutive seeds, due to Li et al. (2006), involves estimating the expected number of non-overlapping 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.

We shall define a measure that is independent of the similarity level p. Consider two seeds s_1 and s_2 and denote by $\sigma[i]$ the number of pairs of 1's aligned together when a copy of s_2 shifted by i positions is aligned against s_1. The shift i takes values from $1-|s_2|$ to $|s_1|-1$, where a negative shift means s_2 starts first. Precisely, if we denote

$$t_1 = *^{|s_2|-1} s_1 *^{|s_2|-1},$$
$$t_{2,i} = *^{|s_2|-1+i} s_2 *^{|s_1|-i-1}, \text{ for } 1-|s_2| \le i \le |s_1|-1,$$

then

$$\sigma[i] = \mathsf{card}\{j \mid 1 \le j \le |s_1| + 2|s_2| - 2, t_1[j] = t_{2,i}[j] = 1\}\ .$$

The *overlap complexity* for two seeds is defined as

$$\mathsf{OC}(s_1, s_2) = \sum_{i=1-|s_2|}^{|s_1|-1} 2^{\sigma[i]}.$$

An example is shown in Fig. 3. Note that the measure is symmetric, that is, $\mathsf{OC}(s_1, s_2) = \mathsf{OC}(s_2, s_1)$, for any seeds s_1 and s_2.

	shift i	$\sigma[i]$
`* * * 1 1 * * 1 * 1 * * *`		
`1 * 1 1 * * * * * * * * *`	-3	1
`* 1 * 1 1 * * * * * * * *`	-2	2
`* * 1 * 1 1 * * * * * * *`	-1	1
`* * * 1 * 1 1 * * * * * *`	0	1
`* * * * 1 * 1 1 * * * * *`	1	2
`* * * * * 1 * 1 1 * * * *`	2	1
`* * * * * * 1 * 1 1 * * *`	3	1
`* * * * * * * 1 * 1 1 * *`	4	2
`* * * * * * * * 1 * 1 1 *`	5	0
`* * * * * * * * * 1 * 1 1`	6	1

Fig. 3. An example of the overlap complexity of two seeds: $\mathsf{OC}(\mathtt{11{*}{*}1{*}1}, \mathtt{1{*}11}) = \sum_{i=-3}^{6} 2^{\sigma[i]} = 25$.

The definition of the overlap complexity deserves a few comments. Note that the “importance” (we should say “weight” but that would be confused with the weight of the seeds) of the number of pairs of 1's aligned together for each shift doubles with each pair of 1's. While this may look as a reasonably natural definition, there is a good intuitive reason behind it. For a shift i, denote by $\sigma_*[i]$ the number of 1's aligned against $*$'s and by $\sigma_{**}[i]$ the number of pairs of $*$'s aligned together. For our example, these arrays are given below:

i	-3	-2	-1	0	1	2	3	4	5	6
$\sigma_*[i]$	5	3	5	5	3	5	5	3	7	5
$\sigma_{**}[i]$	7	8	7	7	8	7	7	8	6	7

The number of overlapping hits (with shift i) is then proportional to $p^{\sigma_*[i]}(p^2+(1-p)^2)^{\sigma_{**}[i]}$, where p is the similarity level; $p^{\sigma_*[i]}$ is the probability that $\sigma_*[i]$ $*$'s take value 1 and $(p^2+(1-p)^2)^{\sigma_{**}[i]}$ is the probability that $\sigma_{**}[i]$ pairs of $*$'s take the same value, 0 or 1. Choosing $p = 0.5$ makes this quantity equal to $2^{-\sigma_*[i]-\sigma_{**}[i]}$. It is reasonable to assume that a fixed-size window in considered when evaluating the overlapping hits, that is, the sum $\sigma[i]+\sigma_*[i]+\sigma_{**}[i]$ is assumed to be a constant, say c; in our example $c = 13$. Then, the number of overlapping hits is proportional to $2^{-\sigma_*[i]-\sigma_{**}[i]} = 2^{\sigma[i]-c} = \frac{1}{2^c}2^{\sigma[i]}$. Since c is constant, this is proportional with $2^{\sigma[i]}$ which gives our definition of overlap complexity.

It is important to mention that the freedom to conveniently choose the value $p = 0.5$ is due to the fact that, even if the optimal seed may change with p (see Table 1 below), the sensitivity changes very little.

For a multiple seed $S = \{s_1, s_2, \ldots, s_k\}$, the overlap complexity is defined by:

$$\mathsf{OC}(S) = \sum_{1\le i\le j\le k} \mathsf{OC}(s_i, s_j).$$

Note that the overlap complexity is invariant with respect to the order of the seeds and reversal (assuming all seeds are reversed simultaneously). This is expected of any measure well correlated with sensitivity.

4 SENSITIVITY OF LOW-OVERLAP SEEDS

We show here that the overlap complexity is, experimentally, well correlated with sensitivity for single seeds. We consider in Table 1 the top sensitivity seeds of Choi et al. (2004) (that is, seeds with highest sensitivity among those with a given weight); their sensitivity ranks for similarity levels $65\%, 70\%, \ldots, 90\%$ are given in columns $2, 3, \ldots, 7$, respectively. As mentioned earlier, the top sensitivity seed may change with the similarity level p. For instance, the first line for weight 11 corresponds to PatternHunter's seed which is the best for similarity levels 65% and 70%, second best for 75%, 80%, and 85%, and only third best for 90%. However, the differences between the sensitivities of these top seeds for any of the similarity levels considered is very small, a crucial observation for our approach, which is independent of similarity level.

The last column of Table 1 gives the overlap complexity rank. In all cases, at least one top sensitivity seed is on top of the overlap complexity ranking. Note that the seeds with the lowest overlap complexity are on top of the overlap complexity ranking.

The opposite is shown in Table 2 where the highest sensitivity of the seeds with lowest overlap complexity is shown. (There may be several seeds with lowest overlap complexity.) Almost all differences are zero. The correlation between the two measures is remarkable.

We cannot make the same comparison for multiple spaced seeds since there are no optimal multiple spaced seeds known.

Table 1. The top sensitivity seeds from Choi et al. (2004); a '-' means not in top 20. The last column gives the overlap complexity rank. Only rankings are shown, not the seeds. Each line corresponds to one seed.

weight	sensitivity rank under a similarity level						overlap complexity rank
	65%	70%	75%	80%	85%	90%	
	1	1	1	1	1	1	1
9	2	2	2	2	2	2	1
	4	4	4	4	4	4	1
	1	1	1	1	1	1	1
10	2	2	4	6	8	9	1
	8	6	2	2	2	5	1
	1	1	2	2	2	3	1
11	2	2	1	1	1	1	2
	6	3	3	5	5	6	2
	1	1	1	1	1	1	1
12	2	2	2	5	3	2	2
	6	3	3	2	4	4	1
	1	3	7	-	-	-	2
13	2	1	1	2	2	2	1
	7	2	2	1	1	1	6
	1	3	7	-	-	-	1
14	2	1	1	1	1	1	1
	5	2	2	3	3	6	1
	1	2	-	-	-	-	4
15	14	1	2	5	5	4	39
	-	5	1	1	1	1	1
	1	9	-	-	-	-	11
16	7	1	2	6	13	20	1
	-	7	1	1	1	3	1
	-	-	5	2	2	1	26
	1	3	-	-	-	-	1
17	6	1	2	4	4	5	1
	-	-	1	1	1	1	2
	1	4	-	-	-	-	36
18	-	1	1	2	3	2	1
	-	-	4	3	1	1	142

5 A POLYNOMIAL-TIME ALGORITHM

The exact algorithms for computing sensitivity are all exponential, see Buhler et al. (2003); Keich et al. (2004); Choi and Zhang (2004), which is expected since the problem is NP-hard, Li et al. (2006). The one of Choi and Zhang (2004) runs in time $\mathcal{O}(n\ell 2^{2(\ell-w)})$, for seeds of length ℓ and weight w. The other two have running times $\mathcal{O}(n\ell^2 2^{\ell-w})$ for Keich et al. (2004) and $\mathcal{O}(nw2^{\ell-w})$ for Buhler et al. (2003).

For multiple seeds, Li et al. (2004) gave a dynamic programming algorithm that runs in time $\mathcal{O}\big((k+L+n)\sum_{i=1}^{k}\ell_i 2^{\ell_i-w}\big)$, where k is the number of seeds, ℓ_i's are the lengths of the seeds and $L = \max_{1\le i\le k}\ell_i$.

Therefore, finding optimal seeds by trying all seeds of a given weight (and length) and selecting the best is computationally very expensive. In fact, it has been shown by Li et al. (2004) to be NP-hard for an arbitrary distribution.

Some heuristic algorithms for computing good multiple seeds are presented in Yang. et al. (2004) and Kong (2007). As with our approach, they find some measures that are well correlated with similarity but they still need to consider exponentially many seeds. We shall compare our seeds with theirs in the next section.

The heuristic algorithm we derive from our overlap complexity is very simple: compute the seed with the lowest overlap complexity. This produces very good multiple seeds but we need to consider exponentially many candidates. To reduce the complexity of this step, we shall start with a fixed seed and repeatedly modify it to improve its overlap complexity. Each improvement consists of swapping a `1` with a $\star$ as long as the overlap complexity improves. Moreover, we greedily choose a swap that produces the greatest improvement. The number of such swaps in each seed will be bounded by the weight of the seeds.

We shall say that `1` *flipped* is $\star$ and vice versa. For a seed s and two positions i, j, we denote by $\mathsf{flip}(s, i, j)$ the seed obtained from s by flipping the letters in positions i and j. For instance, $\mathsf{flip}(\mathtt{1{\star}11{\star}11}, 3, 5) = \mathtt{1{\star}{\star}1111}$. With this notation, the algorithm MULTIPLESEEDS is described in Fig. 4. Remarkably, PatternHunter's seed is obtained by performing only 4 swaps in the algorithm MULTIPLESEEDS$(11, 18)$; see Fig. 5. This can be done by hand!

Let us discuss briefly our choice of the initial seeds in step 6. These are consecutive seeds and have very low sensitivity. One would imagine that starting from different seeds, e.g., random, would produce better results. Somewhat unexpectedly this does not seem to be the case and we preferred to keep a simple, deterministic, and ultimately reliable choice.

Concerning the swapping technique, it is trickier to give a good intuitive explanation. First, such swaps may change very

Table 2. The sensitivity of the top overlap complexity seeds for weights 9 to 18, similarity 70%, and length of random region 64.

weight	optimal sensitivity	sensitivity of a top overlap seed	difference to optimal
9	0.729156	0.729156	0.000000
10	0.595740	0.595740	0.000000
11	0.467122	0.467122	0.000000
12	0.356430	0.356430	0.000000
13	0.264750	0.264750	0.000000
14	0.193514	0.193514	0.000000
15	0.138660	0.138333	0.000327
16	0.098942	0.098942	0.000000
17	0.070004	0.070004	0.000000
18	0.049146	0.049146	0.000000

```
MULTIPLESEEDS($w, k$)
- given: the weight $w$ and the number of seeds $k$
- returns: a multiple seed $S$ with $k$ seeds of weight $w$ and high sensitivity
        // find the length of the seeds – half are equally spaced
        // in the interval $m..M$, the others have length $M$
 1.     $m = \text{round\_up}(\frac{4w}{3})$                  // shortest seed
 2.     $M = 25$                                         // longest seed
 3.     $h = \frac{2(M-m)}{k}$                                   // float
 4.     for $i$ from $1$ to $k$ do
 5.         $\ell_i \leftarrow \min(\text{round\_up}(m + i \times h), 25)$
 6.         $s_i \leftarrow \star^{\ell_i - w} 1^w$
 7.     $S \leftarrow \{s_1, s_2, \ldots, s_k\}$
        // swap 1's and $\star$'s to improve sensitivity
 8.     swaps $\leftarrow 0$
 9.     while $\Big((\exists\, r, i, j$ with $\mathsf{OC}(\{s_1, \ldots, s_{r-1}, \mathsf{flip}(s_r, i, j), s_{r+1},$
                $\ldots, s_k\}) < \mathsf{OC}(S))$ and (swaps $\le k \times w)\Big)$ do
10.         choose a triple $(r, i, j)$ that reduces $\mathsf{OC}(S)$ the most
11.         $S \leftarrow \{s_1, \ldots, s_{r-1}, \mathsf{flip}(s_r, i, j), s_{r+1}, \ldots, s_k\}$
12.         swaps $\leftarrow$ swaps $+ 1$
13.     return($S$)
```

Fig. 4. The MULTIPLESEEDS algorithm which, given the weight and lengths of the seeds, computes a multiple seed with low overlap complexity and, therefore, high sensitivity.

much the overlaps and thus have the potential of improving the overlap complexity. Second, any seed can be transformed into any other seed with the same length and weight using few such swaps and therefore we may potentially reach those seeds with very low overlap complexity we are looking for. Finally, and most convincingly, it works very well in practice.

It is possible that the swaps can be improved, or done differently. For instance, we did not perform more than one swap at the time as that would slow down the algorithm, even if it would remain polynomial.

Note that our whole approach with overlap complexity works within fixed length for seeds. When given only a fixed weight and number of seeds, a problem we need to solve is finding a good

intermediate seeds	pairs swapped
$\star\star\star\star\star\star\star11111111111$	$(1, 12)$
$1\star\star\star\star\star\star1111\star111111$	$(3, 15)$
$1\star1\star\star\star\star1111\star11\star111$	$(2, 9)$
$111\star\star\star\star1\star11\star11\star111$	$(5, 11)$
$111\star1\star\star1\star1\star\star11\star111$	

Fig. 5. Intermediate seeds computed by MULTIPLESEEDS$(11, 18)$ to find PatternHunter's seed $111{\star}1{\star\star}1{\star}1{\star\star}11{\star}111$. The flipped positions are given in the right column..

length set of the seeds. Trying all possible lengths is impractical. We came up with a simple but efficient choice, see steps 1 to 5 in the algorithm. Essentially, we set half of the lengths equal to 25 and the other half "equally" spaced between $\frac{4w}{3}$ and 25. (The code in lines 1 to 5 makes our choice precise.) The number 25 depends on the computer. Our tests were performed on a laptop with only 512 MB of RAM which prevented us from computing the sensitivity of longer seeds. We believe that the addition of longer seeds to some of our multiple seeds would increase the sensitivity but this needs to be tested.

Our choice of seed lengths turns out to be very good as we shall see below. However, for one seed we need to consider all lengths in an interval. In a few cases below we shall do the same for two seeds.

We have shown in the previous section good correlation between overlap complexity and sensitivity but now we compute an approximation of the overlap complexity. Still, the seeds we obtain have high sensitivity as shown in Table 3. We give also the time required for computing each seed.

Table 3. The sensitivity of the single spaced seeds computed by MULTIPLESEEDS compared to the optimal sensitivity for weights 9 to 18, similarity 70%, and length of random region 64. For each weight w, the best length in the interval $\frac{4w}{3} .. \frac{5w}{3}$ was chosen.

weight	optimal sensitivity	swap sensitivity	difference to optimal	time (sec)
9	0.729156	0.726279	0.002877	0.01
10	0.595740	0.594758	0.000981	0.01
11	0.467122	0.467122	0.000000	0.01
12	0.356430	0.354035	0.002395	0.04
13	0.264750	0.264512	0.000238	0.04
14	0.193514	0.192711	0.000803	0.09
15	0.138660	0.138333	0.000327	0.16
16	0.098942	0.098865	0.000076	0.17
17	0.070004	0.069874	0.000130	0.33
18	0.049146	0.048946	0.000200	0.58

Let us consider the time complexity of the MULTIPLESEEDS algorithm. Computing the lengths and initial seeds in steps 1 to 6 takes $\mathcal{O}(kw)$ time. To perform a swap, all possibilities for the triple (r, i, j) in step 9 are considered, that is, $\sum_{i=1}^{k} w(\ell_i - w)$. For each, we compute the new overlap complexity in $\mathcal{O}(\ell_r \sum_{i=1}^{k} \ell_i)$ time. (This is because the overlap complexity of two seeds is computed in time the product of their lengths and here we need only to update the pairs containing the seed s_r.) If we set $L = \max_{1 \le i \le k} \ell_i$, then the total time complexity of the MULTIPLESEEDS algorithm is $\mathcal{O}(k^3L^2w^2(L-w))$. If we assume that, in practice, k is bounded and L is linear in w, then it becomes $\mathcal{O}(w^5)$.

It may be useful to briefly summarize the steps of our approach to constructing multiple spaced seeds. Finding the optimal multiple spaced seed for a given weight and number of seeds involves two exponential stages: (i) there are exponentially many candidate seeds and (ii) computing the sensitivity of each requires exponential time as well. The exponential at (i) hides in fact two exponentials: (i.1) there are exponentially many lengths sets and (i.2) for each length set, there are exponentially many multiple seeds. First, we guess the length set (steps 1-5 of MULTIPLESEEDS); this takes care of the exponential at (i.1). Second, we start with some fixed values for the seeds (step 6), and this eliminates the exponential at (i.2). Finally, we repeatedly modify (polynomially many times) this multiple seed using overlap complexity that is computable in polynomial time as well. This way the exponential at (ii) is avoided. Instead of testing candidates we directly build a multiple spaced seed as required. This is totally different from previous approaches.

6 BETTER MULTIPLE SEEDS

We compare in this section our multiple seeds with the ones computed by other approaches. In Table 4 we compare our seeds with the best of Yang. et al. (2004) and Kong (2007). We picked the best multiple seed of Yang. et al. (2004) and compared it with ours for several similarity levels. The ones of Kong (2007) were computed for a specific similarity level and we give the sensitivity for those. Our seeds are better for all levels. Note that our method for choosing the length set in steps 1 to 5 of the algorithm MULTIPLESEEDS worked well even for two or three seeds. Only in the second last line the lengths given by it would produce a multiple seed of slightly lower sensitivity and we had to use an interval of lengths. This is still very fast.

Table 4. Comparing the seeds computed by MULTIPLESEEDS with previous multiple seeds; first group (lines 1 to 5): best of Yang. et al. (2004), 8 seeds, weight 12; second group (lines 6 and 7): best of Kong (2007), 3 seeds of weight 9; third group (lines 8 and 9): best of Kong (2007), 2 seeds of weight 11. The similarities for which Kong's seeds were computed are given in parentheses. For the second last line, we considered the interval 16..19 for lengths.

former multiple seeds	their sensitivity	sensitivity of our multiple seed	time (sec)
Yang. et al. (2004) 8 seeds of weight 12	0.287255 (60%)	0.313090 (60%)	2.50
	0.500277 (65%)	0.538023 (65%)	
	0.727770 (70%)	0 .765212 (70%)	
	0.897822 (75%)	0.920984 (75%)	
	0.977895 (80%)	0.985577 (80%)	
Kong (2007) 3 (2) seeds of weight 9 (11)	0.185211 (50%)	0.185472 (50%)	0.06
	0.972460 (75%)	0.977626 (75%)	0.05
	0.038393 (50%)	0.038554 (50%)	0.29
	0.815865 (75%)	0.823314 (75%)	0.02

The most difficult test is comparing with the multiple seeds of Li et al. (2004), the sensitivities of which were kindly provided by the authors (Li, 2007; Ma, 2007). The multiple seed of 16 weight 11 seeds in Li et al. (2004) — which is implemented in the best homology search software, PatternHunterII — took 12 days to compute greedily, that is, assuming the first i seeds are known, the $(i+1)$th seed is selected by exhaustive search in a length interval so that it maximizes the sensitivity of all $i+1$ seeds. Remarkably, MULTIPLESEEDS computes a better multiple seed in 17 seconds! It is shown in Table 5 and the comparison with the one of Li et al. (2004) is provided in columns two and three of Table 6. The last column of Table 6 contains the sensitivity (significantly higher) of a multiple seed consisting of 32 weight 11 seeds which we computed in less than 3 minutes. The multiple seed itself is given in the Appendix.

Table 5. Our multiple seed with 16 weight 11 seeds. It was computed in 17 seconds and it has higher sensitivity than PatternHunterII's multiple seed; see Table 6.

16 seeds of weight 11
{ 111⋆11⋆⋆11⋆1111,
111⋆1⋆11⋆⋆1⋆1111,
11⋆1⋆⋆11⋆1⋆1⋆⋆1111,
11111⋆⋆⋆1⋆⋆1⋆1⋆1⋆11,
111⋆1⋆11⋆1⋆⋆⋆1⋆⋆⋆111,
11⋆1⋆1⋆⋆⋆⋆111⋆⋆11⋆⋆11,
11⋆11⋆⋆1⋆⋆⋆⋆1⋆⋆1⋆⋆⋆1111,
111⋆⋆1⋆⋆11⋆⋆⋆⋆⋆1⋆1⋆⋆1⋆11,
1111⋆⋆⋆⋆1⋆⋆⋆1⋆1⋆⋆⋆⋆11⋆⋆11,
11⋆11⋆⋆⋆⋆1⋆⋆⋆⋆1⋆⋆1⋆⋆1⋆111,
111⋆⋆1⋆1⋆⋆⋆⋆1⋆⋆⋆⋆1⋆1⋆1⋆11,
11⋆11⋆⋆⋆1⋆1⋆⋆⋆⋆⋆⋆⋆11⋆1⋆11,
111⋆⋆11⋆⋆⋆⋆⋆⋆⋆11⋆⋆1⋆1⋆1⋆1,
111⋆⋆⋆1⋆1⋆⋆1⋆⋆⋆1⋆⋆⋆⋆⋆1111,
1⋆11⋆1⋆1⋆⋆⋆1⋆1⋆⋆⋆⋆⋆⋆11⋆11,
11⋆1⋆1⋆⋆⋆⋆1⋆⋆1⋆⋆11⋆⋆⋆⋆111 }

Table 6. The sensitivity of our multiple seed of weight 11 from Table 5 compared to that of Li et al. (2004) for length of random region 64.

similarity	sensitivity of the 16 seeds of Li et al. (2004)	sensitivity of our 16 seeds in Table 5	sensitivity of our 32 seeds (see Appendix)
60%	0.566640	0.578242	0.699776
65%	0.781508	0.792108	0.877349
70%	0.924114	0.930081	0.967602
75%	0.984289	0.986152	0.995271
80%	0.998449	0.998716	0.999702
85%	0.999951	0.999963	0.999995
90%	1.000000	1.000000	1.000000
time	12 days	17 sec	171 sec

We computed then, for the same weight 11, any number of seeds between 1 and 16 and compared their sensitivity for similarity level 70% with those of Li et al. (2004) in Table 7. We give also the time required by each computation. The multiple seeds are given in the Appendix. Note that the sensitivity of our multiple seeds with 13, 14, and 15 seeds are higher than the sensitivity of the ones of Li et al. (2004) with an extra seed, that is, 14, 15, and 16 seeds, respectively.

We should mention that the implementation of MULTIPLESEEDS is straightforward and we used the dynamic programming algorithm of Li et al. (2004) for computing sensitivity. The running times can probably be improved but our focus is on fast algorithms and not on efficient implementation.

Table 7. The sensitivity of our multiple seeds with i seeds, $1 \leq i \leq 16$, of weight 11, compared to that of Li et al. (2004) for similarity 70% and length of random region 64. The ones for $i \geq 3$ are computed by the algorithm MULTIPLESEEDS as given whereas for $i = 1, 2$ an interval for lengths was considered.

i number of seeds	sensitivity of the first i seeds of Li et al. (2004)	sensitivity of i seeds computed here	time to compute i seeds (sec)
1	0.467122	0.467122	0.01
2	0.620034	0.621992	0.88
3	0.701920	0.705694	0.09
4	0.754809	0.758224	0.20
5	0.791461	0.797473	0.52
6	0.818647	0.825245	0.82
7	0.839900	0.845990	1.33
8	0.856520	0.863893	1.96
9	0.870671	0.877309	2.95
10	0.882106	0.888385	3.89
11	0.891927	0.898855	5.40
12	0.900161	0.907064	7.50
13	0.907335	0.914018	9.87
14	0.913581	0.920340	11.68
15	0.919134	0.925966	16.17
16	0.924114	0.930081	17.26

7 CONCLUSION AND FURTHER RESEARCH

The introduction of optimal spaced seeds in Ma et al. (2002) followed by multiple spaced seeds in Li et al. (2004) revolutionized homology search. It is therefore important to compute good multiple spaced seeds fast. The optimal ones are hard to compute and research has been done for finding faster ways to compute less than optimal but still good seeds. Our approach is much faster and produces better multiple seeds than the existing ones. This was shown by comparing our results with the best previous ones.

We believe that the dramatic improvement brought by our approach allows looking at multiple seeds in a different way, beyond the improvement in homology search simply due to better multiple seeds. So far, as computing good multiple spaced seeds was a very time-consuming task, the seeds were first computed and then hard-coded in the homology search software. With our approach testing of many seeds for a given purpose becomes possible. Also, the swapping technique we used for fast improvement of overlap complexity may be useful for fast improvement of other, specific, properties as well.

While our experimental results are very good, 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 approximating the number of swaps needed. (The bound we set for the number of swaps in the algorithm was never reached in practice.) On one hand, these theoretical questions are not easy to solve and they are not essential for the practical aspect of our study; we simply build better multiple spaced seeds than all previous ones using a much faster algorithm. On the other hand, they may bring new ideas to further improve our approach.

From practical point of view, the best way of using the overlap complexity is an open problem and should be further investigated. Also the way the lengths are computed could be improved. As mentioned, this is computer dependent in our case. We plan to make more experiments on a computer with a larger RAM. However, all these improvement, important as they might be, are most likely to be incremental, nowhere near the dramatic improvement presented here.

Acknowledgements

We thank Ming Li and Bin Ma for kindly providing the sensitivities of their seeds in Li et al. (2004).

We also thank the anonymous referees for careful reading and useful suggestions that helped improving the presentation.

S.I. was supported by a postdoctoral fellowship from NSERC and L.I. was partially supported by an NSERC grant.

REFERENCES

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D., (1990), Basic local alignment search tool, *J. Mol. Biol.* 215, 403 – 410.

Altschul, S.F., Madden, T.L., Schaffer, 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 Res.* 25, 3389 – 3402.

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

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

Burkhardt, S., and Kärkkäinen, J., (2001), Better filtering with gapped q-grams *Proc. of CPM'01*, Lecture Notes in Comput. Sci. 2089, Springer, 73 – 85.

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

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

Ilie, L., and Ilie, S., (2007) Long spaced seeds for finding similarities between biological sequences, *Proc. of BIOCOMP'07*, to appear.

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

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

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

Kong, Y., (2007) Generalized correlation functions and their applications in selection of optimal multiple spaced seeds for homology search, *J. Comput. Biol.*, to appear.

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

Li, M., (2007) personal communication.

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

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

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

Ma, B., (2007) personal communication.

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

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, *J. Mol. Biol.* 48, 443 – 453.

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

Noé, L., and Kucherov, G., (2005), Yass: enhancing the sensitivity of DNA similarity search, *Nucleic Acids Res.* 33, 540 – 543.

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

Preparata, F.P., Zhang, L., and Choi, K.P., (2005), Quick, practical selection of effective seeds for homology search, *J. Comput. Biol.* 12, 137 – 1152.

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

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

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

Yang, I.-H., Wang, S.-H., Chen, H.-H., Huang, P.-H., and Chao, K.-M., (2004) Efficient methods for generating optimal single and multiple spaced seeds, *Proc. of IEEE 4th Symp. on Bioinformatics and Bioengineering*, Taiwan, 411 – 418.

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

APPENDIX

The multiple seed used in Table 4 for comparison with the one of Yang. et al. (2004):

```
111⋆1⋆11⋆1⋆11111
11⋆11⋆111⋆1⋆1⋆⋆111
11111⋆⋆⋆⋆1⋆⋆11⋆⋆1⋆111
11⋆⋆11⋆1⋆1⋆⋆⋆⋆⋆1⋆1⋆1111
111⋆1⋆1⋆⋆⋆⋆11⋆⋆⋆⋆1⋆⋆11⋆11
111⋆⋆11⋆⋆⋆⋆⋆⋆1⋆11⋆⋆1⋆1⋆11
111⋆1⋆⋆⋆11⋆⋆⋆⋆1⋆1⋆⋆1⋆⋆111
11⋆11⋆1⋆⋆⋆1⋆⋆1⋆⋆⋆11⋆⋆⋆111
```

The multiple seeds used in Table 4 for comparison with the ones of Kong (2007); they are given in the same order in which the sensitivities are given in Table 4:

```
11⋆1⋆111⋆111
111⋆1⋆⋆⋆⋆1⋆⋆⋆1⋆⋆⋆1⋆11
111⋆⋆⋆⋆1⋆⋆⋆⋆⋆1⋆⋆1⋆⋆⋆⋆1⋆11

11⋆1⋆111⋆111
111⋆1⋆⋆⋆⋆1⋆⋆⋆1⋆⋆⋆1⋆11
111⋆⋆⋆⋆1⋆⋆⋆⋆⋆1⋆⋆1⋆⋆⋆⋆1⋆11

11⋆1⋆11⋆⋆11⋆1111
111⋆11⋆⋆1⋆1⋆1⋆⋆111

111⋆111⋆⋆1⋆1111
11⋆1⋆1⋆⋆⋆11⋆⋆⋆⋆⋆⋆1⋆⋆1⋆111
```

Our multiple seed of 32 weight 11 seeds the sensitivity of which is given in the last column of Table 6; the time needed to compute all 32 seeds was 171 seconds:

```
111⋆11⋆11⋆⋆1111
11⋆1⋆111⋆11⋆⋆111
111⋆⋆11⋆1⋆11⋆111
1111⋆1⋆11⋆⋆1⋆1⋆11
111⋆⋆11⋆⋆1⋆⋆11⋆111
11⋆11⋆⋆⋆11⋆1⋆111⋆1
1111⋆1⋆⋆1⋆1⋆1⋆⋆⋆111
11⋆11⋆1⋆1⋆⋆⋆⋆1⋆1111
1111⋆⋆⋆⋆⋆1⋆⋆111⋆1⋆11
111⋆1⋆⋆⋆⋆⋆111⋆⋆⋆11⋆11
111⋆1⋆⋆11⋆⋆⋆⋆1⋆⋆1⋆111
111⋆1⋆1⋆⋆⋆1⋆1⋆⋆1⋆⋆1⋆11
11⋆1⋆⋆⋆11⋆1⋆⋆⋆⋆1⋆⋆⋆1111
11⋆11⋆⋆⋆⋆1⋆11⋆⋆⋆⋆⋆11⋆11
111⋆1⋆⋆1⋆⋆⋆⋆⋆1⋆1⋆⋆1⋆⋆111
111⋆1⋆1⋆⋆⋆1⋆⋆⋆⋆⋆11⋆⋆⋆111
11⋆1⋆⋆1⋆⋆⋆11⋆⋆⋆1⋆1⋆⋆⋆1⋆11
11⋆11⋆⋆⋆1⋆1⋆⋆⋆⋆⋆⋆1⋆1⋆⋆111
11⋆1⋆⋆1⋆⋆⋆1⋆⋆1⋆⋆⋆⋆11⋆⋆111
11⋆11⋆⋆⋆⋆1⋆⋆1⋆⋆⋆1⋆⋆⋆1⋆111
11⋆⋆11⋆⋆1⋆⋆1⋆⋆⋆⋆⋆⋆11⋆1⋆11
111⋆⋆1⋆⋆⋆⋆1⋆⋆⋆⋆11⋆⋆⋆1⋆111
11⋆11⋆⋆1⋆⋆⋆⋆⋆⋆1⋆1⋆1⋆⋆1⋆11
11⋆1⋆11⋆⋆⋆⋆⋆1⋆1⋆⋆⋆⋆11⋆⋆11
111⋆⋆⋆1⋆⋆1⋆1⋆⋆⋆⋆⋆1⋆1⋆⋆111
1111⋆⋆⋆⋆11⋆⋆⋆⋆⋆1⋆⋆1⋆1⋆⋆11
1⋆1⋆1⋆1⋆⋆1⋆⋆⋆1⋆⋆1⋆⋆1⋆1⋆11
111⋆⋆⋆⋆1⋆1⋆⋆⋆1⋆⋆⋆11⋆1⋆1⋆1
1⋆1⋆1⋆⋆11⋆⋆⋆1⋆⋆⋆⋆1⋆⋆11⋆11
111⋆⋆⋆1⋆1⋆⋆⋆1⋆1⋆⋆⋆1⋆⋆11⋆1
111⋆⋆1⋆1⋆⋆⋆⋆1⋆⋆1⋆⋆⋆1⋆1⋆11
11⋆⋆11⋆⋆⋆⋆1⋆⋆11⋆⋆⋆⋆⋆111⋆1
```

The seeds used to obtain the sensitivities in Table 7; the first is PatternHunter's seeds; the set of 16 seeds is given in Table 5:

1 seed
```
111⋆1⋆⋆1⋆1⋆⋆11⋆111
```
2 seeds
```
111⋆1⋆1⋆⋆1⋆11⋆111
1111⋆⋆⋆1⋆⋆1⋆⋆⋆1⋆1⋆111
```
3 seeds
```
111⋆111⋆⋆1⋆1111
11⋆11⋆1⋆⋆⋆⋆1⋆⋆⋆1⋆1⋆111
111⋆⋆⋆1⋆1⋆⋆1⋆⋆⋆1⋆⋆1⋆⋆1⋆11
```
4 seeds
```
111⋆111⋆⋆1⋆1111
11⋆11⋆1⋆⋆⋆⋆1⋆1⋆⋆1111
111⋆⋆⋆1⋆⋆⋆1⋆⋆⋆⋆1⋆11⋆⋆1⋆11
11⋆11⋆⋆⋆1⋆⋆⋆⋆⋆1⋆⋆⋆1⋆1⋆111
```
5 seeds
```
111⋆11⋆11⋆1⋆111
1111⋆⋆1⋆1⋆1⋆⋆⋆11⋆11
11⋆1⋆1⋆⋆⋆1⋆⋆1⋆⋆1⋆⋆⋆1111
11⋆1⋆11⋆⋆⋆⋆⋆⋆⋆1⋆1⋆⋆1⋆⋆111
111⋆1⋆⋆⋆1⋆⋆1⋆⋆⋆⋆⋆1⋆1⋆⋆111
```
6 seeds
```
111⋆11⋆11⋆1⋆111
11⋆1⋆1⋆⋆11⋆⋆1⋆1111
1111⋆⋆⋆1⋆⋆⋆1⋆1⋆⋆⋆11⋆11
111⋆⋆1⋆⋆⋆1⋆⋆⋆11⋆⋆⋆⋆1⋆1⋆11
111⋆1⋆1⋆⋆⋆⋆1⋆⋆1⋆⋆1⋆⋆⋆⋆111
11⋆11⋆⋆1⋆1⋆⋆⋆⋆⋆⋆1⋆1⋆⋆⋆111
```
7 seeds
```
111⋆11⋆1⋆1⋆1111
1111⋆⋆⋆11⋆⋆1⋆1⋆111
111⋆1⋆⋆⋆⋆11⋆1⋆⋆11⋆⋆11
1⋆111⋆⋆⋆1⋆1⋆⋆⋆1⋆⋆⋆⋆1⋆111
11⋆1⋆⋆11⋆⋆⋆⋆⋆⋆1⋆1⋆1⋆⋆1⋆11
111⋆⋆1⋆⋆⋆⋆⋆1⋆⋆1⋆⋆11⋆⋆⋆111
111⋆1⋆⋆⋆1⋆⋆⋆⋆1⋆⋆⋆⋆1⋆11⋆11
```
8 seeds
```
11⋆111⋆1⋆1⋆1111
1111⋆1⋆⋆1⋆⋆11⋆⋆111
11⋆11⋆⋆⋆11⋆⋆1⋆⋆1⋆111
111⋆⋆11⋆⋆⋆⋆⋆1⋆1⋆⋆⋆1⋆111
11⋆1⋆1⋆⋆⋆⋆11⋆⋆⋆⋆⋆⋆1⋆⋆1111
11⋆1⋆⋆1⋆⋆⋆1⋆1⋆⋆⋆⋆1⋆11⋆⋆11
111⋆1⋆⋆⋆⋆1⋆⋆⋆⋆11⋆⋆⋆1⋆11⋆1
111⋆⋆11⋆1⋆⋆⋆⋆⋆⋆⋆1⋆⋆1⋆1⋆11
```
9 seeds
```
111⋆11⋆1⋆1⋆1111
11⋆1⋆1111⋆⋆1⋆⋆111
11⋆11⋆⋆⋆⋆1⋆11⋆⋆1111
1111⋆⋆⋆1⋆⋆⋆1⋆1⋆1⋆⋆1⋆11
11⋆1⋆⋆⋆11⋆⋆⋆⋆1⋆⋆⋆1⋆11⋆11
11⋆1⋆1⋆⋆⋆1⋆⋆1⋆1⋆⋆⋆⋆1⋆⋆111
1⋆111⋆1⋆⋆⋆⋆⋆1⋆⋆⋆⋆1⋆⋆11⋆11
111⋆⋆1⋆⋆1⋆⋆1⋆⋆⋆⋆⋆1⋆⋆111⋆1
111⋆⋆⋆11⋆⋆⋆⋆⋆⋆1⋆1⋆1⋆⋆⋆111
```

10 seeds
```
111⋆1⋆11⋆111⋆11
1⋆111⋆⋆11⋆1⋆⋆1111
1111⋆⋆⋆⋆11⋆⋆11⋆1⋆11
111⋆11⋆⋆⋆⋆⋆1⋆1⋆⋆11⋆11
111⋆1⋆⋆1⋆⋆1⋆⋆⋆⋆⋆1⋆1⋆111
1111⋆⋆⋆1⋆⋆⋆1⋆⋆⋆1⋆⋆⋆1⋆11⋆1
11⋆⋆1⋆1⋆⋆1⋆⋆⋆⋆1⋆11⋆⋆⋆⋆111
11⋆1⋆1⋆1⋆⋆⋆1⋆⋆⋆⋆1⋆1⋆1⋆⋆11
11⋆1⋆⋆1⋆1⋆⋆⋆1⋆⋆⋆⋆⋆11⋆⋆111
111⋆⋆1⋆⋆⋆1⋆⋆1⋆1⋆⋆⋆⋆1⋆⋆111
```

11 seeds
```
111⋆11⋆1⋆1⋆1111
111⋆1⋆111⋆⋆⋆11⋆11
11⋆1⋆1⋆⋆1⋆⋆11⋆⋆1111
111⋆11⋆⋆1⋆1⋆⋆⋆1⋆1⋆11
111⋆⋆1⋆⋆⋆⋆11⋆⋆1⋆⋆11⋆11
111⋆1⋆⋆⋆⋆1⋆1⋆⋆⋆1⋆⋆⋆1⋆111
11⋆⋆11⋆⋆⋆1⋆⋆⋆⋆1⋆11⋆⋆⋆⋆111
11⋆11⋆⋆1⋆⋆1⋆⋆⋆1⋆⋆⋆⋆1⋆1⋆11
1⋆11⋆1⋆⋆1⋆⋆1⋆⋆⋆⋆⋆⋆11⋆⋆111
111⋆⋆⋆11⋆⋆⋆⋆⋆⋆1⋆⋆1⋆11⋆1⋆1
111⋆⋆⋆1⋆1⋆⋆⋆1⋆⋆⋆⋆1⋆⋆11⋆11
```

12 seeds
```
111⋆11⋆1⋆1⋆1111
111⋆1⋆111⋆⋆1⋆⋆111
111⋆11⋆⋆⋆⋆11⋆1⋆111
1111⋆⋆⋆1⋆⋆11⋆1⋆⋆1⋆11
11⋆1⋆11⋆⋆⋆⋆1⋆1⋆⋆⋆11⋆11
111⋆1⋆⋆⋆1⋆⋆⋆1⋆⋆1⋆⋆1⋆111
111⋆⋆⋆1⋆1⋆⋆1⋆⋆⋆1⋆⋆⋆⋆1⋆111
11⋆1⋆1⋆⋆⋆1⋆⋆1⋆⋆1⋆1⋆⋆⋆⋆111
11⋆⋆1⋆⋆1⋆⋆⋆1⋆1⋆⋆⋆⋆11⋆⋆111
1⋆11⋆⋆1⋆⋆1⋆⋆⋆⋆1⋆⋆⋆11⋆⋆111
11⋆⋆11⋆⋆⋆⋆1⋆⋆⋆1⋆⋆⋆1⋆11⋆11
111⋆1⋆⋆⋆11⋆⋆⋆⋆⋆⋆⋆1⋆1⋆1⋆11
```

13 seeds
```
111⋆11⋆11⋆1⋆111
11⋆11⋆1⋆1⋆1⋆⋆1111
111⋆1⋆1⋆⋆1⋆⋆11⋆111
111⋆11⋆⋆11⋆⋆⋆⋆1⋆1⋆11
1111⋆⋆1⋆⋆⋆1⋆1⋆1⋆⋆⋆111
11⋆11⋆⋆⋆⋆⋆⋆11⋆⋆⋆1⋆1⋆111
111⋆⋆⋆11⋆1⋆⋆⋆⋆⋆1⋆⋆1⋆1⋆11
111⋆⋆1⋆⋆1⋆1⋆⋆⋆⋆⋆⋆⋆1⋆⋆1111
1⋆111⋆⋆⋆⋆⋆1⋆⋆⋆1⋆⋆1⋆1⋆⋆111
11⋆⋆1⋆1⋆⋆⋆⋆1⋆⋆⋆11⋆⋆1⋆⋆111
11⋆1⋆⋆⋆⋆11⋆⋆⋆11⋆⋆1⋆⋆1⋆⋆11
1⋆11⋆1⋆⋆⋆1⋆⋆⋆⋆⋆11⋆⋆⋆11⋆11
111⋆1⋆1⋆⋆⋆⋆1⋆⋆1⋆⋆⋆⋆⋆11⋆11
```

14 seeds
```
111⋆11⋆1⋆⋆11111
1111⋆⋆11⋆1⋆1⋆111
11⋆1⋆111⋆⋆⋆⋆11⋆111
111⋆1⋆⋆⋆⋆11⋆⋆1⋆1111
11⋆11⋆⋆⋆1⋆11⋆⋆⋆1⋆1⋆11
111⋆⋆⋆1⋆⋆1⋆⋆1⋆1⋆⋆⋆1111
111⋆1⋆1⋆⋆⋆⋆⋆⋆1⋆11⋆⋆⋆⋆111
1⋆1⋆1⋆⋆1⋆1⋆⋆⋆⋆1⋆⋆11⋆1⋆⋆11
1111⋆⋆1⋆⋆⋆⋆1⋆⋆⋆⋆1⋆⋆11⋆⋆11
111⋆1⋆⋆⋆1⋆⋆⋆1⋆⋆⋆⋆1⋆1⋆⋆111
11⋆1⋆1⋆⋆1⋆⋆1⋆⋆1⋆⋆⋆1⋆⋆⋆111
11⋆⋆11⋆⋆1⋆1⋆⋆⋆⋆⋆1⋆⋆⋆11⋆11
111⋆⋆1⋆1⋆1⋆⋆⋆1⋆⋆⋆⋆⋆1⋆1⋆11
11⋆1⋆⋆1⋆⋆⋆1⋆⋆1⋆⋆⋆11⋆⋆1⋆11
```

15 seeds
```
111⋆⋆1⋆1111⋆111
11⋆111⋆1⋆1⋆⋆1111
1111⋆⋆11⋆1⋆1⋆1⋆⋆11
11⋆1⋆1⋆11⋆⋆1⋆⋆⋆1111
111⋆11⋆⋆⋆⋆11⋆⋆1⋆1⋆11
111⋆1⋆⋆⋆1⋆⋆1⋆1⋆⋆⋆1⋆111
1111⋆⋆⋆1⋆⋆⋆⋆1⋆⋆11⋆⋆1⋆11
11⋆1⋆11⋆⋆⋆⋆⋆⋆1⋆⋆⋆11⋆⋆111
111⋆1⋆⋆⋆⋆1⋆⋆⋆1⋆1⋆⋆1⋆⋆1⋆11
11⋆1⋆1⋆⋆1⋆⋆1⋆⋆1⋆⋆⋆⋆1⋆1⋆11
111⋆⋆1⋆⋆⋆11⋆⋆⋆⋆⋆⋆1⋆⋆1⋆111
11⋆11⋆⋆⋆⋆⋆⋆11⋆⋆⋆1⋆1⋆⋆⋆111
11⋆11⋆⋆1⋆⋆1⋆⋆⋆⋆⋆⋆1⋆1⋆1⋆11
1⋆11⋆⋆1⋆1⋆⋆⋆1⋆⋆⋆1⋆⋆1⋆⋆111
111⋆⋆⋆11⋆⋆⋆⋆1⋆1⋆⋆⋆⋆⋆11⋆11
```