

# Indel seeds for homology search

Denise Mak<sup>1,\*</sup>, Yevgeniy Gelfand<sup>2</sup> and Gary Benson<sup>3,†</sup>

<sup>1</sup>Graduate Program in Bioinformatics, Boston University, Boston, MA 02215, <sup>2</sup>Lab for Biocomputing and Informatics, Boston University, Boston, MA 02215 and <sup>3</sup>Department of Computer Science, Department of Biology, Boston University, Boston, MA 02215

## ABSTRACT

We are interested in detecting homologous genomic DNA sequences with the goal of locating approximate inverted, interspersed, and tandem repeats. Standard search techniques start by detecting small matching parts, called *seeds*, between a query sequence and database sequences. Contiguous seed models have existed for many years. Recently, spaced seeds were shown to be more sensitive than contiguous seeds without increasing the random hit rate. To determine the superiority of one seed model over another, a model of homologous sequence alignment must be chosen. Previous studies evaluating spaced and contiguous seeds have assumed that matches and mismatches occur within these alignments, but not insertions and deletions (indels). This is perhaps appropriate when searching for protein coding sequences (<5% of the human genome), but is inappropriate when looking for repeats in the majority of genomic sequence where indels are common. In this paper, we assume a model of homologous sequence alignment which includes indels and we describe a new seed model, called *indel seeds*, which explicitly allows indels. We present a *waiting time* formula for computing the sensitivity of an indel seed and show that indel seeds significantly outperform contiguous and spaced seeds when homologies include indels. We discuss the practical aspect of using indel seeds and finally we present results from a search for inverted repeats in the dog genome using both indel and spaced seeds.

**Contact:** dyfmak@bu.edu

## 1 INTRODUCTION

Standard heuristic algorithms for homology search in biological sequences (Pearson and Lipman, 1988; Altschul *et al.*, 1990, 1997; Kent, 2002) utilize a two step approach. In the search step, short words from the query sequence, called *seeds* are paired with all matching words in sequences from a target database. The pairing can be done efficiently by first indexing the database words. In the confirmation step, each database match, called a *hit*, is tested to see if it is part of an extended region with homology to the query sequence. Hits are tested using alignment or some approximation to alignment.

An important element in successful homology detection programs is the choice of a good seed. A short seed increases the probability of finding a hit within a homologous region, but

also increases the number of random hits in non-homologous regions and thereby increases the running time. A long seed reduces the number of random hits, but also reduces the probability of hitting a homologous region. In practice, a trade-off is made between sensitivity (the probability of hitting a homologous region) and excessive running time caused by too many random hits. BLAT (Kent, 2002) for example, is well designed for DNA regions sharing very high identity ( $\geq 95\%$ ), allowing the use of long seeds or multiple seeds which provide both excellent sensitivity and very low probability of random hits.

In the last few years, much interest and research has focused on what are called *spaced seeds* (Ma *et al.*, 2002; Buhler and Sun, 2005; Keich *et al.*, 2004; Brejova *et al.*, 2004; Choi and Zhang, 2004; Choi *et al.*, 2004; Sun and Buhler, 2004; Xu *et al.*, 2004; Brejova *et al.*, 2005; Noe and Kucherov, 2005) because they increase sensitivity *without* simultaneously increasing the number of random hits. Patternhunter (Ma *et al.*, 2002) was the first general purpose program to utilize spaced seeds. While a standard contiguous seed is a short word or substring drawn from the query sequence, a spaced seed is a non-contiguous *subsequence*, that is, it consists of a number of explicit positions which must match separated by “don’t care” positions where the query and the homologous region may or may not match. A recent extension of this concept, implemented in YASS (Noe and Kucherov, 2005), is called a “transition constrained seed” and requires that mismatches in the “don’t care” positions must be of the transition type (A to G, or C to T) rather than transversions.

All the recent work on spaced seeds assumes that mutational differences in homologous regions consist solely of substitutions (mismatches) or that insertions and deletions (indels), if present, are widely spaced. While it can be argued that this is a valid assumption for designing seeds to find homologous protein coding regions, it is *not* valid for homology search in sequence where indels are tolerated, such as promoters and non-coding repeats. One of us (Benson) has been involved for several years in the development of software (Tandem Repeats Finder-TRF Benson, 1999 and Inverted Repeats Finder-IRF Warburton *et al.*, 2004) for the detection of approximate DNA repeats. These repeats usually occur within or contain non-coding sequence and along with substitution mutations, they exhibit numerous indel mutations as well. For example, in roughly 10% of the IRs found in the human genome using IRF (Warburton *et al.*, 2004), with arm lengths between 50 bp and 100 bp and arm separation below 500 000 bp, the frequency of indels between the left and right arms exceeds 8%. In a 50 bp repeat, this means at least 4 indel positions, with a limited possibility of wide spacing. This

\*To whom correspondence should be addressed.

†This research was supported in part by NSF grant DBI-0413462.

estimate of indel frequency is low since the detection method uses  $k$ -tuple matching (an alternate name for contiguous seeds). Switching to spaced seeds would improve the sensitivity of IRF, but developing a seed model that is sensitive to the existence of indels would improve the sensitivity even more.

In this paper, we introduce the idea of *indel seeds*. As with spaced seeds, an indel seed is a subsequence of the query which must match exactly to produce a hit and the separating “don’t care” positions may or may not match. The difference is that some separating positions can be of variable size to allow for insertions or deletions between the matching parts. In order to accommodate indels, we model alignments between homologous regions with a four character alphabet: {match, mismatch, insertion in query, insertion in target} and use a Markov chain to specify transition probabilities between these characters. We show how to compute the sensitivity of indel seeds under this model using a “waiting time” (Aki et al., 1984) calculation. We determine optimal seeds under several match/mismatch/indel configurations and show that indel seeds are more sensitive than contiguous or spaced seeds (with equivalent random hit rates) even when the indel frequency is one third the mismatch frequency. We discuss how to use indel seeds in practice, and finally, we present the results of searches for inverted repeat homologies in the dog genome using both indel and spaced seeds.

This paper is organized as follows. In section 2 we define our model of alignments for homologous regions that include insertions and deletions. In section 3, we define indel seeds. In section 4 we explain how indel seeds are used in practice and we discuss the random hit rate of indel seeds. In section 5 we show how to compute the sensitivity of indel seeds. In section 6 we compare the sensitivity of indel, contiguous, and spaced seeds. Finally, in section 7 we present the results of our search for inverted repeats in the dog genome.

## 2 HOMOLOGY MODEL

Two sequences are homologous if they have a common evolutionary ancestor. After the sequences diverge due to duplication or speciation, they typically undergo a variety of mutational events which, over time, transform them into different sequences. Depending on the evolutionary pressures, the sequences may change rapidly or remain similar over many millions of years. Homology detection tools exploit the remaining similarities to identify sequences that are homologous. Optimal seed selection is based on the homology model, which includes 1) the set of mutations that are presumed to transform the sequences, 2) the frequency of those mutations, and 3) the length of alignments of homologous regions.

A simple homology model and the one most frequently studied (Ma et al., 2002; Keich et al., 2004; Buhler and Sun, 2005; Xu et al., 2004; Choi and Zhang, 2004; Sun and Buhler, 2004; Brejova et al., 2005) allows only substitution mutations. In this model, alignments consist solely of matches and mismatches, and are represented by bit strings where a 1 indicates a match and a 0 indicates a mismatch or substitution. For example:

```

A C G T G C G T A A T T T C G
A C C A G C T T T A T T C C G
1 1 0 0 1 1 0 1 0 1 1 1 0 1 1

```

Two common variations assume 1) that the match and mismatch frequencies are independent and identically distributed (iid) across the alignment or 2) that the frequencies are the result of a Markov chain or hidden Markov model where every third position has a higher substitution frequency in order to reflect the higher variability in third codon position in protein coding gene sequences (Buhler and Sun, 2005; Brejova et al., 2004). Another model introduced by Noé and Kucherov (Noe and Kucherov, 2004) represents alignments with a ternary alphabet rather than bit strings in order to accommodate the higher frequency of transition substitutions in DNA evolution. In this case, one character represents a match, one a transition mismatch (A to G or T to C) and one a transversion mismatch (everything else).

**Indel Model.** Our new homology model includes insertion and deletion events as well as substitution. An alignment is represented by a string (hereafter called the *representative string*) over the following four character alphabet:

- 0 – mismatch;
- 1 – match;
- 2 – insertion into database sequence (deletion from query sequence);
- 3 – insertion into query sequence (deletion from database sequence).

For example:

```

Query      A C - G T G C G T A A T T T C G
Database   A C C G A G C - T - - T T T T G
Representative String 1 1 2 1 0 1 1 3 1 3 3 1 1 1 0 1

```

**Normalized alignment length.** In the match-mismatch homology model, the position  $x$  in a representative string (alignment) is the same as the position in the query. In the indel homology model, the alignment and the query will differ in length due to insertions into the database sequence. Since homology search proceeds from the query, we show (in Section 5) how to compute the probability of finding a seed hit *relative to the length of the query*. We therefore use the following notion:

**DEFINITION.** *The normalized length of a representative string of length  $n$  is the number of ‘0’, ‘1’, and ‘3’ characters contained in positions 1 to  $n$ . Normalized position  $k$  in a representative string is the position of the  $k$ th character from the set  $\{0, 1, 3\}$ , counting from the left. For example, the following representative strings all have normalized length 5 and in the middle string, pattern 11 occurs at normalized position 5:*

```
11101 10122211 1331221
```

From this definition, occurrence of the pattern 111 at normalized position 3 corresponds to an *infinite* number of representative strings because any number of 2’s can precede the pattern. This set of strings can be specified as  $2^* 111$ , where we use the regular expression  $2^*$  to denote zero or more 2’s.

**Representative string probabilities.** Mutation frequencies are described by a first-order Markov chain in order to avoid the occurrence of the character pairs “2 3” or “3 2” in a representative string. We exclude these pairs because consecutive indels, one in each sequence, are typically excluded by the choice of alignment parameters: the penalty for a single mismatch is usually less than the combined penalty for two individual indels. For the remainder of

this paper, we use the first-order Markov chain specified in the 4 x 4 transition matrix below:

		To:			
		0	1	2	3
From:	0	$p_0$	$p_1$	$p_g$	$p_g$
	1	$p_0$	$p_1$	$p_g$	$p_g$
	2	$p_0^*$	$p_1^*$	$p_g$	0
	3	$p_0^*$	$p_1^*$	0	$p_g$

where  $p_g$  stands for the probability of a gap symbol ('2' or '3') which we assume to be the same in the query and the database, and  $p_i^* = p_i + p_g (p_i / (p_0 + p_1))$  for  $i = 0, 1$  represents the proportional distribution of  $p_g$  to the characters 0 and 1. Other probability distributions are possible. We assume that the Markov process is stationary so that the probability of a given digit at any position reflects the equilibrium distribution  $\pi = (\pi_0, \pi_1, \pi_2, \pi_3)$ . However, the same is **not** true of any *normalized position*. In particular, this implies that the probability of a '1' at a given normalized position  $x$  depends on  $x$ . In actuality, the situation is simpler than that as the following theorem states. This fact is important for the sensitivity formula derived in Section 5.

**THEOREM 1.** *In a representative string, the probability of a '1' at the first normalized position is*

$$\pi_1 + \pi_2 \left( \frac{p_1}{p_0 + p_1} \right)$$

and at all other normalized positions is

$$p_1^* = p_1 + p_1 \left( \frac{p_g}{p_0 + p_1} \right)$$

(Proof omitted.)

### 3 SEED MODEL

The seed model defines characteristics of homologous alignments that will be recognized by the homology detection program. The model is specified in terms of the alphabet of representative strings and wildcard symbols. A *contiguous seed* is a string of 1's denoting successive matches in the alignment, for example 11111 or  $1^k$  where  $k$  in this case is 5. Ma *et al.* (Ma *et al.*, 2002) described a *spaced seed* which is a string, beginning and ending with a 1 and containing 1's and \*'s where \* is a wildcard that denotes either 1 (match) or 0 (mismatch) at that position. For example, 11\*\*11 indicates that a match or mismatch can occur at each of positions 3 and 4. Thus, 11\*\*11 actually specifies four distinct patterns in the representative string alphabet: 111111, 111011, 110111, 110011.

**DEFINITION.** *An indel seed is a string beginning and ending with a 1 and containing 1's, \*'s, and X's where 1 represents a match, \* is a wildcard that denotes either 1 (match) or 0 (mismatch), and X is a wildcard that indicates zero or one characters from the set {0, 1, 2, 3}, i.e., either a match, mismatch, insertion into the database or insertion into the query. Consecutive X's can represent any pair of numeric digits except "2 3" and "3 2".*

For example, 11XX1 with two wild-cards, permits indels of size zero, one, or two. This seed specifies the following 19 pattern strings:

111, 1101, 1111, 1121, 1131, 11001, 11111, 11221, 11331, 11011, 11021, 11031, 11101, 11121, 11131, 11201, 11211, 11301, 11311

As another example, the indel seed 1X1\*1 permits indels of size zero or one and specifies the following 10 pattern strings:

1101, 1111, 10101, 10111, 11101, 11111, 12101, 12111, 13101, 13111

### 4 USING INDEL SEEDS

In the following we outline how the indel seed 11XX11 is used in practice. The method generalizes to seeds with different numbers of indel positions. To process the query,  $Q$ , of length  $n$ , at a typical index  $i \in [6, n]$ , we extract *three* patterns because the size of the insertion in the query can be zero, one, or two:

$$Q[i - 5, i - 4]Q[i - 1, i], \quad Q[i - 4, i - 3]Q[i - 1, i],$$

and

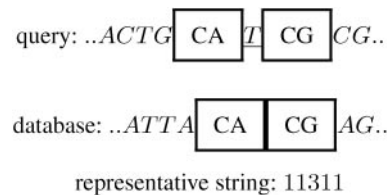
$$Q[i - 3, i - 2]Q[i - 1, i].$$

For example, if  $Q = \text{ACTGCATCGCG}\dots$ , then the patterns extracted at position 9 (underlined) are:



Note that the rightmost pair of characters is the same in all three patterns. The differences are in the position of the leftmost pair, separated by either two, one or zero characters from the rightmost pair. Because of edge effects, at query index  $i = 4$  there is one pattern to extract and at query index  $i = 5$  there are two. Each pattern is treated as a single four character string, in other words, *the spacing between the pairs is ignored*. We process the database sequences in the same way, again selecting three patterns for a typical index because the insertion in the database sequence can also be zero, one, or two.

**Determining what constitutes a match.** In the simplest approach, a match occurs whenever *any* pattern from a group of three in the query matches *any* pattern in a group of three from the database. The difference in spacing is not considered relevant for a match and is interpreted as an indel. For example:



**Random Hit Rate.** Because there are multiple chances to match for each query pattern, the random hit rate is based on the number of 1's in the seed and the number of comparisons. With the four-letter DNA alphabet, when we assume that each letter occurs with equal frequency, the probability of a random match is approximately

$$9 \cdot (1/4)^4$$

**Table 1.** Four possible comparisons between the query and database sequences in a restricted comparison strategy and the representative strings they detect

		Query patterns:		
		11	1 1	1_ 1
Database patterns:	11	-	131	-
	1_1	121	-	1301, 1311, 1031, 1131
	1__1	-	1201, 1211, 1021, 1121	-

The exponent 4 comes from the four characters in the query and database strings (the four 1's in the seed). The factor 9 comes from the fact that each query position has 3 patterns and each pattern has 3 chances of matching to a database position. The probability is actually somewhat less because the patterns at any one position are not independent (as in the example, the last two characters are the same).

**Restricted comparison strategy.** Note that there is flexibility in deciding which patterns from the query and database should be compared to detect matches and this flexibility can be used to reduce the random hit rate if the sensitivity of the seed model is not seriously impaired. For example, a more restrictive comparison approach would allow matching only between two patterns whose insertion spacing differs by one. This might be desirable when the seed is long and there is a high probability of finding an indel within the seed length. For this approach, the probability of a random match is

$$4 \cdot (1/4)^4$$

where the factor of 4 comes from the four possible comparisons between the three query patterns and three database patterns as in Table 1.

## 5 SEED SENSITIVITY

We calculate seed sensitivity using a *waiting time* formula (Aki et al., 1984), which is a general technique for calculating the probability of observing a given event, in a randomly generated string, by the occurrence of the  $k$ th character. Waiting time formulas are equivalent to other methods for calculating seed sensitivity (Keich et al., 2004; Buhler and Sun, 2005; Brejova et al., 2004; Choi and Zhang, 2004).

**Waiting Time Formula.** For clarity of presentation, the full version of this manuscript begins by deriving formulas for spaced seeds. Due to space restrictions, those formulas have been omitted. We assume that we are given an indel seed which specifies a set of patterns in the representative string alphabet. From this set, we eliminate all patterns that contain, as a substring, another pattern from the set. This can happen because an 'X' in the seed allows some of the patterns to be shorter than others. We work with this reduced set of patterns. The following terms are used:

- $T$ : The set of patterns specified by the seed, after eliminating those containing other patterns as substrings.
- $L_i$ : The *normalized length* of  $pattern_i$ .
- $w_i^*$ : The probability of  $pattern_i$  which is dependent on its normalized position. (By Theorem 1, that probability is the same everywhere except when the initial '1' occurs at the first normalized position.)
- $w_i[s]$ : The probability of the suffix of  $pattern_i$  following normalized position  $s$ ,  $s \in [1..L_i - 1]$  which is independent of its position.

- $V_i[s]$ : The set of patterns from  $T$  that can overlap the prefix of  $pattern_i$  when they occur at normalized position  $s$ ,  $s \in [1..L_i - 1]$ .
- $P(pattern_i : x)$ : The probability, across all representative strings, of  $pattern_i$  being the first occurrence of *any* pattern at *normalized* position  $x$ .
- $S(seed : x)$ : The *sensitivity* of the seed, *i.e.*, the *cumulative* probability, across all representative strings, of a first occurrence of any of the set of patterns in  $T$  at all *normalized* positions between 1 and  $x$ .  

$$S(seed : x) = \sum_{k=1}^x (\sum_{a \in T} P(pattern_a : k))$$

We are interested in calculating the probability of first occurrences of every  $pattern_i \in T$  at every normalized string position  $x$ , where first occurrence means a string ends with  $pattern_i$  at position  $x$  and the string prefix up to normalized position  $x - 1$  contains no other pattern in  $T$ . The seed sensitivity is the sum of these probabilities. The general scheme is as follows. For each  $pattern_i$ , and each normalized position  $x$ , calculate the probability of all representative strings with normalized length  $x$  that end with  $pattern_i$  (which is just  $w_i^*$ ). Then, subtract the probability of all strings in this set which contain an earlier first occurrence of any pattern in  $T$ .

To determine earlier first occurrences,  $pattern_i$  is compared with every other pattern (including itself) for overlapping and non-overlapping positions. All patterns are non-overlapping when they occur at  $k \leq x - L_i$ . The probability of strings which both end with  $pattern_i$  and contain one of these earlier occurrences is:

$$w_i^* \cdot \sum_{k=1}^{x-L_i} \left( \sum_{a \in T} P(pattern_a : k) \right) = w_i^* \cdot S(seed : x - L_i)$$

The following determines the probability of the initial 1 in  $pattern_i$ , and thus the value of  $w_i^*$ , when  $pattern_i$  follows an earlier occurrence of some pattern (rather than being the first occurrence of any pattern in the string).

**COROLLARY 2.** *The probability of a '1' at any normalized position following an earlier occurrence of any pattern is  $p_1^*$ .*

(Proof omitted.)

At each position  $j > x - L_i$  some subset of the patterns,  $V_i[L_i - (x - j)] \subseteq T$ , occurring at  $j$ , is consistent with an overlap of  $pattern_i$  at position  $x$ . The probability of these patterns is multiplied by the probability of the suffix of  $pattern_i$  that extends past the overlap:

$$\sum_{j=x-L_i+1}^{x-1} \left[ \left( \sum_{a \in V_i[s]} P(pattern_a : j) \right) w_i[s] \right]$$

where  $s = L_i - (x - j)$

The probability of a first occurrence of  $pattern_i$  at position  $x$  is then,

$$P(pattern_i : x) = w_i^* (1 - S(seed : x - L_i)) - \sum_{j=x-L_i+1}^{x-1} \left[ \left( \sum_{a \in V_i[s]} P(pattern_a : j) \right) w_i[s] \right] \quad (1)$$

where  $s = L_i - (x - j)$

**THEOREM 3.** *The time complexity for computing the cumulative sensitivity of an indel seed,  $S(seed : n)$ , is  $O(nl |T|)$  where  $n$  is the*

length of the homology region,  $l$  is the length of the seed, and  $|T|$  is the number of strings in the set of patterns specified by the seed.

PROOF. Consider a single  $pattern_i$ . From the formula for  $P(pattern_i : x)$ , for every location between 1 and  $n$  in the representative string,  $pattern_i$  computes one add and one multiply for the term  $w_i^*(1 - S(seed : x - L_i))$  and  $L_i - 1$  multiplications in the term  $\sum_{j=x-L_i+1}^{x-1} [(\sum_{a \in V_i[s]} P(pattern_a : j))w_i[s]]$ , where  $L_i \leq l$  and  $s = L_i - (x - j)$ . It suffices to show that the summation terms,  $\sum_{a \in V_i[s]} P(pattern_a : j)$ , each of which represents the probability for an overlap set,  $V_i[s]$ , can all be precomputed in time proportional to the number of patterns  $|T|$ . We sketch the proof here.

An Aho-Corasick tree (AC tree) of all patterns (constructed in time  $O(l|T|)$ ) can be used to determine the overlap set for every position in any  $pattern_i$ . The chain of failure links from the end of any  $pattern_j$  specifies the patterns and positions for which  $pattern_j$  belongs to an overlap set. Over the entire AC tree, these chains form another tree rooted at the single child of the AC root (indicating that a '1' has been read, since all patterns begin and end with a '1'). Each bifurcating node in this tree as well as each parent node of a leaf (if not bifurcating) represents a single overlap set. The number of these nodes is  $< 2|T|$  and they can be identified in a preprocessing step. For a given representative string position  $x$ , once the probabilities of all patterns at  $x$  have been computed, the probability of an overlap set at  $x$  (specified by a bifurcating or leaf parent node in the failure chain tree) can be computed by adding the probabilities of the node's children. This takes time  $O(|T|)$  for all overlap sets if the additions are computed in a post-order traversal. ■

For an indel seed, the number of patterns  $|T|$  depends on the number and location of the X's. An indel seed with a single X (like those we test in the next section) produces

$$4 \cdot 2^{*|} + 2^{*|} = 5 \cdot 2^{*|}$$

seeds where  $|*|$  is the number of match-mismatch wildcards in the seed. The first term specifies the seeds where 'X' holds a character and the second term where 'X' does not.

## 6 COMPARING SEED SENSITIVITIES

We compared the sensitivity of the three seed classes: contiguous, spaced, and indel. For the homology model, we chose several match/mismatch/indel probability configurations that reflect increasing ratios of indels to mismatches. These ratios are not uncommon in human IRs detected by the Inverted Repeats Finder in an exploration of that genome (Warburton *et al.*, 2004). The majority of those human IRs had lengths between 30 and 200 base pairs (bp). For our comparison, we chose two query lengths from this range: 64 and 100. The lower value was chosen to conform with earlier studies of spaced seeds.

For each model, we compared the sensitivity of seeds with equivalent random hit rates. For contiguous and spaced seeds, the random hit rate is  $(1/4)^k$  where the exponent, referred to here as *equivalent weight*, is the number of '1's in the seed, assuming equal probability of the letters in the DNA alphabet. For indel seeds, the random hit rate depends on the number and length of the indel positions in the seed as discussed in Section 4. For this analysis, we used indel seeds with a single X (which requires the selection of two strings at each position in the query and database sequences), so the

Table 2. Three possible comparisons under a restricted comparison strategy for indel seeds and the representative strings they detect

	Query patterns:	
	11	1_1
Database patterns:	11	131
	1_1	101, 111

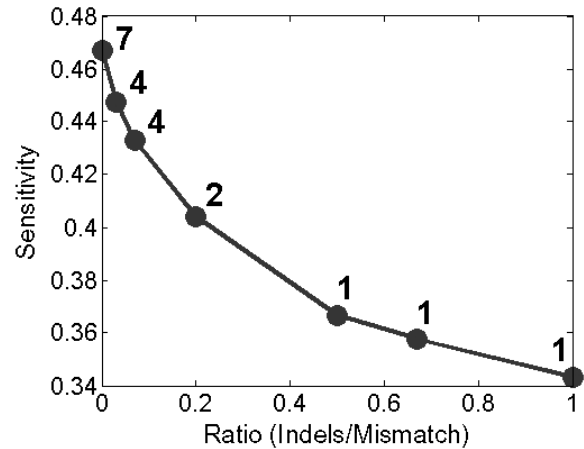


Fig. 1. Increasing the indel to mismatch ratio decreases the number of wildcards in the best spaced seeds (numbers next to data points) as well as the sensitivity. Match probability is 70%.

random hit rate is  $\sim 4 \cdot (1/4)^k = (1/4)^{k-1}$ , that is, the equivalent weight is *one less* than the number of '1's in the seed. For each homology model, we tested seeds with weights equivalent to 8, 9, 10, 11, and 12.

We also evaluated a restricted comparison strategy for indel seeds. Let the equivalent weight of an indel seed in the unrestricted comparison strategy, be its *nominal* weight. Our restricted comparison strategy allows three comparisons between the query and database at each location, as shown in Table 2. The true random hit rate is  $\sim 3 \cdot (1/4)^k$ , which is lower than the nominal rate. Out of necessity, we compared restricted comparison indel seeds with spaced seeds having a weight equivalent to the indel seed's nominal weight, *i.e.*, with spaced seeds that are expected to have more random hits.

**Results.** First, we note that contiguous seeds did more poorly than either spaced or indel seeds in almost all comparisons and their sensitivities are not shown.

Second, we observe two trends with spaced seeds: their sensitivity drops significantly as the ratio of indels to mismatches increases and the effectiveness of added match-mismatch wildcards also declines. Figure 1 graphs the sensitivity of the best spaced seed (equivalent weight = 11 and query length = 64) in several configurations, holding the match probability at 70% and raising the indel/mismatch ratio from zero to one. At each datapoint, the integer specifies the number of wildcards in the best seed. Across the range of ratios, sensitivity drops roughly 13% while there is a rapid decline from 7 wildcards in the PatternHunter seed at a ratio of zero, to 2 wildcards at a ratio of 0.2 and one wildcard at a ratio of 0.5. The trend suggests that a contiguous seed (zero wildcards)

**Table 3.** Sensitivities and optimal seeds at query lengths 64 and 100. Higher sensitivities are shown in bold font. Homology models are specified as (match, mismatch, database insert, query insert). In this table, match probability was held constant at 70%. Last number at top of each column is the ratio of indel probability to mismatch probability. W is equivalent weight, L is query length. Length 64 sensitivities marked with a star (\*) were obtained with a different seed than the one shown

Model	(70, 25, 2.5, 2.5); 0.2		(70, 20, 5, 5); 0.5		(70, 18, 6, 6); 0.66		(70, 15, 7.5, 7.5); 1		
	W	L Spaced seed	Indel seed	Spaced seed	Indel seed	Spaced seed	Indel seed	Spaced seed	Indel seed
8		<b>111*11*111</b>	11*11X1*1111	<b>11111*111</b>	11111X1111	111*11111	<b>11111X1111</b>	11111*111	<b>11111X1111</b>
64		<b>0.799248</b>	0.775944	<b>0.770976</b>	0.766357	0.760703	<b>0.772317</b>	0.743978	<b>0.781979</b>
100		<b>0.927726</b>	0.915175	<b>0.909198</b>	0.906417	0.902463	<b>0.910265</b>	0.891094	<b>0.916372</b>
9		<b>1111*11*111</b>	111*11X11*111	<b>11111*1111</b>	111111*1111	11111X1111	<b>11111X1111</b>	1111*11111	<b>11111X1111</b>
64		<b>0.666672</b>	0.645569	<b>0.628948</b>	0.625259	0.617831	<b>0.631623</b>	0.599931	<b>0.642072</b>
100		<b>0.836377</b>	0.821216	<b>0.802802</b>	0.800117	0.793025	<b>0.805638</b>	0.776882	<b>0.814576</b>
10		<b>11111*11*111</b>	111*11X11*1*111	<b>1111*111111</b>	<b>1111*111X1111</b>	111111*1111	<b>111111X11111</b>	111111*1111	<b>111111X11111</b>
64		<b>0.526124</b>	0.511339*	<b>0.488697</b>	0.487703*	0.477954	<b>0.493522</b>	0.460946	<b>0.503144</b>
100		<b>0.710387</b>	0.698711	0.669123	<b>0.670198</b>	0.657549	<b>0.674724</b>	0.638893	<b>0.684828</b>
11		<b>1111*111*1111</b>	1111*11X1111*111	111111*11111	<b>11111*111X1111</b>	111111*11111	<b>1111111X11111</b>	111111*11111	<b>1111111X11111</b>
64		<b>0.404111</b>	0.394377	0.366688	<b>0.368235</b>	0.357618	<b>0.371836</b>	0.343322	<b>0.379957</b>
100		<b>0.578892</b>	0.570552	0.531516	<b>0.536510</b>	0.520300	<b>0.538502</b>	0.502391	<b>0.548347</b>
12		<b>11111*111*1111</b>	1111*1*11X111*111	11111*111*1111	<b>111111X111*1111</b>	1111111*11111	<b>1111111X111111</b>	1111111*11111	<b>1111111X111111</b>
64		<b>0.301648</b>	0.294727*	0.268688	<b>0.271181</b>	0.260742	<b>0.273027</b>	0.249229	<b>0.279409</b>
100		<b>0.453702</b>	0.449161	0.409587	<b>0.413569</b>	0.396584	<b>0.413819</b>	0.380711	<b>0.422375</b>

**Table 4.** Sensitivities and optimal seeds at query lengths 64 and 100. Higher sensitivities are shown in bold font. Homology models are specified as (match, mismatch, database insert, query insert). In this table, match probabilities are higher than in table 3. Last number at top of each column is the ratio of indel probability to mismatch probability. W is equivalent weight, L is query length

Model	W	L	(75, 10, 7.5, 7.5); 1.5		(80, 10, 5, 5); 1		(85, 15, 2.5, 2.5); 0.33	
			Spaced seed	Indel seed	Spaced seed	Indel seed	Spaced seed	Indel seed
8			1111*1111	<b>1111X11111</b>	11111*111	<b>11111X1111</b>	<b>11111*111</b>	1111*11X111
64			0.876635	<b>0.922186</b>	0.968706	<b>0.980779</b>	<b>0.978886</b>	0.978114
100			0.966614	<b>0.984287</b>	0.996388	<b>0.998369</b>	<b>0.998096</b>	0.998057
9			1111*11111	<b>111111X1111</b>	1111*11111	<b>1111X111111</b>	<b>111*11*1111</b>	<b>111*11X11*111</b>
64			0.773066	<b>0.837862</b>	0.924164	<b>0.948027</b>	<b>0.943899</b>	0.943214
100			0.911442	<b>0.949127</b>	0.985147	<b>0.992035</b>	0.991157	<b>0.991239</b>
10			111111*1111	<b>11111X111111</b>	111111*1111	<b>11111X111111</b>	<b>11111*11*111</b>	<b>1111X111*1111</b>
64			0.652459	<b>0.729514</b>	0.855497	<b>0.892439</b>	<b>0.886191</b>	0.890837
100			0.824181	<b>0.883914</b>	0.958263	<b>0.974373</b>	0.972386	<b>0.974289</b>
11			111111*11111	<b>1111111X11111</b>	111111*11111	<b>11111X1111111</b>	1111*111*1111	<b>111*111X11*1111</b>
64			0.533202	<b>0.612365</b>	0.769472	<b>0.817654</b>	0.813606	<b>0.820006</b>
100			0.716664	<b>0.792340</b>	0.911418	<b>0.940129</b>	0.938616	<b>0.943307</b>
12			111111*1111111	<b>1111111X111111</b>	1111111*11111	<b>1111111X111111</b>	11111*111*1111	<b>1111*11X111*1111</b>
64			0.423409	<b>0.497833</b>	0.673449	<b>0.727906</b>	0.727865	<b>0.736757</b>
100			0.600568	<b>0.683390</b>	0.844450	<b>0.885424</b>	0.886638	<b>0.894515</b>

would eventually outperform a spaced seed. We observe this for match probability 75% and ratio 1.5 for equivalent weights 8 and 9 (not shown). Our intuition for this trend is the following. Spaced seeds can never match across an indel, so indel characters have the effect of chopping representative strings into smaller pieces. Within these smaller pieces, a longer spaced seed, with more wildcards, has fewer positions in which to match and so has lower sensitivity overall.

Third, with increasing indel to mismatch ratios, indel seeds outperform spaced seeds. Tables 3 and 4 give the sensitivities of the spaced and indel seeds in the homology models tested. In both tables, “winning seeds” and their sensitivities are shown in a

bold font. In Table 3 the match probability is held constant at 70%. When the indel to mismatch ratio is 0.2, spaced seeds are superior by around 1%. At ratio 0.5, indel seeds for the higher equivalent weights do better and at ratio 0.66 and above, indel seeds are clearly superior. For example, at ratio 1 and equivalent weights 10 and 11, the gain in sensitivity for the indel seed is 4.6%.

Table 4 shows a mix of match probabilities. Except for ratio 0.33 at the lowest equivalent weights, the indel seeds are superior. For example at 80% matching, ratio 1, and equivalent weight 11, the gain in sensitivity for the indel seed is almost 3%, and at 75% matching, ratio 1.5, it is 7.6%.

**Table 5. Sensitivity error correction.** Sensitivities of optimal seeds from tables 3 and 4 calculated on 100 000 pairs of sequence strings, generated according to the indicated homology model as described in the text, for query length 64. Higher sensitivities are shown in bold font. Sensitivity difference is the gain in sensitivity over the standard calculation. Homology models are specified as (match, mismatch, database insert, query insert). Last number at the top of each column is the ratio of indel probability to mismatch probability. W is equivalent weight, L is query length. The seed marked with a star (\*) is optimal at query length 64 in table 3 but is not shown in that table

W	Model L	(70, 20, 5, 5); 0.5		(70, 15, 7.5, 7.5); 1		(75, 10, 7.5, 7.5); 1.5	
		Spaced seed	Indel seed	Spaced seed	Indel seed	Spaced seed	Indel seed
10		1111*111111	<b>11111X11111*</b>	111111*1111	<b>111111X11111</b>	111111*1111	<b>11111X111111</b>
	64	0.53586	<b>0.55074</b>	0.53357	<b>0.59884</b>	0.72930	<b>0.81476</b>
	sensitivity difference	+0.047163	+0.063037	+0.072624	+0.095696	+0.076841	+0.085246
11		111111*11111	<b>11111*111X1111</b>	111111*11111	<b>1111111X11111</b>	111111*11111	<b>1111111X11111</b>
	64	0.40279	<b>0.43109</b>	0.40295	<b>0.46348</b>	0.61086	<b>0.70799</b>
	sensitivity difference	+0.036102	+0.062855	+0.059628	+0.083523	+0.077658	+0.095625
12		11111*111*1111	<b>111111X111*1111</b>	1111111*11111	<b>1111111X11111</b>	11111*1111111	<b>1111111X11111</b>
	64	0.30066	<b>0.32093</b>	0.29561	<b>0.34708</b>	0.48975	<b>0.58839</b>
	sensitivity difference	+0.031972	+0.049749	+0.046381	+0.067671	+0.066341	+0.090557

Finally, the restricted comparison strategy, while not as sensitive as unrestricted comparison, outperforms spaced seeds with *higher* random hit rates. At match probability 80%, ratio 1, and query length 100, the best restricted comparison indel seeds slightly outperformed the best spaced seeds at nominal weights 11 and 12 (data not shown). We observed the same results at match probability 75%, ratio 1.5, and query length 100 for nominal weights 9 through 12, with a gain of 3% in sensitivity at nominal weight 12.

**Seed sensitivity as a lower bound.** Seed sensitivity calculations actually underestimate the sensitivity of indel and spaced seeds when using an indel homology model for the following reason. When indels are allowed, sometimes more than one optimal alignment is possible. For example, the following are two optimal alignments for the same pair of sequences.

```

      Query  A T C T G G A T T G C
      Database A T A T - G A T T C C
Representative String 1 1 0 1 3 1 1 1 0 1

      Query  A T C T G G A T T G C
      Database A T A T G - A T T C C
Representative String 1 1 0 1 1 3 1 1 0 1
    
```

Note that in the first alignment, the indel seed, 11X111, does not occur, but in the second alignment it does. Thus we can not always classify the representative string from the first alignment as excluding the seed 11x111. Note also that for some alignments that do not contain a seed, there may be suboptimal alignments that do. For these reasons, our calculated values for indel and spaced seed sensitivities are lower bounds on the true sensitivities. We have estimated the error for several of the optimal seeds from Tables 3 and 4 using the following procedure.

We generated 100 000 random representative strings according to one of our Markov chain homology models, and then for each representative string, generated a pair of sequence strings that match the alignment. For the sequence strings, we assumed equal probabilities for the letters A, C, G, and T. We then exhaustively checked if any alignment, including any suboptimal alignment, of the two sequences contained the seed being tested. We analyzed three different homology models for query length 64.

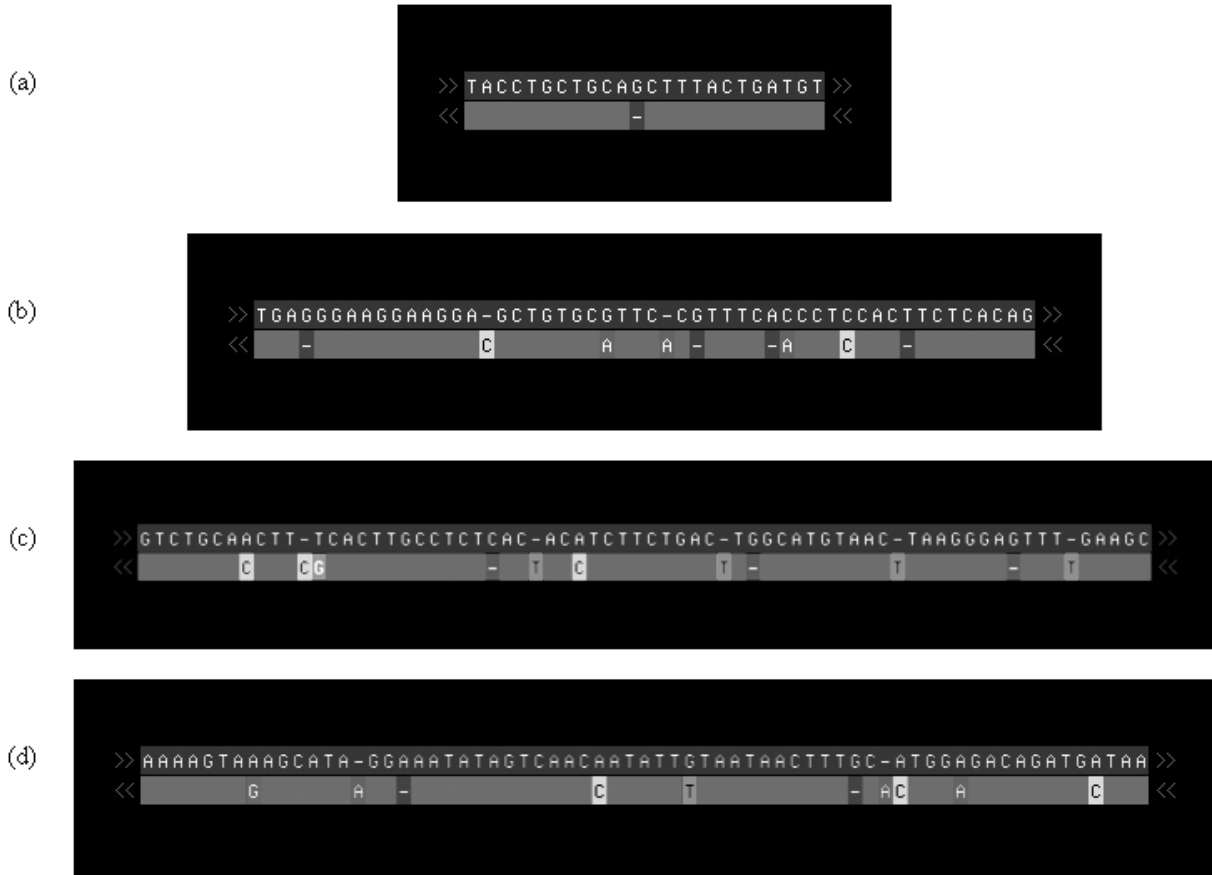
The seed sensitivities for the representative strings were nearly identical to our calculated values (data not shown). The seed sensitivities for the sequence pairs are shown in Table 5. We found a gain in sensitivity of ~ 3.1–9.5% for the seeds over the standard calculation, but in every case the gain was higher for the indel seed than the spaced seed.

## 7 INVERTED REPEAT SEARCH IN THE DOG GENOME

We tested the ability of indel and spaced seeds to find inverted repeats (IRs) in a modified version of the Inverted Repeats Finder (IRF) (Warburton *et al.*, 2004). The best seeds (equivalent weight = 12) were chosen from the homology model for 70% matching and indel to mismatch ratio = 1.0 (Table 3). The program was tested on the first 33 000 000 bases of chromosome I of the dog genome (Lindblad-Toh *et al.*, 2005). Typically when using IRF, we first mask known interspersed and tandem repeats so they are not reported as IRs. Interspersed repeats from the same family appear as IRs if they are coincidentally inserted in reverse orientation, tandem repeats from the same family act similarly. In addition, some common tandem repeats such as ATATATAT look like IRs. Sequence masked for interspersed repeats was obtained from the UCSC genome browser website (which uses RepeatMasker) and was additionally masked for tandem repeats using the Tandem Repeats Database (Benson, 2005). Roughly 39% of the sequence was masked and did not participate in IR detection. IRF scans through a sequence, recording seeds at each position by linking locations of identical seeds. The seed at the current forward-most position finds its ‘hits’ by scanning backward through the list of its reverse complement. A user specified maximum lookback distance sets the maximum spacer length between IR arms. For each hit, an alignment is computed and the program reports those alignments which score above a user specified minimum. Parameters used were: alignment scoring: match = 2, mismatch = -5, indel = -5; minimum alignment score = 40; lookback = 15,000,000. Testing was performed on a 2.8GHz PC with 2GB RAM. Table 6 shows the results.

**Table 6.** Results of IR search in dog genome using indel and spaced seeds

Seed	Repeats	Unique Repeats	Avg. Length (uniques)	Middle 50% Range (uniques)		Hits	Time (Minutes)
				Indel to Mismatch Ratio	Match %		
Spaced	21 365	228	56.6	0.2–1.1	82%–89%	38 717 975	23.7
Indel	21 752	614	34.3	1.0–3.1	89%–93%	41 063 612	28.6



**Fig. 2.** Examples of IRs found in the dog genome. In each part, upper is the left arm and lower is the right arm. Only mutational differences are shown in the right arm. a) typical of 53 repeats with a single indel found with the **indel** seed but not possible to find with the spaced seed, left arm 24 bp, arms 14.5 Mb apart, % match = 95.8, b) found uniquely by **indel** seed, left arm 50 bp, arms 7.6 Mb apart, % match = 82.7, indel to mismatch ratio = 2.0, c) found uniquely by **indel** seed, left arm 65 bp, arms 12 kb apart, % match = 84.2, ratio = 2.66, d) found uniquely by **spaced** seed, left arm 65 bp, arms 4.9 Mb apart, % match = 85.1, ratio = 0.66.

First notice that the indel and spaced seed are each able to find repeats that the other misses (the unique repeats column). Excluding these, both seeds found 21 138 repeats. Since the predicted sensitivity of these seeds is between 38% and 42% for the homology model used, we expect that many repeats were not found. Next, notice that the indel seed finds roughly 270% more unique repeats than the spaced seed. The unique repeats found by the indel seed are 1) significantly shorter as a group than the unique repeats found by the spaced seed, and 2) have higher indel to mismatch ratios (averages agree, but the middle 50% range is shown to better illustrate the variation). Surprisingly, the match percentage is generally higher for the indel seed than for the spaced seed. In fact, the spaced seed found no unique repeats with match percentage greater than 93% while the indel seed found 53. Each of the 53 looks like the one

shown in figure 2 (part a) with a single indel in the middle. These are impossible to find with the spaced seed. But it should be clear that there exist other repeats with an indel or mismatch offset from the center which were not found by either of the seeds tested here.

Finally, note that the indel seed produces 2 345 637 or ~ 6% more hits than the spaced seed. And because it does more alignments, the run with the indel seed took 20% longer. The higher number of hits does not appear to be the cause for the higher number of repeats detected by the indel seed (recall the 53 repeats described above). Rather, we believe the higher number of hits is caused by hit *clumping* within repeats which is magnified for the indel seed because multiple hits are examined at each sequence location. Two lines of evidence support this idea. First, runs on randomly generated sequence show less than 1% difference in the number of hits for

the seeds used here. Random sequences rarely contain IRs as long or longer than these seeds (13 and 14 characters) and so hit clumping is not expected to occur. Second, in a trial with the same dog sequence but a lookback of only 1 000 000, a similar increase in the indel hits was observed. Masking the IRs found and rerunning the sequence eliminated roughly 1/4 of the excess hits for the indel seed. To test if the remainder were due to IRs present but not reported, the minimum score was lowered to 30 which corresponds to repeats as small as 15 characters. Running the sequence, masking the IRs and rerunning eliminated the hit imbalance. These results suggest that indel seeds should be used in conjunction with hit filtering to avoid processing redundant clumped hits.

The IRs found in the dog sequence with arm separation  $> 500\,000$  bp (97% of those found) have not, to our knowledge, been reported before. Several examples found uniquely by the indel seed and the spaced seed are shown in Figure 2. The ones found by the indel seed are typical in that they tend to have higher indel to mismatch ratios than those tested in section 6.

## 8 CONCLUSION

We have presented a new seed model for homology search which explicitly allows indels. We give a waiting time formula for calculating indel seed sensitivity and show that indel seeds are superior to spaced and contiguous seeds in reasonable homology models which include indels. We discuss how indel seeds are used in practice and present the results of a limited search for inverted repeats in the dog genome using indel and spaced seeds with equivalent random hit rates.

## REFERENCES

Aki, S., Kuboki, H. and Hirano, K. (1984). On discrete distributions of order  $k$ . *Ann. Inst. Statist. Math.*, **36**, 431–440.

- Altschul, S., Gish, W., Miller, W., Myers, E. and Lipman, D. (1990). Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
- Altschul, S., Madden, T., Schäffer, A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, **25**, 3389–3402.
- Benson, G. (1999). Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Research*, **27**, 573–580.
- Benson, G. (2005). TRDB at <http://tandem.bu.edu/cgi-bin/trdb/trdb.exe>.
- Brejova, B., Brown, D. and Vinar, T. (2004). Optimal spaced seeds for homologous coding regions. *J. Bioinform. Comput. Biol.*, **1**, 595–610.
- Brejova, B., D. Brown, G. and Vinar, T. (2005). Vector seeds: an extension to spaced seeds. *Journal of Computer and System Sciences*, **70**(3), 364–380.
- Buhler, J. and Sun, Y. (2005). Designing seeds for similarity search in genomic DNA. *Journal of Computing and System Sciences*, **70**, 342–363.
- Choi, P.K. and Zhang, L. (2004). Sensitivity analysis and efficient method for identifying optimal spaced seeds. *Journal of Computer and System Sciences*, **68**, 22–40.
- Choi, K., Zeng, F. and Zhang, L. (2004). Good spaced seeds for homology search. *Bioinformatics*, **20**, 1053–1059.
- Keich, U., Li, M., Ma, B. and Tromp, J. (2004). On spaced seeds for similarity search. *Discrete Applied Mathematics*, **138**(3), 253–263.
- Kent, W. (2002). BLAT—the BLAST-like alignment tool. *Genome Res.*, **12**, 656–664.
- Lindblad-Toh, K. and *et al.* (2005). Genome sequence, comparative analysis and haplotype structure of the domestic dog. *Nature*, **438**, 803–819.
- Ma, B., Tromp, J. and Li, M. (2002). Patternhunter: faster and more sensitive homology search. *Bioinformatics*, **18**, 440–445.
- Noe, L. and Kucherov, G. (2004). Improved hit criteria for dna local alignment. *BMC Bioinformatics*, **5**, 149–158.
- Noe, L. and Kucherov, G. (2005). Yass: enhancing the sensitivity of DNA similarity search. *Nucleic Acids Research*, **33**, W540–W543.
- Pearson, W. and Lipman, D. (1988). Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, **85**, 2444–2448.
- Sun, Y. and Buhler, J. (2004). Designing multiple simultaneous seeds for dna similarity search. In *Proceedings, RECOMB*, pages 76–84.
- Warburton, P., Giordano, J., Cheung, F., Gelfand, Y. and Benson, G. (2004). Inverted repeat structure of the human genome: the X chromosome contains a preponderance of large highly homologous inverted repeats which contain testes genes. *Genome Res.*, pages 1861–1869.
- Xu, J., Brown, D., Li, M. and Ma, B. (2004). Optimizing multiple spaced seeds for homology search. In *Proceedings, Combinatorial Pattern Matching*.