Optimizing Multiple Seeds for Protein Homology Search

Daniel G. Brown

Abstract—We present a framework for improving local protein alignment algorithms. Specifically, we discuss how to extend local protein aligners to use a collection of *vector seeds* or ungapped alignment seeds to reduce noise hits. We model picking a set of seed models as an integer programming problem and give algorithms to choose such a set of seeds. While the problem is NP-hard, and Quasi-NP-hard to approximate to within a logarithmic factor, it can be solved easily in practice. A good set of seeds we have chosen allows four to five times fewer false positive hits, while preserving essentially identical sensitivity as BLASTP.

Index Terms—Bioinformatics database applications, similarity measures, biology and genetics.

1 Introduction

Pairwise alignment is one of the most important problems in bioinformatics. Here, we continue an exploration into the seeding and structure of local pairwise alignments and show that a recent strategy for seeding nucleotide alignments can be expanded to protein alignment. Heuristic protein sequence aligners, exemplified by BLASTP [1], find almost all high-scoring alignments. However, the sensitivity of heuristic aligners to moderate-scoring alignments can still be poor. In particular, alignments with BLASTP score between 40 and 60 are commonly missed by BLASTP, even though many are of truly homologous sequences. We focus on these alignments and show that a change to the seeding strategy gives success rates comparable to BLASTP with far fewer false positive hits.

Specifically, multiple spaced seeds [2] and their relatives, vector seeds [3], can be used in local protein alignment to reduce the false positive rate in the seeding step of alignment by a factor of four. We present a protocol for choosing multiple vector seeds that allows us to find good seeds that work well together. Our approach is based on solving a set-cover integer program whose solution gives optimal thresholds for a collection of seeds. Our IP is prone to overtraining, so we discuss how to reduce the dependency of the solution on the set of training alignments, both by increasing the false positive rate of the seeds found slightly and by making the program less sensitive to outliers. The problem we are trying to solve is NP-hard and Quasi-NP-hard to approximate to a sublogarithmic factor, so we present heuristics for it, though most instances are of moderate enough size to use integer programming solvers.

Our successful result here contrasts with our previous work [3] in which we introduced vector seeds. There, we found that using only one vector seed would not substantially improve BLASTP's sensitivity or selectivity. The use of multiple seeds is the important change in the present work. This successful use of multiple seeds is similar to what has been reported recently for pairwise nucleotide alignment [4], [5], [6], but the approach we use is different since protein aligners require extremely high sensitivity. We note that, independently of our work, the authors of PatternHunter, the first program to use optimized spaced seeds, have developed a protein aligner based on seeding approaches similar to those we discuss here [7]; however, they have not offered theoretical justification for their approach, which, in some sense, we provide here.

Our results confirm the themes developed by us and others since the initial development of spaced seeds. The first theme is that spaced seeds help in heuristic alignment because the very surprisingly conserved regions that one uses as a basis for building an alignment happen more independently in true alignments than for unspaced seeds. In protein alignments, there are often many small regions of high conservation, each of which has a chance to have a hit to a seed in it. With unspaced seeds, the probability that any one of these regions is hit is low, but, when a region *is* hit, there may be several more hits, which is unhelpful. By contrast, a spaced seed is likely to hit a given region fewer times, wasting less runtime, and will also hit at least one region in more alignments, increasing sensitivity.

The second theme is that the more one understands how local and global alignments look, the more possible it is to tailor alignment seeding strategies to a particular application, reducing false positives and improving true positives. Here, by basing our set of seeds on sensitivity to true alignments, we choose a set of seed models that hit diverse types of short conserved alignment subregions. Consequently, the probability that one of them hits a given alignment is high since they complement each other well.

2 Background: Heuristic Alignment and Spaced Seeds

Since the development of heuristic sequence aligners [1], the same approach has been commonly used: identify short, highly conserved regions and build local alignments

• *The author is with the School of Computer Science, University of Waterloo, 200 University Ave., West, Waterloo, ON N2L 3G1, Canada. E-mail: browndg@uwaterloo.ca.*

Manuscript received 1 Nov. 2004; revised 2 Jan. 2005; accepted 11 Jan. 2005; published online 30 Mar. 2005.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBBSI-0183-1104.

around these "hits." This avoids the use of the Smith-Waterman algorithm [8] for pairwise local alignment, which has $\Theta(nm)$ runtimes on input sequences A and B of length n and m, respectively. (We will use the notation $A[i]$ to represent the ith character of sequence A.)

Instead, assuming random sequences, the expected runtime of this heuristic search method is $h(n,m) + a(n,m)$, where $h(n,m)$ is the amount of time needed to find hits in the two sequences and $a(n,m)$ is the expected time needed to compute the alignments from the hits. Most heuristic aligners have $h(n,m) = \Theta(n + m + nm/k)$, while $a(n,m) = \Theta(nm/k)$ for some large constant k. There are many assumptions in these formulas. First, even when we align sequences with true homologies, most hits are between unrelated positions, so the estimation of the runtime need not consider whether the sequences are related. Further, this simplification assumes that each hit found in the first phase results in a constant amount of work being done in the second phase to identify that it is false (or that true hits are rare). It is the speedup factor of k that is important here; assuming m and n are large, the overall runtime is much faster.

Most heuristic aligners look at the scores of matching characters in short regions and use high-scoring short regions as hits. For example, BLASTP [1] hits are three consecutive positions in the two sequences where the total score, according to a BLOSUM or PAM scoring matrix, of aligning the three letters in one sequence to the three letters of the other sequence is at least +13. Finding such hits can be done easily, for example, by making a hash table of one sequence and searching positions of the hash table for the other sequence, in time proportional to the length of the sequences and the number of hits found. BLASTP uses more complicated data structures for this process, but the principle is similar.

2.1 Seeding Models

To generalize BLASTP's hits, we defined *vector seeds* [3], [9]. A vector seed is a pair (v, T). Vector $v = (v_1, \ldots, v_k)$ is a vector of position multipliers and T is a threshold. Given two sequences A and B, let $s_{i,j}$ be the score in our scoring matrix of aligning the $A[i]$ to $B[j]$. If we consider position i in A and j in B, we then get an hit to the vector seed at those positions when $v \cdot (s_{i,j}, s_{i+1,j+1}, \ldots, s_{i+k-1,j+k-1}) \geq T$. In this framework, BLASTP's seed is ((1, 1, 1), 13).

Vector seeds generalize the earlier idea of spaced seeds [2] for nucleotide alignments, where both scores and the vector are 0/1 vectors and where T, the threshold, equals the number of 1s in v. A spaced seed requires an exact match in the positions where the vector is 1 and the places where the vector is 0 are "don't care" positions. In our original work with vector seeds [3], the freedom to allow positions of v to have values beside 0 and 1 was not extremely useful, so the vector seeds we discuss here all have binary vectors v.

Spaced seeds have the same expected number of junk hits as unspaced seeds. For unrelated noise DNA sequences, this is $nm4^{-w}$, where w is the number of ones in the seed (its *support*). Their advantage comes because more distinct internal subregions of a given alignment will match a spaced seed than the unspaced seed; this happens because the hits are more independent of each other. The probability that an alignment of length 64 with 70 percent conservation matches a good spaced seed of support 11 can be greater than 45 percent because there are likely to be more subregions that match the spaced seed than the unspaced seed; by contrast, the default BLASTN seed, which is 11 consecutive required matches, hits only 30 percent of alignments.

Spaced seeds have three advantages over unspaced seeds. First, their hits are more independent, which means that it is more likely that a given alignment has at least one hit to a seed; fewer alignments have many. Second, the seed model can be tailored to a particular application: If there is structure or periodicity to alignments, this can be reflected in the design of the seeds chosen. For example, in searching for homologous codons, they can be tailored to the three-periodic structure of such alignments [10], [11]. Finally, the use of multiple seeds allows us to boost sensitivity well above what is achievable with a single seed, which, for nucleotide alignment, can give near 100 percent sensitivity in reasonable runtime [4].

Keich et al. [12] have given an algorithm for a simple model of alignments to compute the probability that an alignment hits a seed; this has been extended by both Buhler et al. [10] and Brejova et al. [11] to more complex sequence models. Choi et al. [13] have also shown experimental results for spaced seeds with high sensitivity across a wide range of homologies. Kucherov et al. [14] show how to adapt spaced seeds to the interesting case of alignments where no subregion of the alignment has a higher score than the entire alignment.

2.2 Some Newer Seeding Models

Another seeding model, which has recently arisen [7], [15] is of *ungapped alignment seeds*. These were developed by Brown and Hudek [15] to anchor global alignments of ambiguous DNA sequences and, independently, by Kisman et al. [7] in their heuristic protein aligner, tPatternHunter.

An ungapped alignment seed is a vector v, a global threshold T, and a vector of positional minimum scores b. There is a match between positions in the two sequences when the vector of pairwise match scores is at least as large, position-by-position, as the minimum scores vector b and where the dot product of the position-by-position scores and the multiplier vector v is at least T. These seeds are a compromise between spaced seeds and consecutive seeds: They require spaced positions to have good scores (those where the lower bound vector b has high values), while also focusing on the quality of the local alignment at the seed by possibly examining all of the positions of the seed. It is not possible to cast an ungapped alignment seed in the language of vector seeds because of the requirement that each individual position's score is greater than its bound. It is possible to cast a vector seed as an ungapped alignment seed, by setting the b vector to $-\infty$ in all positions, thus removing the position-by-position lower bound requirement.

Csürös [16] has also extended this framework of seeding to look at variable-length seeds, where the length of the regions that must match depends on their positional scores. While this approach can also be brought into the framework of the present work, we have not done so in our experiments.

2.3 Multiple Seeds

Another important extension to these ideas of seeding has been the use of multiple seeds of different sorts in basing alignments. In this approach, an attempt is made to perform extension when any of a collection of seed models has a hit. This will work well if each chosen seed has a very low false positive rate so that their total false positive rate is still below that of one seed of comparable sensitivity.

Several authors [2], [3], [4], [6], [10], [17] have proposed using multiple seeds and given heuristics to choose them. This problem was recently given a theoretical framework by Xu et al. [5] and, independently, Kucherov et al. [18] studied heuristic algorithms for identifying sets of good seeds. In work unrelated to the present work, Kisman et al. [7] have heuristically used multiple ungapped alignment seeds (though not called by that term) for protein alignment. To the best of our knowledge, the present work is the first work to choose multiple seeds for protein alignment with a theoretical basis.

3 Choosing a Good Set of Seeds

Spaced seeds have made a substantial impact in nucleotide alignments, but less in protein alignment. Here, we show that they have use in this domain as well. Specifically, multiple vector seeds or multiple ungapped alignment seeds, with high thresholds, give essentially the sensitivity of BLASTP with four times fewer noise hits. Slightly fewer alignments are hit, but the regions of alignment hit by the vector seeds are all of the same good ones as hit by the BLASTP seed and a few more. In other words, BLASTP hits more alignments, but the hits found by BLASTP and not the vector seeds are mostly in areas unlikely to be expanded to full alignments.

We adapt a framework for identifying sets of seeds introduced by Xu et al. [5]. We model multiple seed selection as a set cover problem and give heuristics for the problem. For our purposes, one advantage of the formulation is that it works with explicit alignments: Since real alignments may not look like a probabilistic model, we can pick a set of seeds for sensitivity to a collection of true alignments. Unfortunately, this also gives rise to problems, as the thresholds may be set high due to overtraining for a given set of alignments.

Most of our experiments concern themselves with vector seeds, but the framework can be expanded straightforwardly to ungapped alignment seeds as well. This is because we do not compute theoretical sensitivity of the seeds, but, instead, only identify hits in existing real alignments. Indeed, our framework is quite broad and extends to many different models for seeding as long as the assumption that false positives are additive is reasonably accurate and that one can compute that false positive rate for the seed models. Where the ungapped alignment seeds require some thought, we present the addition needed for them.

3.1 Background Rates

One important detail that we need before we begin is to the background hit rate for a given vector or ungapped alignment seed. We noted previously [3] that this can be computed for vector seeds, given a scoring matrix; it is also straightforward to compute for ungapped alignment seeds as well. Namely, from the scoring matrix, we can compute the distribution of letters in random sequences implied by the matrix; this can then be used to compute the distribution of scores found in unrelated sequences. Using this, we can compute the probability that unrelated sequences give a hit to a given seed at a random position, which we call the false positive rate for that seed. In fact, we can easily compute the entire probability distribution on the score for a given seed vector at a random position. Similarly, we can compute this probability under the constraint that positional scores have minimum value, thus expanding to ungapped alignment seeds.

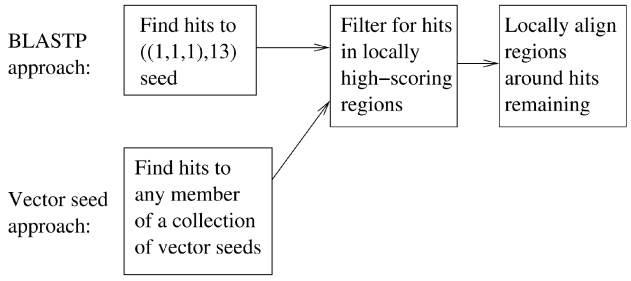

Fig. 1. Flowchart contrasting BLASTP's approach to heuristic sequence alignment to the one proposed here. The only difference is in the initial collection of hits. The smaller collection of hits found with the variations on seeds gives as many hits to true alignments that survive to the third stage as does BLASTP, yet far fewer noise hits must be filtered out.

For the default BLASTP seed, the probability that two random unrelated positions have a hit is quite high, 1/1,600. Because of this high level of false positives, BLASTP must filter hits further in hopes of throwing out hits in unrelated sequences. Specifically, BLASTP rapidly examines the local area around a hit and, if this region is not also well-conserved, the hit is thrown out. Sometimes, this filtering throws out all of the hits found in some true alignments and, thus, BLASTP misses them, even though they hit the seed. One way of modeling this filtering is to view BLASTP as testing two seeds simultaneously: The vector seed ((1, 1, 1), 13) and an ungapped alignment seed that looks at the region surrounding the seed hit.

Our goal in using other seed models here is to reduce the false positive rate, while still hitting the overwhelming majority of alignments and hitting them in places that are highly enough conserved as to make a full alignment likely. A flowchart of our proposal, and the approach of BLASTP, is in Fig. 1.

For a set Q of alignment seeds, we say that its false positive rate is the probability that *any* seed in Q has a hit to two random positions in unrelated sequences. This is not equal to the sum of the false positive rates for all seeds in Q since hits to one seed may overlap hits to another. However, we will use this approximation in our optimization. As we extend to a very large collection of seeds in Q, this can become worrisome as the same false positive may be counted many times. However, this may be appropriate, in fact, depending on how the search is done to find the false hits.

3.2 An Integer Program to Choose Many Seeds

Here, we give an integer program to find the set of seeds that hits all alignments in a given training set with overall lowest possible false positive rate. We will show that our IP encodes the Set-Cover problem and that it is NP-hard to solve and Quasi-NP-hard even to approximate to a sublogarithmic factor. However, for moderate-sized training sets, we can solve it, in practice, or use simple heuristics to get good solutions.

Given a set of alignment seeds Q, we say that they hit a given alignment a if any member of Q has a hit to the alignment. Our goal in picking such a set will be to minimize the false positive rate of the set Q, with the requirement that we hit *all* alignments in a training collection, A.

This optimization goal is the alternative to the goal of Xu et al. [5]. In that work, we maximized seed sensitivity when a maximum number of spaced seeds is allowed; given that all possible seeds had the same false positive rate, this was equivalent to maximizing sensitivity for a given false positive rate. This alternative goal of minimizing false positives when we want 100 percent sensitivity on the training set is appropriate for protein alignment; however, we want to achieve extremely high sensitivity, as close to 100 percent as possible.

3.2.1 The Integer Program

Here, we show how to cast this seed selection problem as an integer program. Recall that a seed model is the vector v of multipliers or for an ungapped alignment seed, the vector v of multipliers, and the vector b of positional lower bounds. We will call this vector or vectors the "pattern" of a seed. We can then view choosing a set of vector or ungapped alignment seeds as choosing thresholds for each pattern.

More formally, suppose we are given a collection of alignments $A = \{a_1, \ldots, a_m\}$ and a set of seed patterns $P = \{p_1, \ldots, p_n\}$. We will choose thresholds $(T_1^*, \ldots, T_n^*)$ for the patterns of P such that the seed model set $Q^* = \{(p_1, T_1^*), \ldots, (p_n, T_n^*)\}$ hits all alignments in A and the false positive rate of Q^* is as low as possible. The T_i^* may be ∞, which corresponds to not choosing the pattern p_i at all.

We require that each alignment a must be hit, so one of the thresholds must be low enough to hit a. To verify this, we compute the best-scoring hit for each seed pattern p_i in each alignment a_j; let the score of this hit be $T_{i,j}$. If we choose T_i^* so that it is at most $T_{i,j}$, then the seed (p_i, T_i^*) will hit alignment a.

To model this as an integer program, we have a collection of integer variables $x_{i,T}$ for each possible threshold value for seed pattern p_i. We note that we are requiring that this number is a small number or can be granularized reasonably since each possible threshold will get its own constraint. For simple seeds from a BLOSUM matrix, the scores at a position come in a small range of integers, so the possible reasonable thresholds form a small range; let T_m be the smallest such threshold. We will set variable $x_{i,T}$ to 1 when the threshold or seed vector x_i is at most T; for each pattern p_i, its threshold chosen is the smallest T, where $x_{i,T} = 1$.

To compute the false positive rate, we let $r_{i,T}$ be the probability that a random place in the background model has score exactly T according to the seed model (p_i, T). We add these up for all of the false hits with score equal to or greater than the chosen thresholds. Our integer program is as follows:

$$\begin{aligned} \min \quad & \sum_{i,T} x_{i,T} r_{i,T}, \text{ such that} \\ \sum_i x_{i,T_{i,j}} \quad & \geq \quad 1 \text{ for all alignments } a_j \\ x_{i,T} \quad & \geq \quad x_{i,T-1} \text{ for all thresholds } T > T_m \\ x_{i,T} \quad & \in \quad \{0,1\} \text{ for all } i \text{ and } T. \end{aligned}$$

Our framework is quite general: Given *any* collection of alignments and the sensitivity of a collection of seeds to the alignments, one can use this IP formulation to choose thresholds to hit all alignments while minimizing false positives. In particular, one could require that a hit satisfy multiple seeds simultaneously or use more complicated hit formulations. Of course, for these harder models, one might have a more difficult time optimizing the integer program.

3.2.2 NP-Hardness

We now show that the problem of optimizing the seed set to minimize the false positive rate while hitting all alignments is NP-hard and that it is Quasi-NP-hard to approximate to within a logarithmic factor [19]. (That is, assuming NP does not have polynomial-time deterministic algorithms running in $O(n^{O(\log \log n)})$ time, no polynomial-time algorithm exists with approximation ratio $o(\log n)$.)

We show this by giving an approximation-preserving reduction of the Set-Cover problem to this problem. Since Set-Cover is Quasi-NP-hard to approximate to within a logarithmic factor [19], so is our problem.

An instance of Set-Cover is a ground set S and a collection $\mathcal{T} = \{T_1, \ldots, T_m\}$ of subsets of S; the goal is the smallest cardinality subset of T whose union is S. The connection to our problem is clear: We will produce one alignment per ground set member and, for each of the elements of T, we will have one seed. For simplicity, we will assume that $S = \{1, \ldots, n\}$. To fill the construction out, we will assign the vector seed

$$v_i = ((1, \overbrace{0, \ldots, 0}^{i}, 1), 1)$$

to every ground set element s_i. In a model of sequence where all positions are independent of all other, each of these seeds has the same false positive rate, so the false positive rate will be proportional to the number of ground set members chosen.

Then, for each set $T_j \in \mathcal{T}$, we create an alignment A_j of length $2n^2 + 4n$ by pasting together in n blocks of length $2n + 4$. If i is in T_j, then we make the ith block of the alignment have the first and $i + 2$nd position be of score 1, while all other positions in the block have score zero, while if $i \notin T_j$, then the ith block is all score zero. Then, it is clear that if we choose the seed v_i, we will hit all alignments A_j, where $i \in T_j$. If we desire the minimum false positive rate to hit all alignments, this is exactly equivalent to choosing the minimum cardinality set to cover all of the T_j.

Thus, we have presented an approximation-preserving transformation from Set-Cover to our problem and it is both NP-hard and Quasi-NP-hard to approximate to within a logarithmic factor.

3.2.3 Expansions of the Framework

In our experiments, we use the vector seed requirement as a threshold; one could use a more complicated threshold scheme to focus on hits that would be expanded to full alignments. That is, our minimum threshold for $T_{i,j}$ could be the highest-scoring hit *that is expanded to a full alignment* of seed vector v_j in alignment a_i. We could also have a more complicated way of seeding alignments and, still, as long as we could compute false positive rates, we could require that all alignments are hit and minimize false positive rates.

Also, we can limit the total number of vector seeds used in the true solution (in other words, limit the number of vectors with finite threshold). We do this by putting an upper bound on $\sum_i x_{i,T}$ for the maximum threshold T. In practice, one might want an upper bound of four or eight seeds, as each chosen seed requires a method to identify hits and one might not want to have to use too many such methods in the goal of keeping fewer indexes of a protein sequence database, for example.

Further, we might want to not allow seeds to be chosen with very high threshold. The optimal solution to the problem will have the thresholds as on the seeds as high as possible while still hitting each alignment. This allows overtraining: Since even a tiny increase in the thresholds would have caused a missed alignment, we may easily expect that, in another set of alignments, there may be alignments just barely missed by the chosen thresholds. This is particularly possible if thresholds are allowed to get extremely high and only useful for a single alignment. This overtraining happened in some of our experiments, so we lowered the maximum so that they were either found in a fairly narrow range (+13 to +25) or set to ∞ when a seed was not used. As one way of also addressing overtraining, we considered lowering the thresholds obtained from the IP uniformly or just lowering the thresholds that have been set to high values.

And, finally, the framework can be extended to allow a specific number of alignments to be missed. For each alignment, rather than requiring that

$$\sum_i x_{i,T_{i,j}} \geq 1,$$

which requires that some threshold be chosen so that the alignment is hit, we can add a $0/1$ slack variable to count how many are missed, changing the constraint to

$$\sum_i x_{i,T_{i,j}} + s_j \geq 1.$$

Then, if we require that

$$\sum_j s_j \leq M,$$

this allows at most M alignments to be so missed. This may be appropriate to allow the optimization framework to be less sensitive to a small number of outliers. We show experiments with this slightly expanded framework in the next section.

We note one simplification of our formulation: False hit rates are not additive. Given two spaced seeds, a hit to one may coincide with a hit to the other, so the background rate of false positives is lower than estimated by the program. When we give such background rates later, we will distinguish those found by the IP from the true values.

3.2.4 Solving the IP and Heuristics

To solve this integer program or its variations is not necessarily straightforward since the problem is NP-hard. In our experiments, we used sets of approximately 400 alignments and the IP has been able to solve directly quickly, using the CPLEX 9.0 integer programming solver.

Straightforward heuristics also work well for the problem, such as solving the LP relaxation and rounding to 1 all variables with values close to 1, until all alignments are hit, or setting all variables with fractional LP solutions to 1 and then raising thresholds on seeds until we start to miss alignments.

We finally note that a simple greedy heuristic works well for the problem, as well: Start with low thresholds for all seed patterns and repeatedly increase the threshold whose increase most reduces the false positive rate until no such increase can be made without missing an alignment. This simple heuristic performed essentially comparably to the integer program in our experiments, but, since the IP solved quickly, we present its results.

One other advantage to the IP formulation is that the false-positive rate from the LP relaxation is a lower bound on what can possibly be achieved; the simple greedy heuristic offers no such lower bound.

4 Experimental Results

Here, we present the results of experiments with our multiple seed selection framework in the context of protein alignments. Our goal is to identify collections of seed models which together have extremely high sensitivity to even moderately strong alignments, while admitting a very low false positive rate.

Since we pick seeds with a relatively small number of alignments, we run the serious risk of overtraining. In particular, the requirement that our set of seeds has 100 percent sensitivity on the training data need not require that it also have comparable sensitivity overall. In one example, the particular choice of training examples was apparently quite unrepresentative since a 100 percent sensitivity to this set of alignments still gave only 96 percent sensitivity on a testing set. (Or, presumably, the testing set may be unrepresentative.) As a simple way of exploring this, we examined what happened when we lowered the threshold on some seeds that were chosen by the integer program to modestly increase their false positive rates and sensitivity in the hope of still keeping very high sensitivity.

We first present simple experiments with vector seeds and with ungapped alignment seeds on a small sample of alignments discovered with BLASTP; in this section, we also allow for seed sets that miss a small number of the training alignments.

TABLE 1
Hit Rates for Optimal Seed Sets for Various Sets of Training Alignments when Applied to an Unrelated Test Set

Training alignments	False positive rate	Test alignment fraction hit
404	1/12,600	407/423 (96%)
403	1/9200	401/407 (98.5%)
409	1/8000	398/402 (99.0%)
394	1/10,700	395/400 (98.75%)
415	1/9500	410/416 (98.5%)

Then, we explore how well these seed sets do in hitting alignments that we did not use BLASTP to identify. Here, we note that our vector seed sets do not appear to do as well as BLASTP for sensitivity to alignments in general, but they do hit more alignments with high-scoring short regions; presumably, these alignments are more likely true.

4.1 Preliminary Experiments

We begin by exploring several sets of alignments generated using BLASTP. Our target score range for our alignments is BLASTP score between +40 and +60 (BLOSUM score +112 to +168). These moderate-scoring alignments can happen by chance, but also are often true. Alignments below this threshold are much more likely to be errors, while, in a database of proteins we used, such alignments are likely to happen to a random sequence by chance only one time in 10,000, according to BLASTP's statistics.

We begin by identifying a set of BLASTP alignments in this score range. To avoid overrepresenting certain families of alignments in our test set, we did an all-versus-all comparison of 8,654 human proteins from the SWISS-PROT database [20]. (We note that this is the same set of proteins and alignments we used in our previous vector seed work [3]. We have used this test set in part to confirm our belief that, while a single seed may not help much, in comparison to BLASTP, many seeds will be of assistance.) We then divided the proteins into families so that all alignments with BLASTP score greater than 100 are between two sequences in the same family and there are as many families as possible. We then chose 10 sets of alignments in our target score range such that, in each set of alignments, a particular family will only contribute at most eight alignments to that set. Note that, since our threshold for sharing family membership is a BLASTP score greater than 100 and the alignments we are seeking score between +40 and +60, many chosen alignments will be between members of different families. We divided the sets of alignments into five training sets and five testing sets. It is possible that the same alignments will occur in a training and testing set as we did not take any efforts to avoid this, though the set of possible alignments is large enough to make this a rare occurrence.

We note that we are using this somewhat complicated system specifically because we want to avoid imposing a preexisting bias on the set of alignments: Many true yet moderate-scoring alignments will be between proteins with different function or from different biological families. For the same reason, we have used alignments from dynamic programming as our standard, rather than structural alignments of known proteins or curated alignments because our goal is to improve the quality of heuristic alignments. Certainly, many of the alignments we consider will not be precise; still, a heuristic dynamic programming-based alignment that finds a hit between two proteins and then uses the same scoring matrix as BLASTP will find the exact same, potentially inaccurate, alignment as did BLASTP.

TABLE 2
Seeds and Thresholds Chosen by Integer Programming for 409 Test Alignments

Seed vector	IP Threshold	Lowered threshold
$(1,0,0,0,1,1,1)$	21	20
$(1,0,0,1,0,1,1)$	21	20
$(1,0,1,0,0,1,1)$	20	20
$(1,0,1,1,0,0,1)$	19	19
$(1,0,1,1,0,1)$	23	22
$(1,0,0,1,1,1)$	18	18
$(1,1,1,0,0,1)$	20	20
$(1,1,0,1,1)$	21	20

4.1.1 Multiple Vector Seeds

We then considered the set of all 35 vector patterns of length at most 7 that include three or four 1s (the *support* of the seed). We used this collection of vector patterns as we have seen no evidence that nonbinary seed vectors are preferable to binary ones for proteins and because it is more difficult to find hits to seeds with higher support than four due to the high number of needed hash table keys.

We computed the optimal set of thresholds for these vector seeds such that every alignment in a training set has a hit to at least one of the seeds, while minimizing the background rate of hits to the seeds and only using at most 10 vector patterns. Then, we examined the sensitivity of the chosen seeds for a training set to its corresponding test set. The results are found in Table 1. Some seed sets chosen showed signs of overtraining, but others were quite successful, where the chosen seeds work well for their training set as well and have low false positive rate.

We took the best seed set with near 100 percent sensitivity for both its training and testing data, which was the third of our experimental sets and used it in further experiments. This seed set is shown in Table 2. We note that this seed set has five times lower false positive rate $(1/8,000)$ than does BLASTP, while still hitting all of its testing alignments but four (which is not statistically significant from zero). We also considered a set of thresholds where we lowered the higher thresholds slightly to allow more hits and possibly avoid overtraining on the initial set of alignment. These altered thresholds are shown as well in Table 2 and give a total false positive rate of $1/6,900$. (This set of thresholds also hits all 402 test alignments for that instance.)

TABLE 3
Weakening Sensitivity to Testing Alignment Reduces Sensitivity on Training Alignments

Training alignments missed	False positive range	Testing alignment sensitivity range
0	1/7700 - 1/11500	98.5% - 99.8%
1	1/9800 - 1/12100	98.5% - 98.8%
2	1/11400 - 1/13100	98.3% - 98.7%
3	1/12700 - 1/14700	97.8% - 98.3%
4	1/13600 - 1/15900	96.9% - 98.5%

TABLE 4
Ungapped Alignment Seeds Offer Similar Performance to Vector Seeds

Training alignments missed	False positive range	Testing alignment sensitivity range
0	1/5200 - 1/6900	98.7% - 100%
1	1/7100 - 1/7600	98.7% - 99.0%
2	1/7500 - 1/9200	98.8% - 99.3%
3	1/10100 - 1/13900	97.8% - 99.0%
4	1/10600 - 1/13200	97.5% - 98.7%

4.1.2 A Weaker Requirement on the Sensitivity

As noted previously, we can alter our integer program so that it does not require 100 percent sensitivity on the training data set. We performed experiments on this formulation, using five subsets of the training alignments chosen as before, where we allowed between zero and five alignments from the training set to be missed by the seed set. We show results in Table 3, using again a randomly chosen testing set for each training set. The training data sets varied in size from 304 to 415, while the testing sets ranged from 392 to 407 in size.

Unsurprisingly, if we did not hit all alignments in the training set, we often miss alignments in the testing set as well. However, the ranges of the sensitivities we saw in testing data for the seed sets picked allowing some misses in the training data were much less wide, suggesting that there may be fewer seed thresholds lowered merely to accommodate a single outlier in the training data. As such, if slightly lower sensitivity is acceptable, this approach may give much more predictable results than training to require all alignments to be hit.

4.1.3 Multiple Ungapped Alignment Seeds

Ungapped alignment seeds can be seen as breaking the model we have for alignment speed. The most straightforward implementation of ungapped alignment seeds would involve a hash table keyed on the letters corresponding to the positions in the bounds vector b, where there is a nontrivial lower bound on the score of a position. Still, even after the first step, where we identified pairs of positions satisfying the minimum bounds scores, we still need another test to verify that a pair of positions satisfies the requirement of the dot product of the local alignment score with the vector v of positional multipliers being higher than the threshold. Similar limitations affect any such two-phase seed, such as requiring that two hypothetically aligned positions satisfy two vector seeds at once.

If we assume, however, that testing a hit to the simple hash-table to verify if the dot product of the local alignment score with the vector of multipliers v has score greater than the threshold T so rapidly that we can throw out misses without having to count them, then we return to the case from before, where we need count only the fraction of positions expected to pass both levels of filtration. This assumption may be appropriate, assuming that the small amount of time taken to throw out a hash-table hit that does not satisfy the dot product threshold is much, much smaller than the amount of time needed to throw out a hit to the whole ungapped alignment seed that still does not make a good local alignment.

With this in mind, we tested our set of moderate alignments on a simple collection of ungapped alignment seed patterns to identify whether ungapped alignment seeds form a potentially superior seed filtering approach to vector seeds. Of course, since they include vector seeds as a special case, this is trivial, but our interest is primarily whether the advantage of ungapped alignments is large enough to merit their consideration over that of vector seeds.

In our experiments, we used ungapped alignment seeds where the vector of score lower bounds consisted of only the values 0 and $-\infty$ (which results in no score restriction); we also allowed the vector of pairwise multipliers to only be the all-ones vector. This simple approach, which was used independently in the multiple aligner of Brown and Hudek [15] and in the tPatternHunter protein aligner [7], simply requires a good local region, with certain specified positions having positive score. We required that the bounds vector have at most four active positions and considered seed lengths between three and six. Note that, in this model, the bounds vector $(0, 0, 0, -\infty)$ behaves quite differently than the bounds vector $(0, 0, 0)$ because we will be adding pairwise scores of four positions in the former case and three in the latter.

The results of our experiment are shown in Table 4. We used the same testing and training data sets as for Table 3. In general, these results are slightly worse than the results of our original experiments with vector seeds when we require 100 percent sensitivity to testing data, but improve when we allow some misses in the training data. Typical false positive rates on the order of $1/10,000$ are common with testing sensitivity of approximately 99 percent, as before; again, the corresponding false positive rate for BLASTP's seed is approximately $1/1,600$.

A positive note to the ungapped alignment seeds is that there seems to be less overtraining: As the training sensitivity is allowed to go down slightly, the testing sensitivity does not plummet as quickly as for vector seeds.

One reason for this is that an ungapped alignment seed, both times they have been implemented [7], [15], still requires high-scoring short local alignment around the seed. As we show in the next section, focusing on very narrow alignments in seeding may be inappropriate and

TABLE 5
Hits in Locally Good Regions of Alignments

Subregion score threshold	Seed $((1,1,1),13)$	Multiple seeds	Multiple seeds Thresholds raised	Seed $((1,1,1),15)$
Any region	99.3%	96.5%	97.0%	91.8%
+25	77.8%	73.2%	75.2%	63.9%
+30	47.7%	46.7%	47.6%	41.1%
+35	24.3%	24.3%	24.4%	23.0%
+40	13.0%	13.0%	13.0%	12.8%

Shown are the fraction of 2950 alignments found with a hit to a seed that is found in a region of length 10 whose total subalignment score is above a given threshold.

one should instead focus on longer windows around a hit before discarding it with a filter.

4.2 A Broader Set of Alignments

Returning to our set of vector seeds from Table 2, we then considered a larger set of alignments in our target range of good, but not great scores to verify if the advantage of multiple seeds still holds. We used the Smith-Waterman algorithm to compute all alignments between pairs of a 1,000-sequence subset of our protein data set and computed how many of them were not found by BLASTP. Only 970 out of 2,950 Smith-Waterman alignments with BLOSUM62 score between +112 and +168 had been identified by BLASTP, even though alignments in this score range would have happened by chance only one time in 10,000 according to BLASTP's statistics.

Almost all of these 2,950 alignments, 2,942, had a hit to the BLASTP default seed. Despite this, however, only 970 actually built a successful BLASTP alignment. Our set of eight seeds had hits to 1,939 of the 1,980 that did not build a BLASTP alignment and to 955 of the 970 that did build a BLASTP alignment, so, at first glance, the situation does not look good. However, the difference between having a hit and having a hit in a good region of the alignment is where we are able to show substantial improvement.

The discrepancy between hits and alignments comes because the BLASTP seed can have a hit in a bad part of the alignment, which is filtered out. Typically, such hits occur in a region where the source of positive score is quite short, which is much more likely with an unspaced seed than with a spaced seed. We looked at all of the regions of length 10 amino acids of alignments that included a hit to a seed (either the BLASTP seed or one of the multiple seeds), and assigned the best score of such a region to that alignment; if no ungapped region of length 10 surrounded a hit, we assumed it would certainly be filtered out. The data are shown in Table 5 and show that of the alignments hit by the spaced seeds, they are hit in regions that are essentially identical in conservation to where the BLASTP seed hits them. For example, 47.7 percent of the alignments contain a 10-amino acid region around a hit to the ((1, 1, 1), 13) seed with BLOSUM score at least +30, while 46.7 percent contain such a region surrounding a hit to one of the multiple seeds with higher threshold. If we use the lower thresholds that allow slightly more false positives, their performance is actually slightly better than BLASTP's.

Table 5 also shows that the higher-threshold seed ((1, 1, 1), 15), which has a worse false positive rate (1/5,700) than our ensembles of seeds, performs substantially worse: Namely, only 64 percent of the alignments have a hit to the single seed found in a region with local score above +25, while 73 percent of the alignments have a hit to one of the multiple seeds with this property. This single seed strategy is clearly worse than the multiple seed strategy of comparable false positive rate and the optimized seeds perform comparably to BLASTP in identifying the alignments that actually have a core conserved region.

Our experiments show that multiple seed models can have an impact on local alignment of protein sequences. Using many spaced seeds, which we picked by optimizing an integer program, we find seed models with a comparable chance of finding a good hit in a moderate-scoring alignment than does the BLASTP seed, with four to five times fewer noise hits. The difficulty with the BLASTP seed is that it not only has more junk hits and more hits in overlapping places, it also has more hits in short regions of true alignments, which are likely to be filtered and thrown out.

5 Conclusions

We have given a theoretical framework to the problem of using spaced seeds for protein homology search detection. Our result shows that using multiple vector or ungapped alignment seeds can give sensitivity to good parts of local protein alignments essentially comparable to BLASTP, while reducing the false positive rate of the search algorithm by a factor of four to five.

Our set of vector seeds is chosen by optimizing an integer programming framework for choosing multiple seeds when we want 100 percent sensitivity to a collection of training alignments. The framework is general enough to accommodate many extensions, such as requiring a fixed amount of sensitivity on the training (not only 100 percent), allowing only a small number of seeds to be chosen or allowing for many different sorts of seeding strategies. We have mostly used it to optimize sets of vector seeds because they encapsulate an approach to homology search for nucleotides that has been very successful.

One difficulty with our approach is that it relies on a theoretical estimate of the runtime of a homology search program: namely, that the program will take time proportional to the number of false positives found by the seeding method. As seeding methods become more complex, such as the two-step ungapped alignment seeds, it may become harder to identify what a "false positive" is, in particular, if a false positive fits through one step of a filter, but is quickly discarded before the next step, should it count toward the estimated runtime? Using our framework, we identified a set of seeds for moderate-scoring protein alignments whose total false positive rate in random sequence is four-to-five times lower than the default BLASTP seed. This set of seeds had hits to slightly fewer alignments in a test set of moderate-scoring alignments found by the Smith-Waterman algorithm than found by BLASTP; however, the BLASTP seeds hit subregions of these alignments that were actually slightly worse than hit by the spaced seeds. Hence, given the filtering used by BLASTP, we expect that the two alignment strategies would give comparable sensitivity, while the spaced seeds give four times fewer false hits.

ACKNOWLEDGMENTS

The author would like to thank Ming Li for introducing him to the idea of spaced seeds. This work is supported by the Natural Science and Engineering Research Council of Canada and by the Human Frontier Science Program. A preliminary version of this paper [21] appeared at the Workshop on Algorithms in Bioinformatics, held in Bergen, Norway, in September, 2004.

REFERENCES

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman, "Basic Local Alignment Search Tool," *J. Molecular Biology,* vol. 215, no. 3, pp. 403-410, 1990.

[2] B. Ma, J. Tromp, and M. Li, "PatternHunter: Faster and More Sensitive Homology Search," *Bioinformatics,* vol. 18, no. 3, pp. 440-445, Mar. 2002.

[3] B. Brejova, D. Brown, and T. Vinar, "Vector Seeds: An Extension to Spaced Seeds Allows Substantial Improvements in Sensitivity and Specificity," *Proc. Third Ann. Workshop Algorithms in Bioinformatics,* pp. 39-54, 2003.

[4] M. Li, B. Ma, D. Kisman, and J. Tromp, "Patternhunter II: Highly Sensitive and Fast Homology Search," *J. Bioinformatics and Computational Biology,* vol. 2, no. 3, pp. 419-439, 2004.

[5] J. Xu, D. Brown, M. Li, and B. Ma, "Optimizing Multiple Spaced Seeds for Homology Search," *Proc. 15th Ann. Symp. Combinatorial Pattern Matching,* pp. 47-58, 2004.

[6] Y. Sun and J. Buhler, "Designing Multiple Simultaneous Seeds for DNA Similarity Search," *Proc. Eighth Ann. Int'l Conf. Computational Biology,* pp. 76-84, 2004.

[7] D. Kisman, M. Li, B. Ma, and L. Wang, "TPatternHunter: Gapped, Fast and Sensitive Translated Homology Search," *Bioinformatics,* 2004.

[8] T. Smith and M. Waterman, "Identification of Common Molecular Subsequences," *J. Molecular Biology,* vol. 147, pp. 195-197, 1981.

[9] B. Brejova, D. Brown, and T. Vinar, "Vector Seeds: An Extension to Spaced Seeds," *J. Computer and System Sciences,* 2005, pending publication.

[10] J. Buhler, U. Keich, and Y. Sun, "Designing Seeds for Similarity Search in Genomic DNA," *Proc. Seventh Ann. Int'l Conf. Computational Biology,* pp. 67-75, 2003.

[11] B. Brejova, D. Brown, and T. Vinar, "Optimal Spaced Seeds for Homologous Coding Regions," *J. Bioinformatics and Computational Biology,* vol. 1, pp. 595-610, Jan. 2004.

[12] U. Keich, M. Li, B. Ma, and J. Tromp, "On Spaced Seeds for Similarity Search," *Discrete Applied Math.,* vol. 138, pp. 253-263, 2004.

[13] K.P. Choi, F. Zeng, and L. Zhang, "Good Spaced Seeds for Homology Search," *Bioinformatics,* vol. 20, no. 7, pp. 1053-1059, 2004.

[14] G. Kucherov, L. Noé, and Y. Ponty, "Estimating Seed Sensitivity on Homogeneous Alignments," *Proc. Fourth IEEE Int'l Symp. BioInformatics and BioEng.,* pp. 387-394, 2004.

[15] D. Brown and A. Hudek, "New Algorithms for Multiple DNA Sequence Alignment," *Proc. Fourth Ann. Workshop Algorithms in Bioinformatics,* pp. 314-326, 2004.

[16] M. Csürös, "Performing Local Similarity Searches with Variable Length Seeds," *Proc. 15th Ann. Symp. Combinatorial Pattern Matching,* pp. 373-387, 2004.

[17] K. Choi and L. Zhang, "Sensitive Analysis and Efficient Method for Identifying Optimal Spaced Seeds," *J. Computer and System Sciences,* vol. 68, pp. 22-40, 2004.

[18] G. Kucherov, L. Noé, and Y. Ponty, "Multiseed Lossless Filtration," *Proc. 15th Ann. Symp. Combinatorial Pattern Matching,* pp. 297-310, 2004.

[19] U. Feige, "A Threshold of $\ln n$ for Approximating Set Cover," *J. ACM,* vol. 45, pp. 634-652, 1998.

[20] A. Bairoch and R. Apweiler, "The SWISS-PROT Protein Sequence Database and Its Supplement TrEMBL in 2000," *Nucleic Acids Research,* vol. 28, no. 1, pp. 45-48, 2000.

[21] D. Brown, "Multiple Vector Seeds for Protein Alignment," *Proc. Fourth Ann. Workshop Algorithms in Bioinformatics,* pp. 170-181, 2004.

Daniel G. Brown received the undergraduate degree in mathematics with computer science from the Massachusetts Institute of Technology in 1995 and the PhD degree in computer science from Cornell University in 2000. He then spent a year as a research scientist at the Whitehead Institute/MIT Center for Genome Research in Cambridge, Massachusetts, working on the Human and Mouse Genome Projects. Since 2001, he has been an assistant professor in the School of Computer Science at the University of Waterloo.