BIOINFORMATICS APPLICATIONS NOTE
Vol. 25 no. 6 2009, pages 822–823
doi:10.1093/bioinformatics/btp054

Sequence analysis

Fast computation of neighbor seeds

Lucian Ilie[1,*] and Silvana Ilie[2]
[1]Department of Computer Science, University of Western Ontario, London N6A 5B7 and
[2]Department of Mathematics, Ryerson University, Toronto, M5B 2K3, Ontario, Canada

Received on December 01, 2008; revised on January 07, 2009; accepted on January 22, 2009
Advance Access publication January 28, 2009
Associate editor: John Quackenbush

ABSTRACT

Motivation: Alignment of biological sequences is one of the most frequently performed computer tasks. The current state of the art involves the use of (multiple) spaced seeds for producing high quality alignments. A particular important class is that of neighbor seeds which combine high sensitivity with reduced space requirements. Current algorithms for computing good neighbor seeds are very slow (exponential).

Results: We give a polynomial-time heuristic algorithm that computes better neighbor seeds than previous ones while being several orders of magnitude faster.

Contact: ilie@csd.uwo.ca

1 ALIGNMENTS AND SPACED SEEDS

Alignment of biological sequences is the most basic operation in computational biology. With the advances in the sequencing technologies, it becomes increasingly important to have fast, accurate alignment algorithms. Dynamic programming technics (Needleman and Wunsch, 1970; Smith and Waterman, 1981) could not handle long sequences and heuristic algorithms [FASTA (Lipman and Pearson, 1985), BLAST (Altschul *et al*., 1990)] were developed. BLAST uses a filtration technique in which short consecutive matches are identified first (*hits*) and then extended into local alignments. The next step [PatternHunter, Ma *et al*. (2002)] was to consider hits whose matches are no longer consecutive but according to a *spaced seed* such as `111*1**1*1**11*111`; `1` is a 'match', `*` a 'don't' care. It is intuitively clear that several spaced seeds have a higher chance of a hit. As a consequence, *multiple spaced seeds* were introduced in PatternHunter II (Li *et al*., 2004) and shown to reach almost perfect sensitivity. Their only drawback is high space requirement. This issue was recently addressed by Csűrös and Ma (2007) where a set of *neighbor seeds* were derived from the same parent seed. The neighbor seeds are 'close' to the parent so that their hits are easily computed from parent's hash table, thus saving space.

Quite a few papers have been written on (multiple) spaced seeds, concerning either algorithms for computing seeds with high sensitivity or adapting seeds for more specific biological tasks; see, for instance, the references of Ilie and Ilie (2007). We mention here only a significant application of neighbor seeds, due to Zhang *et al*. (2008), to obtain one of the best algorithms for the difficult multiple sequence alignment problem.

While various ideas have been employed, two aspects are common to all methods and applications. First, spaced seeds can improve significantly both the quality and the speed of alignments. Second, the algorithms to compute good seeds are slow. To our knowledge, the only polynomial-time heuristic algorithm to compute good multiple spaced seeds is due to Ilie and Ilie (2007).

In this note, we extend the approach of Ilie and Ilie (2007) to neighbor seeds. Similar to general multiple spaced seeds, we are able to compute better seeds much faster.

2 DEFINITIONS

A *spaced seed* is a finite 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* is a finite set of spaced seeds. The sensitivity of a seed is measured using the initial model of Ma *et al*. (2002). Let R be a Bernoulli random sequence of length n (the probability of a 1 in R is called *similarity*) and s a seed. 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. A multiple seed hits when at least one of its seeds does so. The *sensitivity* of a (multiple) seed is the probability that it hits R at or before position n.

In order to define neighbor seeds, we extend the definition of Hamming distance. Denote the original Hamming distance, between two strings of the same length, by d. For two strings $s_1,s_2\in\{0,1\}^*$, the *generalized Hamming distance*, denoted also by d, is defined as the minimum distance between all possible shifts of the two strings, which are assumed padded with `*`'s in order to have the same length. For instance, assume $\ell_1\geqslant\ell_2$, for $\ell_1=|s_1|$ and $\ell_2=|s_2|$. Then, $d(s_1,s_2)=\min_{0\leqslant i\leqslant\ell_1-\ell_2} d(s_1, {}^{*^i}s_2{}^{*^{\ell_1-\ell_2-i}})$.

Given a parent seed p, a seed s is a Δ-*neighbor* of p if $d(p,s)\leq 2\Delta$. The idea is to construct a multiple seed consisting of such neighbors of p which has high sensitivity.

3 THE ALGORITHM

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. For instance, if $s_1=$ `11**1*1` and $s_2=$ `1*11`, then the possible shifts are $-3,-2,\ldots,6$ and $\sigma[-3..6]=(1,2,1,1,2,1,1,2,0,1)$.

The *overlap complexity* (Ilie and Ilie, 2007) of two seeds is defined as $\mathsf{OC}(s_1,s_2)=\sum_{i=1-|s_2|}^{|s_1|-1}2^{\sigma[i]}$. For a multiple seed $S=\{s_1,s_2,\ldots,s_k\}$, the overlap complexity is defined by: $\mathsf{OC}(S)=\sum_{1\leq i\leq j\leq k}\mathsf{OC}(s_i,s_j)$. For the example above, we have $\mathsf{OC}($`11**1*1,1*11`$)=25$.

*To whom correspondence should be addressed.

NEIGHBORSEEDS(p, k, Δ)

1. **for** i **from** 1 **to** k **do**
2. $\quad s_i \leftarrow p*^{\lfloor i*\Delta/4 \rfloor}$ [copy p and extend the length]
3. $S \leftarrow \{s_1, s_2, \ldots, s_k\}$
4. swaps $\leftarrow 0$; $w \leftarrow$ WEIGHT(p)
5. **while** (swaps $\leqslant kw$) **do**
6. $\quad (r_0, i_0, j_0) \backslash\text{gets} \underset{d(p,\mathsf{swap}_{s_r}(i,j)) \leqslant 2\Delta}{\text{argmin}} \mathsf{OC}(\mathsf{swap}_S(r,i,j))$
7. $\quad$ **if** $(\mathsf{OC}(\mathsf{swap}_S(r_0, i_0, j_0)) < \mathsf{OC}(S))$
8. $\quad\quad$ **then** $S \leftarrow \mathsf{swap}_S(r_0, i_0, j_0)$
9. $\quad\quad$ **else return**(S)
10. $\quad$ swaps $\leftarrow$ swaps $+ 1$
11. **return**(S)

Fig. 1. The algorithm for computing k Δ-neighbors of p.

Table 1. Our eight neighbor seeds of weight 13 whose sensitivities are given in the last colum of Table 2

Parent	`1111*1**11**11*1*111`
neighbor seeds	`111*111*1***11*1*111`
	`111**1*1111**1*1*111`
	`11*1*1**11**1*11*1111`
	`1111*1***1*111*1*11*1`
	`1111*1**1***11*111*1*1`
	`1*11*1**11**11***11111`
	`1111*1**11***1*1**11*11`
	`1111***111**1**1*111**1`

As shown by (Ilie and Ilie, 2007), low overlap complexity and high sensitivity are very well correlated. Thus, the exponential-time computation of sensitivity will be replaced by a polynomial-time computation of overlap complexity.

We need to eliminate testing exponentially many candidate seeds as well. We start by introducing the following notation. For a seed s and two positions i,j, we denote by $\mathsf{swap}_s(i,j)$ the seed obtained from s by swapping the letters in positions i and j. Note that the swap operation preserves the weight of a seed. For instance, $\mathsf{swap}_{1*11*11}(3,5) = 1**1111$. For a multiple seed $S = \{s_1, \ldots, s_k\}$, we denote by $\mathsf{swap}_S(r,i,j)$ the multiple seed obtained from S by replacing s_r with $\mathsf{swap}_{s_r}(i,j)$.

In order to find good neighbor seeds, for a given weight w and a number of seeds k, we shall compute first a high sensitivity seed of weight w using the algorithm of (Ilie and Ilie, 2007). (Note that exponential-time algorithms are slow but feasible for single spaced seeds. The situation is completely different for multiple seeds.) The neighbor seeds are then computed as follows. Initially, they are all set to be the same as the parent seed. Then we consider all possible swaps that preserve the condition on the Hamming distance and choose the one that gives the greatest reduction in overlap complexity. The algorithm NEIGHBORSEEDS is described in Figure 1; p is the parent seed and k is the number of neighbor seeds within 2Δ distance from p.

The complexity of the algorithm NEIGHBORSEEDS is $\mathcal{O}(k^4 w^5)$ [the lengths of all seeds are $\Theta(w)$]; this is essentially $\mathcal{O}(w^5)$ since k can be assumed to be a constant.

Table 2. Our eight neighbor seeds of weight 13 compared with those of Csűrös and Ma (2007) ($\Delta = 2$); the length of the random region is 64

Similarity	Sensitivity	
	Csűrös and Ma (2007)	our seeds
60%	0.193032	0.193744
65%	0.379192	0.380516
70%	0.615959	0.617515
72.3%	0.630923	0.632470
75%	0.830499	0.831559
80%	0.955390	0.955725
85%	0.994852	0.994875
90%	0.999863	0.999863
Time	days (see text)	1.42 s

4 EXPERIMENTAL RESULTS

For eight neighbor seeds of weight 13 and $\Delta = 2$, our seeds are given in Table 1 and their sensitivity is compared to that of Csűrös and Ma (2007) in Table 2. We considered also the similarity level $72.3 \approx 47/65$ which is considered by Csűrös and Ma (2007). Our seeds are better at all levels. The time to compute the seeds in Csűrös and Ma (2007) was not provided. It is mentioned, however, that the greedy algorithm of Li *et al*. (2004) was used, which produced 16 seeds in 12 days. We can only infer that the seeds of Csűrös and Ma (2007) took days to compute, while ours took <2 s. Our algorithm computes more sensitive neighbor seeds while being many orders of magnitude faster.

5 CONCLUSION

In this article, as well as in Ilie and Ilie (2007), we gave algorithms that compute various types of spaced seeds, based on two main ideas: (i) overlap complexity replaces sensitivity and (ii) all exponential trials are avoided by swapping `1`'s and `*`'s. The arguments for using our ideas in computing all multiple spaced seeds, as used by a wide variety of alignment algorithms, are: (i) very simple implementation, (ii) improved sensitivity, and (iii) many orders of magnitude faster than all previous algorithms.

Conflict of Interest: none declared.

REFERENCES

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

Csűrös,M. and Ma,B. (2007) Rapid homology search with neighbor seeds. *Algorithmica*, **48**, 187–202.

Ilie,L. and Ilie,S. (2007) Multiple spaced seeds for homology search. *Bioinformatics*, **23**, 2969–2977.

Li,M. *et al*. (2004) Pattern-HunterII: highly sensitive and fast homology search. *J. Bioinform. Comput. Biol.*, **2**, 417–440.

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

Ma,B. *et al*. (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.

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

Zhang,Z. *et al*. (2008) MANGO: multiple alignment with *n* gapped oligos. *J. Bioinform. Comput. Biol.*, **6**, 521–541.