Vector seeds: an extension to spaced seeds allows substantial improvements in sensitivity and specificity

Broňa Brejová, Daniel G. Brown, and Tomáš Vinař

School of Computer Science, University of Waterloo, Waterloo ON N2L 3G1 Canada
{bbrejova, browndg, tvinar}@math.uwaterloo.ca

Abstract. We present improved techniques for finding homologous regions in DNA and protein sequences. Our approach focuses on the core region of a local pairwise alignment; we suggest new ways to characterize these regions that allow marked improvements in both specificity and sensitivity over existing techniques for sequence alignment. For any such characterization, which we call a *vector seed*, we give an efficient algorithm that estimates the specificity and sensitivity of that seed under reasonable probabilistic models of sequence. We also characterize the probability of a match when an alignment is required to have multiple hits before it is detected. Our extensions fit well with existing approaches to sequence alignment, while still offering substantial improvement in runtime and sensitivity, particularly for the important problem of identifying matches between homologous coding DNA sequences.

1 Introduction

We study techniques for faster and more sensitive pairwise local alignment. Recent advances [10, 8, 5] have demonstrated modifications of the basic approach introduced in BLAST [1] that lead to significant improvements in both sensitivity and running time of local alignment. Here, we present a framework unifying and further extending these approaches, leading to even better performance.

The traditional approach for fast local alignment problem is represented by the BLASTN [1] program. BLASTN first identifies all pairs of short exact matches between the two sequences (hits). Every hit is then extended to a longer alignment, and alignments with high scores are reported, while those with low scores are discarded. High scoring alignments that do not contain a hit cannot be found by this approach.

Sensitivity can be increased by decreasing the required length of the hit; however, this also increases the number of spurious hits (decreasing *specificity*) and thus also increases the running time. Thus, there is a tradeoff between sensitivity and specificity induced by particular definition of a hit.

Recently, several researchers have reported alterations to the hit definition that improve sensitivity without decreasing specificity. Kent [8] in his program BLAT allows a *fixed number of mismatches* in the region that makes up a hit. For example, we may require at least 11 matches in a region of length 12.

PatternHunter [10] uses so called *spaced seeds*, which allow arbitrary numbers of mismatches in fixed positions of the hit. For example, a region of length 9 of an alignment is a hit to the PatternHunter seed 110110001 if there is a match on the first, second, fourth, fifth and ninth positions of the region. The other positions (the zeros in the seed) are not relevant.

Ma et al. [10] also introduce the idea of *optimizing the seed*, i.e. choosing the seed with given specificity which has the highest sensitivity under a given probabilistic model of alignment (region with 70% similarity of length 64). In follow-up work, more realistic probabilistic models of alignments (Markov chains, hidden Markov models) have been shown to yield optimal seeds with better performance on real data [4, 5]. (For our purpose, the performance of a seed is its sensitivity and specificity.) These methods also allow to create seeds tailored for particular application such as finding homologous protein coding regions.

Protein alignments are scored by substitution matrices such as BLOSUM62 [6] that define different scores for different matching and mismatching amino acid pairs. Therefore techniques considering only matches and mismatches for finding hits do not work very well. BLASTP [1] defines a hit as several consecutive positions with total score exceeding a given *threshold*.

These techniques are often supplemented by requiring *two non-overlapping hits* that are evenly spaced in both sequences. This method also increases sensitivity compared to a single stronger hit [2].

Here, we generalize these approaches into a new model of *vector seeds* (Section 2). Our model is general, allowing us to produce seeds for nucleotide alignments that incorporate positions that are required to match, positions that are free to vary, sets of positions that allow for a maximum number of variation, and more. Moreover, vector seeds allow easy transfer of techniques developed specifically for match/mismatch models into models based on scoring matrices. For example, we can apply spaced seed ideas to protein homology search.

We also provide an algorithm to predict the sensitivity of a vector seed, given a probabilistic model of alignments (Section 3). Our algorithm extends the original algorithm used to compute the sensitivity of PatternHunter's spaced seeds [7] to the case of vector seeds. We have previously extended this algorithm to more realistic probabilistic models of alignments, such as HMMs [4], and the sensitivity of our new seed models can also be computed for HMMs. This algorithm can be used to find the seed with the best predicted sensitivity for a given family of vector seeds. We further extend the algorithm to allow it to predict the sensitivity of two-hit alignment methods.

Our extensions universally allow greater sensitivity and specificity over existing pairwise alignment methods. For coding DNA alignments, we greatly improve over the performance of BLAT or PatternHunter seeds specifically chosen for this problem [4], allowing false positive rates several times smaller than previously existed, or offering large advantages in specificity with comparable sensitivity (Section 4). Our methods offer substantially improved performance over BLAT or PatternHunter, with minimal additional required changes.

2 Alignments and vector seeds

Vector seeds are a new way of defining a *hit*, the conserved part of an alignment that triggers alignment extension in the homology search program. A good definition of hit allows efficient identification of all hits in a sequence database and leads to high sensitivity and specificity. In this section we introduce vector seeds as well as probabilistic models for predicting their performance.

Vector seeds. To define a hit in the vector seed model, we represent ungapped pairwise local alignments as a sequence of real numbers, each corresponding to a position in the alignment. Alignments may potentially contain gaps. However, the hit must be located inside a single ungapped region of an alignment. Here, we model only individual ungapped fragments of such alignments.

In the simplest case, we represent pairwise alignments as binary sequences. Zero represents a mismatch and one a match. For protein alignments, we represent the alignment between sequences $Y = y_1y_2\dots y_n$ and $Z = z_1z_2\dots z_n$ by the sequence of positional scores, $(s_{y_1,z_1}, s_{y_2,z_2}, \dots, s_{y_n,z_n})$, where $S = (s_{i,j})$ is the scoring matrix. We call such sequence of positional scores an *alignment sequence.* Now we are ready to formally define a vector seed.

Definition 1. A vector seed *is an ordered pair $Q = (v, T)$, where v is the seed vector $(v_1, v_2, \dots, v_\ell)$ of real numbers and T is the seed threshold value.*

An alignment sequence $X = (x_1, x_2, \dots, x_n)$ hits the seed Q at position p *if $\sum_{i=1}^{\ell}(v_i \cdot x_{p+i-1}) \geq T$. That is, the dot product of the seed vector and the alignment sequence of length ℓ beginning at position p is at least the threshold T. The number of nonzero positions in the vector v is the* support *of the seed.*

Vector seeds generalize the spaced seeds of PatternHunter, the mismatching seeds of BLAT, and the minimum word score seeds used by BLASTP. For example, the BLAT seed that requires seven matching positions in nine consecutive positions in a nucleotide alignment is the vector seed $((1,1,1,1,1,1,1,1,1),7)$. The PatternHunter seed 110110001 can be represented as the vector seed $((1,1,0,1,1,0,0,0,1),5)$. The BLASTP rule that a hit is three consecutive positions having total score at least 13 corresponds to the vector seed $((1,1,1),13)$.

However, vector seeds can also encode more complicated concepts. For example, if the alignment sequence is binary, the vector seed $((1,2,0,1,2,0,1,2),8)$ requires matches in all positions with seed vector value of two, but allows one mismatch in the three positions with value one. The positions with value zero are not relevant to a hit. Or, if the alignment sequence is over the values $\{0,1,10\}$, then the seed $((10,10,1,10),301)$ matches either the alignment vector $(10,10,1,10)$ or the vector $(10,10,10,10)$, but no others.

Of course, more complicated vector seeds than these rather simple examples could be developed; the framework is general enough to allow vector seeds matching any half-space of R^ℓ. However, for simplicity, we will focus on a few families of vector seeds: for nucleotides, we will consider seed vectors with only zeros, ones and twos, where the total number of allowed mismatches is at most one or two, while for amino acids, we will consider short binary seed vectors.

Vector seeds are not universally expressive. For example, the seed $((1,1,0,1,1,0,1,1,0,1,1),8)$ corresponds to requiring matches in the first two positions of each of four consecutive codons. There is no way in the vector seed model to encode the requirement that three of the four codons are matched this way; the seed $((1,1,0,1,1,0,1,1,0,1,1),6)$ also allows one mismatch each in two codons.

Identifying hits in a sequence database. Assume we are given two sequences (or sequence databases) and want to find all hits between them. If hits are required to be exact matches of length k, the common approach is to create a hash table of all k-mers in one of the sequences and then search for each k-mer of the other sequence in the table. If hits are not exact matches (such as in BLAT or BLASTP), we can take each k-mer in the second sequence, generate a list of k-mers that would produce a hit and search for these k-mers in the hash table.

This approach extends to the vector seed scenario. Notice that we need to hash only characters on positions corresponding to non-zero elements in the vector seed. Hence, we seek vector seeds with small support that allow for a small number of hash table entries to be examined for each position in a query sequence. Otherwise, our results would not be of practical use!

In particular, if most examined hash table entries are expected to be empty, the time to find the false positives will dominate the time required to verify whether seed hits are false positives; this is undesirable. For example, all alignments of at least 50% identity and length at least 100 contain a hit to the vector seed with vector $(1,1,\ldots,1)$ of length 100 and with threshold 50. This seed will also be useless, as we would have to find matches to each 100 base pair sequence, and searching for such sequences could not be done efficiently.

Seed probabilities. So far, we have described how to use a single vector seed to generate a set of hits. However, given a family of seeds, we desire the seed that will perform best. To allow such optimization, we represent properties of alignments by a probabilistic model and search for a seed maximizing probability of at least one hit in an alignment sampled from the model. However, to control runtime, one must also control the false positive rate. Given two probabilistic models, one modeling alignments of unrelated sequences, and one for true alignments, we seek seeds with high sensitivity to true alignments, and low false positives.

Probabilistic models of true alignments. We model gap-free local alignments with probabilistic processes that generate sequences of real numbers. We investigate three models for local alignments.

The simplest is the model introduced by PatternHunter for nucleotide alignments, with alignments of length 100 and each position having probability 0.3 of being zero (mismatch) and 0.7 of being one (match), independently.

The other model we use for nucleotide alignments is a three-periodic model of alignments in protein coding regions, where each triplet is emitted as a unit, chosen from a probability distribution over $\{0,1\}^3$. Each triplet is independent of the others in this model. Such models can be used to effectively model the conservation in coding alignments, which are of key importance [9]. Recently [4], we have shown that the optimal spaced seed for coding alignments is quite different from the one that optimizes PatternHunter's model, and showed that this simple codon model is moderately effective at representing these alignments. To estimate seed sensitivity, we represent this probabilistic model as an 8-state hidden Markov model emitting individual binary characters.

For protein sequences, we represent the alignment as a sequence of BLOSUM62 scores ranging from -4 to 11, and we use a positionally independent model similar to PatternHunter's. An amino acid scoring matrix implies a probability distribution on pairs of residues being aligned in true alignments. In particular, an entry $s_{i,j}$ in a scoring matrix implies that the probability of residue i aligning with residue j in related sequences is approximately $b^{s_{i,j}}$ times the probability of them aligning by chance, for some base b [2], and we use this observation to compute a distribution on positional scores implied by the matrix.

Alignments are typically made up of more than one ungapped fragment, separated by gaps, and in our experiments, we treat the number and lengths of these fragments as random variables. Assume we know the probability $p_Q(\ell)$ that an ungapped alignment of a fixed length ℓ, generated by a probabilistic model M, has a match to a given vector seed Q. These probabilities can be used to compute the probability that a seed matches an alignment sequence whose length is a random variable L, by simply computing $P_Q(L) = \sum_\ell p_Q(\ell) \Pr[L = \ell]$. We have noted that alignments between homologous proteins usually consist of one long ungapped fragment and some number of shorter fragments. If the lengths of the long and short fragments are from known distributions L and S, and the number of short fragments is a random variable F, we can compute the probability that an alignment has any matches:

$$1 - (1 - P_Q(L)) \left(\sum_{n=0}^{\infty} (1 - P_Q(S))^n \Pr[F = n] \right).$$

Background model. To control the false positive rate, we need to be able to compute the probability p that a hit of a given seed occurs purely by chance at a given pair of unrelated sequence positions. The expected number of spurious hits is then pnm where n and m are the lengths of the two sequences.

For both nucleotide alignment models, our background probability distribution is a simple noise model, with zero emitted with probability 0.75, and one with probability 0.25. For protein alignments, we use the probabilities of the background distribution built into the BLOSUM62 matrix to determine the probability that residues are aligned by chance, and then use that to compute the score distribution.

Given this background probability, we can either use the algorithm given in the next section to compute the probability of a random match between two sequences, assuming that the sequence length is ℓ, or, if the seed is particularly simple, we can just compute it directly. For example, in the background model for nucleotide alignments, the probability of a $0/1$ seed with support s and threshold T having a match at a random place is just

$$\sum_{k=T}^{s} \binom{s}{k} 0.25^k 0.75^{s-k}.$$

3 Computing sensitivity for vector seeds

Here, we show how to compute the sensitivity of a vector seed to detect alignments that come from a position-independent alignment model. Our method is analogous to the original Keich et al. [7] algorithm, except that the alphabet has changed and need not be binary, and that the definition of a hit is the more complicated dot product property. In recent work [4], we show how to extend the original Keich et al. algorithm to the case where the alignment sequence is generated by a hidden Markov model. The extension to HMMs for vector seeds is straightforward, and we omit it for brevity.

Suppose that the probabilistic model generates an alignment sequence $X = (x_1, x_2, \ldots, x_n)$, and that the value in each position is chosen from a small finite set D of real numbers. For each $d \in D$, $\Pr[x_i = d] = p_d$, and all positions are independent. We seek the probability that sequence X generated by this process has a hit of a given vector seed $Q = (v, T)$, where $|v| = \ell$. Let D^* be the set of all sequences of numbers from D whose length is at most ℓ. In the analysis of our algorithm's runtime, we assume that we can represent vectors from D^* as integers and manipulate them in constant time. This assumption is reasonable in our experimental setting.

We compute the probability by dynamic programming, where the subproblem is the probability $P_Q(k, Z)$ that a sequence of length k, which begins with a given sequence Z from D^*, hits the seed Q. We are looking for $P_Q(n, \lambda)$, the probability of a hit when we make no conditions on the string, and the string is of length n. (We use λ to denote the empty sequence.)

We first identify sequences which are *guaranteed hits* and *possible hits*. Let F be a set of all sequences Z from D^* for which all extensions Z' to sequences from D^ℓ have $Z' \cdot v \geq T$. (That is, all extensions are a hit of the seed.) Let M be a set of all sequences Z from D^* for which there *exists* an extension Z' from D^l with $Z' \cdot v \geq T$. (Members of Z *can* be extended to seed hits.)

With these definitions, $P_Q(k, Z)$ can be computed as follows:

1. If $k < \ell$, then $P_Q(k, Z) = 0$.
2. Otherwise, if $Z \in F$, then $P_Q(k, Z) = 1$.
3. Otherwise, if $Z \notin M$, then $P_Q(k, Z) = P_Q(k-1, Y)$, where $Z = z_1 \ldots z_m$ and $Y = z_2 \ldots z_m$.
4. Otherwise, $P_Q(k, Z) = \sum_{d \in D} p_d P_Q(k, Z \cdot d)$.

The computation can be rearranged so that the third case is never reached, by always shifting forward enough positions when d is added to the end of Z in the fourth case. We move to the longest suffix of $Z \cdot d$ that is in M, and skip each instance of the third case. For each entry in M, and each value in D, this skip value can easily be computed in $O(l)$ time.

Sets F and M can be found in $O(|M|)$ time. We initially compute the best and worst possible score for each suffix of the seed Q. Initially, F is empty and M contains the empty string. In each iteration we choose one string X from M

and using the pre-computed scores we identify which elements d from D permit or require an extension of Xd to a sequence of length ℓ to hit the seed.

Thus, for an arbitrary vector seed Q, the algorithm computes $P_Q(n, \lambda)$ in $O(|M|(\ell + |D|n))$ time, if the entries of the dynamic programming table are stored in a data structure with $O(1)$ access. Note that the running time is dependent on the size of M; however, seeds with many matching strings are less useful in practice. Also, we need only keep the table P_Q for ℓ values of k, so the memory requirement is $O(|M|\ell)$.

Some algorithm extensions. We note four other fairly straightforward extensions. The first is to seed pairs, where two seeds, Q_1 and Q_2, must both have a hit in the sequence. We first compute the P_{Q_1} and P_{Q_2} matrices, and then, when we reach a prefix Z that is in the set F of guaranteed hits for either seed, we keep the prefix Z, and switch to requiring a hit to the other seed. The overall runtime is at most twice the runtime to compute P_{Q_1} and P_{Q_2}.

We can also compute the probabilities when a hit of either of the two seeds is required, or expand the set of matching strings in a variety of ways, by simply changing the sets M and F. For example, one can easily examine a single seed and the BLAST-style vector seed $(1^k, k)$, by simply adding the sequence 1^k to F and all of its $k-1$ prefixes to M.

We can also consider multi-hit models. Here, we require that the alignment contains at least p hits at least ℓ positions apart. This can be incorporated by keeping matrices $P_{Q,a}$, where $P_{Q,a}(k, Z)$ is the probability of at least a hits in a sequence of length k starting with Z. The recurrence is the same as before, except for the following modifications. First, $P_{Q,1}(k, X) = P_Q(k, X)$. Second, if $Z \in F$ and $a > 1$, then $P_{Q,a}(k, Z) = P_{Q,a-1}(k - \ell, \lambda)$. This is because if a sequence of length k starts with a hit, we need $a - 1$ hits in the rest of the sequence. We investigate the theoretical properties of two-hit BLAST and its variations, for both protein and nucleotide models. We also study two-hit vector seeds in general, comparing their sensitivity and specificity to one-hit models.

Finally, the probabilities p_d also need not be the same for all positions, as long as positions are independent. One can instead incorporate a position-specific probability distribution on D. This is equivalent to computing the sensitivity of a seed to a position-specific score matrix, if we assume that the true positives are generated with the probabilities implied by the score matrix. Of course, such scoring matrices are simply special cases of HMMs, which our algorithms can also expand to cover, using the techniques in our recent paper [4], but the algorithm is especially straightforward for these profiles.

4 Experiments

We performed four experiments to verify the usefulness of vector seeds. Our first two experiments investigate their predicted performance in the simple Pattern-Hunter model of alignments. In one case, we compute the best single seeds, and in the other case, we study pairs of seeds that join together well.

In our other experiments, we used models trained for DNA coding regions and for proteins, and we computed both theoretical performance of seeds and actual performance on real sequences.

In each experiment, we computed the probability of one hit and of two hits for the seeds we considered. We also computed the false positive probabilities for these seeds (assuming that two-hit models were satisfied when two matches occurred within 100 positions of each other). In experiments on real data, we also computed how many alignments from our set can be detected by each seed.

4.1 Predicted performance in the PatternHunter model

One hit required. First, we studied all vector seeds (v, T) with vector entries zero or one, support s satisfying $8 \leq s \leq 15$, threshold T satisfying $s-2 \leq T \leq s$, and whose length is at most $\min\{s+4, 17\}$. We have evaluated sensitivity of these seeds in a simple PatternHunter model.

Our results are summarized in Figure 1 and Table 1. Seeds with both permitted mismatches and the structure of spaced seeds have a large advantage over either alone. For example, the no-mismatch seed PH-10 has false negative rate 22.4% and false positive rate 9.54×10^{-7}. By contrast, the two-mismatch vector seed VS-13-15 has false negative rate 4.11% with false positive rate 9.23×10^{-7}.

This seed may not be practical, as there are 4^{15} possible hash table entries, but the more practical one mismatch seed VS-11-12 has false negative rate 4.9%, with twice the false positive rate (2.2×10^{-6}). This is to be compared to BLAT-11-12 with roughly the same specificity, but much lower sensitivity (almost three times as many false negatives).

Spaced seeds permitting errors are much more useful than unspaced seeds allowing errors. For example, the one-mismatch seed BLAT-11-12, with one hit, has false negative rate 14.8%, while the best vector seed, VS-11-12, has false negative rate 4.9%, three times lower. Both have the same false positive rate, and are equally simple to implement.

Two hits of the same seed required. The situation is even more dramatic if we may require two hits in the alignment. For example, the seed VS-9-11, allowing two mismatches, has unacceptably high false positive rate for one hit. If we require two hits, however, the false positive rate drops to 1.58×10^{-6}, comparable that for one hit to the VS-11-12 seed. Yet the false negative rate is an astonishing 0.1%. While there is some overhead involved in throwing out the many single hits that aren't extended, this can still be done extremely quickly.

If one seeks much better specificity, VS-11-12, with two hits, allows false positive rate 4.8×10^{-10}, over three orders of magnitude better than for one hit to PH-10, with comparable false negative rate (19.1%).

The vector seeds with support between ten and twelve may be appropriate for practice with two hit models. If an input sequence is large, say 20 Mb, the expected number of entries in each site of a hash table will be at least one for seeds of these supports, and the added work to identify double hits should be moderate for these seeds. Our tentative recommendation is two hits to the seed

Seed				One hit		Two hits	
Vector	T	Support	Name	False negative	False positive	False negative	False positive
1111111111	10	10	BLAST-10	42.0%	9.54×10^{-7}	77.9%	9.10×10^{-11}
11111001101011	10	10	PH-10	22.4%	9.54×10^{-7}	55.6%	9.10×10^{-11}
111111111111111	13	15	BLAT-13-15	11.3%	9.23×10^{-7}	34.4%	8.52×10^{-11}
11111011111011111	13	15	VS-13-15	4.11%	9.23×10^{-7}	18.9%	8.52×10^{-11}
11101110110101111	12	13	VS-12-13	10.4%	5.96×10^{-7}	34.5%	3.56×10^{-11}
111111111111	11	12	BLAT-11-12	14.8%	2.21×10^{-6}	41.9%	4.86×10^{-10}
1111110011010111	11	12	VS-11-12	4.89%	2.21×10^{-6}	19.1%	4.86×10^{-10}
101111011001111	9	11	VS-9-11	$< 0.01\%$	1.26×10^{-4}	0.1%	1.58×10^{-6}

Table 1. Theoretical performance of seeds of different support and with different allowed number of mismatches. BLAST-X – unspaced seeds of support X; BLAT-X-Y – unspaced seeds with allowed mismatches; PH-X – optimized spaced seeds of support X; VS-X-Y – optimized vector seeds (allowing both spaced seeds and mismatches).

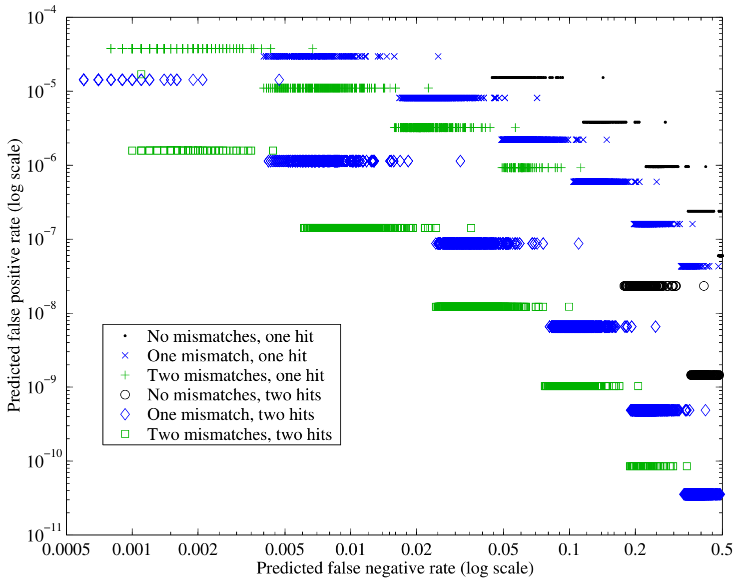

Fig. 1. False positive and false negative rates of many seeds with zero, one or two mismatches permitted, according to the simple Bernoulli model of alignments. Seeds in a horizontal row in the figure have the same support, threshold, and number of hits; the unspaced BLAT or BLAST seeds always have the lowest sensitivity. Two hits to the best one-mismatch seed of support 12 are found in 81% of alignments, comparable to two hits to the best PatternHunter seed of weight 9 or one hit to the best PatternHunter seed of weight 10, yet these have orders of magnitude more false positives.

VS-11-12, with false negative rate 19.1% (better than one hit to the best seed of support 10), and false positive rate 4.86×10^{-10}.

Seed vector	False Negative	False Positive
No augmentation	10.4%	5.96×10^{-7}
1111111111111 (worst)	8.7%	1.18×10^{-6}
1110010111111111 (median)	6.1%	1.16×10^{-6}
11110101101110111 (mirror)	5.6%	1.18×10^{-6}
11101110110101111 (best)	5.4%	1.16×10^{-6}

Table 2. Theoretical performance of two-seed models when a seed of support 13 and threshold 12 is added to the seed VS-12-13 from Table 1.

Interestingly, we found that the sensitivity of a seed to one hit is an excellent predictor of its sensitivity to two hits. The sensitivity of a seed to two hits is quite consistently close to the cube of the sensitivity to one hit, with correlation coefficient $r^2 = .9982$. The fourteen seeds of support 11 allowing no errors with highest sensitivity to one hit are also the best for sensitivity to two hits; this pattern is consistent for other supports and seed lengths. This suggests that one need only consider sensitivity of seeds to one hit, perhaps computing sensitivity to two hits for appealing seeds.

Two seeds are better than one. One can use a pair of seeds instead of one, allowing matches to either seed. We can avoid twice completing alignments that hit both seeds, so runtime will roughly double for false positives and not change at all for true positives found with both seeds. We considered adding a different seed of support 13 with threshold 12 to the seed VS-12-13, which has false negative rate 10.4% by itself.

The results are shown in Table 2. The best pair halves the false negative rate while the worst augmentation (non-spaced seed with one mismatch allowed) only improves false negatives slightly, yet will still double runtime. Interestingly, one of the best seeds to augment the seed with is its mirror image. This seed has the same sensitivity by itself as VS-12-13.

Of course, there is no evidence that the best seed pair includes the best solo seed in it; however, it is a reasonable heuristic. (The best pair found here was also superior to 1000 random pairs of seeds.) It is also sensible in the context that one is unhappy with the results of a search merely using the first seed.

4.2 DNA seeds for coding alignments

We conducted further experiments on a data set consisting of alignments of homologous coding regions from human and fruit fly. The initial set contained 339 protein alignments. One protein alignment can yield several DNA alignments (or *fragments*) because the coding regions for each protein can be interrupted by non-coding introns. The final data set contained 972 ungapped fragments in the training set and 810 gapped fragments in the testing set, after discarding weak and short fragments. A detailed description of the data set can be found elsewhere [4] and the data set can be obtained at `http://www.bioinformatics.uwaterloo.ca/supplements/03seeds`. We model aligned coding regions by the

Seed Vector	T	Support	Name	Actual False Negative	Predicted False Negative	Predicted False Positive
1111111111	10	10	BLAST-10	56.7%	71.4%	9.54×10^{-7}
11011000011011011	10	10	CPH-10	14.0%	32.8%	9.54×10^{-7}
110110110000110110010110111	13	15	CVS-13-15	4.0%	19.1%	9.23×10^{-7}
110110120000120110010110111	15	15	CVS2-15-15	4.9%	21.1%	7.0×10^{-7}
110110110000110110000110111	13	14	CVS-13-14	13.1%	33.1%	1.6×10^{-7}
110110120000110110000110112	15	14	CVS2-15-14	13.8%	34.4%	1.4×10^{-7}
120120120000120120010120112	20	15	CVS2-20-15	9.3%	34.6	2.7×10^{-7}
11111111111	10	11	BLAT-10-11	16.2%	32.4%	7.9×10^{-6}
111111111	9	9	BLAST-9	46.3%	60.2%	3.82×10^{-6}
11001011000011011	9	9	CPH-9	8.14%	24.0%	3.82×10^{-6}
1101100001100001101101 1	11	12	CVS-11-12	5.2%	19.1%	2.2×10^{-6}
12011000011000012011011	13	12	CVS2-13-12	5.6%	20.5%	2.2×10^{-6}
12022012000012 (two hits)	12	8	CVS2-12-8	6.0%	29.3%	2.3×10^{-6}

Table 3. Theoretical and actual performance of various coding detection seeds. BLAST-X – unspaced seeds of support X; BLAT-X-Y – unspaced seeds with allowed mismatches; CPH-X – spaced seeds of support X optimized in coding-aware model; CVS-X-Y – vector seeds (allowing spaced seeds and mismatches) optimized in coding-aware model; CVS2-X-Y – same as CVS, except values from $\{0, 1, 2\}$ are allowed.

three-periodic model described earlier. We used the training set to estimate the probability that a codon triplet has a given alignment pattern and the alignment length distribution.

First we have investigated 1372 binary vector seeds (v, T) with supports $s \in \{10, \ldots, 15\}$ and $T \geq s - 2$. The seeds we have investigated have codon structure: they can be divided into triplets, where each triplet is either $(0, 1, 0)$, $(1, 1, 0)$ or $(0, 0, 0)$. In real alignments, the second codon position is most likely to be conserved and the third often varies.

We compared the theoretical performance of the codon-aware spaced seeds versus their performance on our test set. Here, we also found quite striking advantages of vector seeds over seeds that are unspaced or that do not allow error. The model is a good predictor of the performance of a seed, though our seeds do better than predicted, as the model is not aware of highly conserved parts of alignments. Results for some interesting seeds are shown in Table 3.

We then chose the best seeds allowing mismatches, at each support/threshold combination (the seeds denoted by CVS-X-Y in Table 3), and explored the effect of fixing the middle positions of some of their codons, by setting the corresponding position in the seed vector to two and raising the threshold by one. Results for this experiment are shown in Table 3. It shows the performance of a collection of BLAST and BLAT seeds, optimized spaced seeds, and optimized vector seeds, chosen for sensitivity to this model.

This experiment highlights the expressive richness of vector seeds. The seed CVS2-15-15 in Table 3 matches fully 95% of alignments. This is comparable to

Property	N	mean	std. dev.
Number of fragments	566	5.80	4.23
Length of longest fragment	566	59.6	20.4
Length of other fragments	2718	20.4	13.1

Table 4. Parameters characterizing the gamma distributions that approximate the properties of protein alignments from our data set.

the sensitivity of the BLAST-7 seed, whose false positive rate is ninety times higher. It is also preferred over the BLAT-10-11 seed for both sensitivity and specificity, and over the CPH-10 seed, which has comparable specificity.

Lastly, we note that the seeds we examined are also overwhelmingly preferred for two-hit study as well. While they are long enough that they reduce sensitivity below 65% in two-hit models still better than the BLAST-9 seed, they also admit tens of thousands of times fewer false positives. One can also use shorter seeds for use with two hits, which still allows for simpler hash table structures. The last seed in Table 3 is especially appealing; the support for this seed is just eight, making for very simple hash table structures, while the performance is comparable to one hit to the longer seed CVS2-13-12 in the table.

4.3 Seeds for protein alignments

We also studied the use of vector seeds for protein alignments. Our data set consisted of randomly chosen BLASTP 2.0.2 alignments [2] of pairs of human sequences, taken from 8654 human proteins in the SWISSPROT database [3], release 40.38. We chose 566 alignments with score between 50 and 75 bits using the BLOSUM62 scoring matrix, with a requirement that there are at most eight alignments from each collection of genes connected by BLASTP hits. (This restriction avoids choosing too many matches from one family of very common proteins; without it, the majority of alignments would be from only one family.)

To train our probabilistic model we estimated the distribution of the number of ungapped fragments, the length of the longest ungapped fragments, and the length of all other fragments as gamma distributions from this set of alignments. These parameters are summarized in Table 4. Other parameters of the probability distribution were obtained directly from BLOSUM62 matrix.

We investigated 237 seeds of vector length at most six with between three and five ones and the remaining positions all zero. We chose values of T between 11 and 18 for seeds with 3 ones, between 12 and 22 for seeds with 4 ones, and between 15 and 25 for seeds with 5 ones. There is an observational bias in our experiments since the test set consists of alignments found by two-hit BLASTP. This guarantees that the two-hit BLASTP seed will match all alignments!

The advantage of spaced seeds over unspaced seeds is not as dramatic as for nucleotide matches, as seen in Figure 2 and Table 5. This is because BLASTP seed matches for proteins are more independent than BLASTN hits are for nucleotides. For example, the probability of a pair of amino acids scoring at least +5 is only 0.22, in homologous protein sequence, while the probability of a 1 is

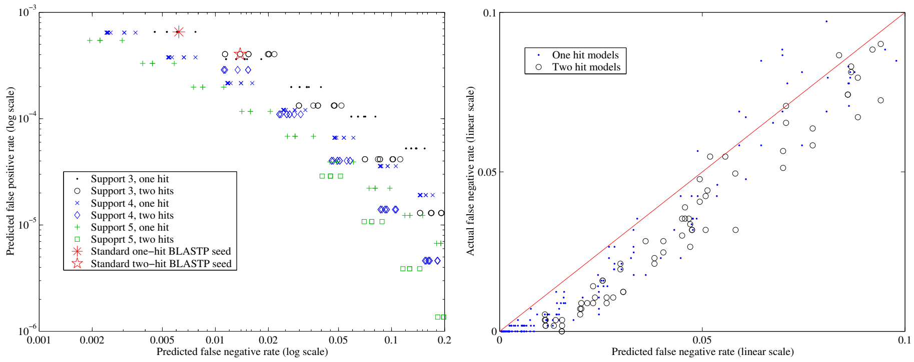

Fig. 2. Theoretical and actual performance of seeds for protein alignments. The left graph shows the predicted false positive rate and false negative rate for several seeds of different supports and threshold. The commonly used one-hit and two-hit BLASTP seeds are shown in red; both are dominated by vector seeds with the same support and by vector seeds with different support. The right graph compares the actual performance of seeds for protein alignments with their theoretical performance. The model does a good job of predicting the sensitivity of seeds, though for the seeds with highest sensitivity, the accuracy is worse. This is due to observation bias (alignments were found by BLASTP), and because the data set is of size 566.

0.7 in the PatternHunter model. Thus, for proteins, immediately following a hit, there is a lower probability of another hit.

Also, a disadvantage of spaced seeds is that they offer fewer positions for a possible match; when ungapped fragments are as short as they are for our protein alignments (with fragments of length 20 being reasonably common), this reduction affects seed sensitivity.

We also performed an experiment to verify that the false positive rates predicted were close to those found in reality. We chose 100 proteins at random and built a hash table for each one-hit seed in Table 5. We then counted the number of seed hits when we chose 100 other random proteins. For all seeds, the false positive rate was within 10% of what was predicted by the background model.

Still, there is some advantage to the use of spaced seeds. The spaced seed $((1, 1, 1, 0, 1), 14)$ offers a theoretical advantage of 61% fewer false positives over the standard BLAST seed,$((1, 1, 1), 13)$, with slightly lower false negative rate; for that matter, the obvious spaced seed of support 3, $((1, 0, 1, 1), 13)$, is also preferred over the standard BLAST seed, with 27% fewer missed alignments (though the difference is small).

The advantage of two-hit models is not as great as for nucleotides, either. One notable discovery is that seeds depending on more than three positions have greater sensitivity than those depending on just three. Given that there are only 160,000 amino acid 4-mers, for large protein databases, most hash table entries would be populated if one used seeds depending on four positions.

If one is willing to tolerate an error rate comparable to the theoretical performance of the one-hit BLAST seed of threshold 15 (approximately 3.4% false

Seed Vector	Threshold	Hits	Actual False Negative	Predicted False Negative	Actual False Positive	Predicted False Positive
110111	15	2	2.47% (14/566)	4.04%		2.9×10^{-5}
111	13	1	0.18% (1/566)	0.62%	6.9×10^{-4}	6.5×10^{-4}
1011	13	1	0.18% (1/566)	0.45%	6.9×10^{-4}	6.5×10^{-4}
11101	14	1	0.18% (1/566)	0.24%	6.7×10^{-4}	6.4×10^{-4}
111101	15	1	0.00% (0/566)	0.19%	5.6×10^{-4}	5.4×10^{-4}
111	15	1	3.53% (20/566)	3.39%	2.0×10^{-4}	1.9×10^{-4}
10111	16	1	0.35% (2/566)	1.18%	2.3×10^{-4}	2.2×10^{-4}
111 (*)	11	2	0.00% (0/566)	1.38%		2.1×10^{-3}
1011	11	2	0.35% (2/566)	1.14%		2.1×10^{-3}

Table 5. Theoretical and actual performance of various protein seeds. The unspaced seeds are commonly used BLAST seeds, which are potentially improved upon by the spaced seeds listed after them. We also show the false positive rates for the one hit models in comparisons between 200 unrelated proteins.

negatives), the one-hit seed $((1,0,1,1,1),16)$, offers comparable false positives, with one third the false negative rate (1.2%). Or, if one is willing to use seeds of support 5, the two-hit seed $((1,1,0,1,1,1),15)$ offers 4.0% false positives and seven times fewer false positives in theory.

A data set not derived from BLASTP. To try to avoid the observation bias we noted before, we aligned sequences that are not reported as aligned by BLASTP, but which are connected by a sequence of alignments that *are* reported by BLAST. If BLAST defines a graph on the set of proteins, these are nonadjacent nodes in the same connected component. We aligned all such pairs from connected components of size at most 30, and discovered 396 alignments in our target score range, again using the BLOSUM62 scoring matrix.

However, all match the one-hit BLASTP seed with threshold 13, and all but three match the two-hit BLASTP seed with threshold 11. Presumably, all were incorrectly thrown out during one of BLASTP's many filtering steps.

In this new data set, we do see some advantage to spaced seeds, especially among less sensitive seeds. For example, the spaced vector seed $((1,1,0,1),16)$ matches 381 (96.2%) of these alignments, while the unspaced seed $((1,1,1),16)$ matches 367 (92.6%) of them. Similarly, the spaced support 4 seed $((1,1,0,1,1), 18)$ matches 382 alignments (96.4%), while the unspaced seed $((1,1,1,1),18)$ matches 373 (94.2%). We found similar results for two hit models.

Predictions versus reality. Finally, the simple model of sequences allows good predictions of seed sensitivity. Figure 2 shows the predicted and actual sensitivity of the seeds we studied. The model generally does a good job at predicting the sensitivity of the seeds, except for the most sensitive seeds. This is expected, since our test set includes only 566 alignments. Our set is also subject to observation bias: the alignments we tested were found with BLASTP, which guarantees that they have a hit to the two-hit seed $((1,1,1),11)$!

5 Conclusion

We have presented an extension to previous models for hits that are used in large-scale local alignments. Our vector seeds offer a much wider vocabulary for seed matches than previously studied seeds. For example, they allow certain positions to be more important than others, they allow a fixed number of mismatches in some positions and an arbitrary number in others, and more.

Our extensions to spaced seeds or seeds with constrained mismatches allow substantially improved pairwise local alignment, with vastly improved sensitivity. Especially with the coding sequence nucleotide alignments (possibly one of the most important large-scale local alignment problems), alignment programs using our seeds can reduce their false negative rates by over half, with no change to false positives over BLAT or PatternHunter.

We have also shown an algorithm that allows us to predict the sensitivity and specificity of a vector seed on probabilistic models of true and random alignments. This allows us to choose an optimal seed for a given alignment task. Our algorithm is an extension to the original Keich *et al.* algorithm for predicting seed sensitivity in simple models. Our method is practical as long as the number of alignment sequences matching a given seed is moderate. Extensions to our algorithm allow one to predict the sensitivity of a seed in multihit models, or when using multiple seeds.

We show that spaced seeds can be helpful for proteins as well. The improvements are not as dramatic as for nucleotides, mostly because the seeds themselves are so short, yet they are still useful, and if one is willing to allow a moderate false positive rate, spaced seeds are strongly preferred over unspaced seeds. We also show the contexts under which two-hit models are preferred over one-hit models for proteins.

Our results offer substantial improvement over the current state of the art, with minimal change required in coding.

Future work. Finally, we discuss a few extensions which we have only begun to consider. In PSI-BLAST [2], alignment phases after the first are based on position-specific scoring matrices, which model the probabilities expected in sequence matching a profile. Much as with the standard BLASTP model of aligning sequences that comes from the scoring matrix, here as well, one may desire a seed that has a higher probability of matching the sequence than the usual $(1^k, T)$ consecutive seed. It is impractical to compute the probabilities for thousands of seeds for each profile, yet it is quite reasonable to compute the match rates for each of a small set of diverse good seeds, and use the best of these seeds in that round. (One may also compute predicted false positive rates, using the standard background probabilities with the new scoring matrices.)

We are also interested in whether the representation of vector seed hits as half-spaces in a lattice can help in optimizing them. Given a certain length, support and threshold, is it possible to find the best seed, even for a simple position-independent model, without using essentially exhaustive search? While the exhaustive search is possible for small seed families, for the vector seed models, this becomes absurd as the families of possible seeds grows to the millions.

Clearly, one could use heuristic methods. However, sometimes, there is a large difference between the best seed and seeds at the 99th percentile, so one would want a very good heuristic.

Finally, we demonstrated that the vector seeds are not universally expressive. For example, the is no way in the vector seed model to require that three of the four codons are match in the first two positions each. Is it possible to extend vector seeds so that they are more expressive?

Acknowledgments. This work has been supported by a grant from the Human Science Frontier Program, grant from the Natural Sciences and Engineering Research Council of Canada, and by an Ontario Graduate Scholarship. We would like to thank anonymous referees for many constructive comments on the paper.

References

1. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.
2. S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3392, 1997.
3. A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Research*, 28(1):45–48, 2000.
4. B. Brejová, D. Brown, and T. Vinař. Optimal spaced seeds for hidden Markov models, with application to homologous coding regions. In *Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching (CPM)*, 2003. To appear.
5. J. Buhler, U. Keich, and Y. Sun. Designing seeds for similarity search in genomic dna. In *Proceedings of the 7th Annual International Conference on Computational Biology (RECOMB)*, pages 67–75, 2003.
6. S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89(22):10915–10919, 1992.
7. U. Keich, M. Li, B. Ma, and J. Tromp. On spaced seeds. Unpublished.
8. W. J. Kent. BLAT–the BLAST-like alignment tool. *Genome Research*, 12(4):656–664, 2002.
9. I. Korf, P. Flicek, D. Duan, and M. R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17 Suppl 1:S140–8, 2001.
10. B. Ma, J. Tromp, and M. Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, March 2002.