

Designing Multiple Simultaneous Seeds for DNA Similarity Search

Yanni Sun
yanni@cse.wustl.edu
Department of Computer Science
and Engineering
Washington University
St. Louis, MO 63130

Jeremy Buhler
jbuhler@cse.wustl.edu
Department of Computer Science
and Engineering
Washington University
St. Louis, MO 63130

ABSTRACT

The challenge of similarity search in massive DNA sequence databases has inspired major changes in BLAST-style alignment tools, which accelerate search by inspecting only pairs of sequences sharing a common short “seed,” or pattern of matching residues. Some of these changes raise the possibility of improving search performance by probing sequence pairs with several distinct seeds, any one of which is sufficient for a seed match. However, designing a set of seeds to maximize their combined sensitivity to biologically meaningful sequence alignments is computationally difficult, even given recent advances [16, 6] in designing single seeds.

This work describes algorithmic improvements to seed design that address the problem of designing a set of n seeds to be used simultaneously. We give a new local search method to optimize the sensitivity of seed sets. The method relies on efficient incremental computation of the probability that an alignment contains a match to a seed π , given that it has already failed to match any of the seeds in a set Π . We demonstrate experimentally that multi-seed designs, even with relatively few seeds, can be significantly more sensitive than even optimized single-seed designs.

Categories and Subject Descriptors

F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—*pattern matching*; H.3.3 [Database Management]: Information Search and Retrieval—*information filtering, search process*

General Terms

Algorithms, Design, Performance

Keywords

biosequence comparison, similarity search, seed design, genomic DNA, Mandala

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

RECOMB'04, March 27–31, 2004, San Diego, California, USA.
Copyright 2004 ACM 1-58113-755-9/04/0003 ...\$5.00.

1. INTRODUCTION

Similarity search using large DNA databases is critical to identifying biologically meaningful features in genomes. Comparing a database of known features to an unannotated genome can help identify genes and remains the method of choice for finding non-gene features such as transposable elements [23] and putative regulatory regions [10, 15]. Pairwise comparison of entire mammalian genomes has been used to support gene-structure prediction [14], long-range orthology detection [20], and inference of genome rearrangements [19].

Modern biosequence databases place substantial computational demands on tools for fast detection of similarity. These demands, whose growth parallels the exponentially increasing size of the databases [17], have revived innovation in heuristics for fast similarity search in DNA [21, 12, 16, 6]. All of these heuristics, like their ancestors BLASTN [1] and FASTN [18], work by first identifying short *seed matches* between DNA sequences, then more carefully aligning those pairs of sequences that exhibit one or more seed matches.

In this work, a *seed* π is defined to be an ordered list of indices $\{x_1 \dots x_w\}$. To ensure that each possible seed has a unique representation in this notation, we fix $x_1 = 0$ for all seeds. We say that two sequences S and T exhibit a *seed match* at offsets i and j if, for $1 \leq k \leq w$, $S[i + x_k] = T[j + x_k]$. The number of inspected positions w is the *weight* of π , while the distance $s = x_w - x_1 + 1$ is its *span*. Traditionally, seed matching heuristics have used the contiguous seed $\pi_c = \{0, 1, \dots, w-1\}$, but more recent work [16, 6, 3] has shown the desirability of utilizing seeds which inspect a discontinuous set of sequence positions.

In a large genomic DNA comparison, the seed matching phase, which entails a complete scan of the sequence database, can easily take 50% or more of total computation time. One way to reduce this cost is to move it offline by preprocessing the database. Preprocessing-based search engines like FLASH [7] and BLAT [12] index a database D so that all seed matches between D and some query sequence q can be found in time proportional to q 's length plus the number of matches. Depending on the density of seed matches, preprocessed searches may cost an order of magnitude less [12] than a linear scan of D . A second way to accelerate seed matching is to delegate it to specialized hardware, such as a field-programmable gate array (FPGA) [9] or network processor [22]. Such hardware, combined with high-speed I/O from the database's storage, could greatly reduce the time needed to scan D while freeing the host processor

to investigate any seed matches in parallel. Preprocessing and hardware-accelerated seeded alignment are therefore attractive options for designing new, faster similarity search tools.

Preprocessing- and hardware-based search share one feature of critical importance for algorithm design: both approaches can efficiently utilize multiple seeds at once. Let $\Pi = \{\pi_1 \dots \pi_n\}$ be a set of seeds, and suppose we seek all seed matches induced by *at least one seed* from Π . Preprocessing produces one index for each seed $\pi \in \Pi$, requiring n lookups instead of one; however, for reasonable n , this cost is still negligible compared to a linear scan of the database. Hardware-accelerated search engines can exploit the fine-grained parallelism and wide data paths of specialized hardware to perform n single-seed searches in parallel on the database as it streams through the device.

Alternative search technologies open up an algorithmic design space of multi-seed heuristics, but navigating this space is nontrivial. For single seeds, the PatternHunter search engine [16] and our own work on the Mandala seed design software [6] have shown the utility of designing a seed to optimize sensitivity in a probabilistic model \mathcal{M} of the alignments being sought. However, Mandala’s design methods, while applicable to multi-seed design, scale poorly with the size n of the seed set. When faced with a need for multiple seeds, previous work has generally resorted to random seed choices [7, 4], which provide no real guarantees of high sensitivity for the small n (3–5) of most use to resource-constrained hardware or indexing methods. We have therefore sought a robust, efficient approach to multi-seed design in the spirit of successful methods for single seeds.

This work describes methods for designing locally optimal sets of multiple seeds for simultaneous use. Formally, we address the following problem:

PROBLEM 1.1. *Let \mathcal{M} be a Markov model that generates aligned pairs of biosequences (without gaps), and let parameters w , s , and n be given. Find a set Π of n seeds, each of weight w and span at most s , that maximizes the probability that at least one seed from Π matches an alignment chosen at random from \mathcal{M} .*

Our approach to this problem greedily covers the highest-probability alignments of \mathcal{M} with seeds that match them. To support this strategy, we show how to compute the probability that a seed π matches an alignment from \mathcal{M} , conditional on some other set Π of seeds failing to match. Our new methods considerably reduce the cost of designing multi-seed sets compared to the local search algorithm of [6]. Empirical validation shows that multi-seed designs, even with maximum spans as short as 22, can exhibit substantially improved sensitivity relative to single optimized seeds of the same or even lower weight.

The remainder of this work is organized as follows. Section 2 presents a brief rationale for multi-seed design and describes the greedy covering approach, along with its extension to a beam search strategy. Section 3 describes our enhancements to exact and Monte Carlo methods for seed evaluation to efficiently compute conditional probabilities. Section 4 evaluates the new methods for multi-seed design and validates the predicted sensitivity and specificity of designs computed by these methods on a large mammalian genome comparison. Finally, Section 5 concludes and suggests directions for future work.

2. MULTI-SEED DESIGN: RATIONALE AND APPROACH

In this section, we first present a qualitative argument in favor of multi-seed designs. We then formally define a sensitivity measure for sets of seeds and describe how we improve on the local search optimization of [6] to more efficiently probe multi-seed design space.

2.1 Why Multi-Seed Design?

Seed-matching heuristics must optimize a tradeoff between sensitivity – measured by the true positive rate – and specificity – measured by one minus the false positive rate. The true positive rate is the chance that an alignment of interest contains a seed match, while the false positive rate is the probability of a seed match occurring by chance in sequences that lack a biologically meaningful alignment.

The specificity of a single seed π is largely controlled by its weight w . Given two i.i.d. random DNA sequences with equal base frequencies, a false positive match to π occurs about once in 4^w aligned bases. The only way to substantially reduce π ’s false positive rate is to increase its weight; however, this increase is usually detrimental to sensitivity because it further constrains the set of alignments that can be detected by π .

Multiple seeds provide a potentially more attractive way to trade sensitivity for specificity. Let π be a seed of weight w , and suppose there exists a set Π of seeds, each of weight $w' > w$, that together are as sensitive as π . Under the i.i.d. model, the total expected number of chance matches to Π is at most $|\Pi|/4^{w'}$. If $|\Pi| < 4^{w'-w}$, then the set Π causes fewer expected false positives than π . Although this analysis uses an oversimplified sequence model, its qualitative conclusion remains significant: adding seeds to a seeded alignment algorithm at most *linearly* increases its false positive rate, while reducing the seed weight produces an *exponential* increase.

Evidence from real sequence comparisons suggests that a set Π of seeds can indeed be more sensitive than a single, shorter seed π , even when π is optimal for its weight. In [6], we report two seeds of weight 12 that together proved *more* sensitive in practice than the best single seed of weight 11 in a comparison of human and mouse noncoding DNA. While the false positive rate of the seed pair was lower than that of a single, shorter seed, the increased cost of checking for matches to two seeds caused a search using them to require more time to compute overall. However, this limitation can be circumvented by search techniques, such as the aforementioned indexing and hardware acceleration, that parallelize or move offline the computational burden of multi-seed matching.

2.2 Definition of Seed Sensitivity

We measure the sensitivity of a seed with respect to a probabilistic alignment model \mathcal{M} , which describes the distribution of matches and substitutions in ungapped alignments of some fixed length ℓ . Abstractly, a sample from \mathcal{M} is a bit string α of length ℓ , with 1’s where the aligned sequences match and 0’s where they do not. \mathcal{M} , which takes the form of a (possibly nonstationary) k th-order Markov process, can be trained on biologically meaningful alignments obtained from real genomic sequence comparisons.

We say that a seed $\pi = \{x_1 \dots x_w\}$ *matches* an alignment α if, for some offset i and $1 \leq j \leq \ell$, $\alpha[i + x_j] = 1$. We

denote this event as $E_\pi(\alpha)$ and the complementary event as $\overline{E_\pi}(\alpha)$. The *match probability* of π in \mathcal{M} is given by

$$\Pr_{\alpha \sim \mathcal{M}}(E_\pi(\alpha)). \quad (1)$$

We subsequently drop the alignment α from our event notation when referring to events over a random alignment from \mathcal{M} . *Optimizing* a seed means choosing its non-fixed positions $x_2 \dots x_w$ to maximize its match probability under \mathcal{M} .

We say that a set Π of seeds matches α if at least one of its component seeds matches. We denote this disjunction of events as $E_\Pi(\alpha)$ and define the combined match probability for Π using Definition (1), substituting E_Π for E_π .

2.3 Greedy Covering Algorithm for Multi-Seed Design

Finding an optimal set of seeds for a model \mathcal{M} is challenging because, at least for limited alignment lengths ℓ , seed design space appears to lack an easily exploitable structure for optimization. For this reason, existing work relies either on exhaustive search [16] or on local search techniques [6] to find globally and locally optimal seeds, respectively. Exhaustive search is infeasible for multi-seed designs because the search space grows exponentially with the number of seeds n , so we focus on improving local search.

Mandala’s local search algorithm for multi-seed design works as follows. Given parameters w and s , the search starts with a set Π_0 of n randomly chosen seeds, each with common weight w and span at most s . The algorithm then chooses i and j , $1 \leq i \leq n$ and $2 \leq j \leq w$, and finds the best seed set Π_1 in the neighborhood of sets derived from Π_0 by deleting position x_j of the i th seed $\pi_i \in \Pi_0$ and replacing it with a position between 1 and $s - 1$ not currently inspected by π_i . This process iterates cyclically through i and j until no further local improvement is possible. To choose a single optimized seed, we simply run the above procedure with $n = 1$.

Empirically, two factors increase the running time of Mandala’s local search precipitously with the set size n . Firstly, when evaluating match probabilities using the fast dynamic programming algorithm of [6], the cost of evaluating the match probability $\Pr(E_\Pi)$ for a seed set Π is proportional to the size of a deterministic finite automaton (DFA) A_Π that accepts all bit strings containing a match to Π . We have observed that, for small n , adding one seed to Π roughly doubles the final size of A_Π after DFA minimization. Secondly, the number of evaluations needed to converge to a local optimum grows superlinearly in n . Growth of $\Omega(n)$ is expected because each new seed adds w inspected positions that can be changed by local search. However, we find empirically that optimizing larger seed sets Π also requires more complete iterations over all seeds in Π before the search converges.

To address concerns both of slow convergence and of evaluating large seed sets, we have developed a greedy covering heuristic for choosing seed sets. Given a partial seed set Π_0 of size $n' < n$, we form a set Π of size $n' + 1$ by choosing the next seed π to maximize the *conditional match probability* $\Pr(E_\pi | \overline{E_{\Pi_0}})$. Each step of the heuristic attempts to cover the highest-probability alignments not already matched by some seed in the current partial set. Starting from a single locally optimal seed, $n - 1$ iterations of greedy covering produce a seed set of size n .

Greedy covering can run faster than Mandala’s local search for two reasons. First, most of its seed set evaluations are performed on partial sets of size $< n$, while the old local search always evaluates sets of the full size n . Second, because each covering step optimizes only a single seed, the number of evaluations needed remains roughly constant per seed added and so grows linearly with n .

2.4 Extension to Beam Search

There is no guarantee that a seed set Π derived by greedy covering will have an aggregate match probability as high as a set in which all seeds were designed simultaneously. To improve the quality of the chosen seed set, we can simply restart the greedy covering algorithm several times from different initial seeds. A more aggressive, more compute-intensive strategy uses a beam search [2] to extend several intermediate seed sets, each of size $n' < n$.

Beam search is initialized by finding a number of locally optimal single seeds. The best b of these seeds are saved and used in the next round of optimization. For each saved seed π_0 , we find N seeds π (for some N), each of which locally optimizes $\Pr(E_\pi | \overline{E_{\pi_0}})$. The b seed pairs $\{\pi_0, \pi\}$ with highest combined match probability over all $b \cdot N$ pairs are again saved, and the search proceeds to find N candidate third seeds for each pair. This process iterates until all seed sets in the beam have size n , at which time the best seed set overall is chosen.

3. COMPUTING CONDITIONAL MATCH PROBABILITIES

Our approach to multi-seed design requires that we compute, for a set of seeds Π and an additional seed π , the conditional match probability $\Pr(E_\pi | \overline{E_\Pi})$ for an alignment model \mathcal{M} . In this section, we describe enhancements to the methods of [6] to efficiently compute conditional match probabilities. A single evaluation of $\Pr(E_\pi | \overline{E_\Pi})$ is no faster than computing $\Pr(E_{\Pi \cup \pi})$. However, we can exploit the fact that optimization performs many evaluations with a common set Π of fixed seeds by factoring redundant work associated with Π out of each evaluation.

3.1 Exact Computation via DFAs

For seeds π whose span is not much greater (about 10–15) than their weight, the most efficient method known [6] to compute $\Pr(E_\pi)$ uses dynamic programming over a finite automaton. Briefly, we first construct a DFA A_π that accepts precisely those alignments (represented as bit strings) containing a seed match to π . We then compute by dynamic programming the probability that A_π accepts a random alignment of length ℓ from model \mathcal{M} . For a k th-order Markov model, we can compute the match probability for a seed of weight w and span s in time $\Theta(\ell 2^k |A_\pi|)$, where $|A_\pi| = \Theta(w 2^{s-w})$. This algorithm extends straightforwardly to compute $\Pr(E_\Pi)$ for a set of seeds Π . In practice, minimizing A_π before dynamic programming greatly accelerates the computation, provided it is done by a subquadratic-time method such as Hopcroft’s algorithm [11].

Without modifying the methods of [6], we can compute $\Pr(E_\pi | \overline{E_\Pi})$ for a seed π and a set of seeds Π as

$$\Pr(E_\pi | \overline{E_\Pi}) = \frac{\Pr(E_{\Pi \cup \pi}) - \Pr(E_\Pi)}{1 - \Pr(E_\Pi)}. \quad (2)$$

In the context of local search, Π is fixed, while π may change several hundred times over the course of optimization. We therefore seek to factor out those parts of the computation that do not depend on π , rather than repeating them for each π .

Let A_π and A_Π be automata accepting alignments containing a match to π and Π , respectively. We first form the complement automaton $\overline{A_\Pi}$ that accepts iff A_Π does not. We then form the cross-product $A_c = A_\pi \otimes \overline{A_\Pi}$, which accepts iff A_π accepts but A_Π does not. By construction, A_c accepts an alignment from \mathcal{M} with probability $p = \Pr(E_\pi \cap \overline{E_\Pi})$, so that $p/(1 - \Pr(E_\Pi))$ is the probability we want. $\overline{A_\Pi}$ can be computed once and cached before optimization.

The cross-product construction avoids the significant costs associated with explicit construction and minimization of $A_{\Pi \cup \pi}$ for each new seed π . In exchange, it demands construction of A_π and of the cross product A_c . However, if this cross product is constructed lazily (i.e. including only states reachable from its initial state) from minimized A_Π and A_π , we have in practice found that the resulting DFA is typically at most twice the size of the minimized $A_{\Pi \cup \pi}$ and costs less to construct given A_Π .

3.2 Conditional Sampling for Monte Carlo

For a seed π of weight w and span s , the cost of computing $\Pr(E_\pi)$ is proportional to $w2^{s-w}$. While this cost is manageable (well under a second) when, e.g., $w = 11$ and $s \leq 22$, increasing $s - w$ to around 20 increases the cost 1000-fold. Avoiding longer spans altogether seems unwise when exploring multi-seed designs – packing multiple seeds into a short span could more quickly incur diminishing returns, as each seed samples almost the same positions as the others.

To explore the regime of longer spans, we turn to a Monte Carlo approach to estimate match probabilities. Monte Carlo estimation computes $\Pr(E_\Pi)$ in a model \mathcal{M} by generating T similarities at random from \mathcal{M} and computing the fraction that contain a seed match to Π . We set $T = 2 \times 10^6$ to estimate match probabilities to about three digits of accuracy.

Computing $\Pr(E_{\Pi \cup \pi})$ (and hence, using Equation (2), $\Pr(E_\pi | \overline{E_\Pi})$) by direct Monte Carlo is feasible but requires checking every seed of $\Pi \cup \pi$ against each sampled alignment. Moreover, this approach is not ideal for differentiating among seeds π given a fixed set Π , since relatively few samples fall into the event subspace $\overline{E_\Pi}$. A more desirable approach focuses sampling on this subspace, permitting either more accurate comparison of different π 's by their conditional probabilities or fewer samples, and hence less computation, to achieve a given level of accuracy.

Sampling from the subspace $\overline{E_\Pi}$, up to some target number of samples T , can be implemented with rejection sampling, but this technique still incurs the computational cost of generating alignments from all of \mathcal{M} . Because only 1 in about $1/(1 - \Pr(E_\Pi))$ samples is accepted, rejection sampling is an inefficient solution when Π covers a substantial fraction of \mathcal{M} 's probability mass. A more robust approach to Monte Carlo again exploits the fact that the fixed seeds Π remain constant over a large number of evaluations of different Π 's. By paying a preprocessing penalty, we can sample directly from the subspace $\overline{E_\Pi}$ at speeds close to that of sampling from the full space of \mathcal{M} .

Let A_Π be the automaton of the previous section, which accepts alignments containing a seed match to Π . Every

alignment α sampled from \mathcal{M} traces a path through A_Π ; if α is in $\overline{E_\Pi}$, the path ends somewhere other than A_Π 's final state f (which is unique by the construction in [6]) after ℓ steps. We wish to sample such paths, and *only* such paths, with probability proportional to their likelihoods under \mathcal{M} .

Suppose that we have generated t bits of α , traversing a path that ends at state q of A_Π , and let r^0 and r^1 be the targets of transitions from q on 0 and 1, respectively. To generate α 's next bit, we need to sample from its distribution *conditional on the path not ending at f after ℓ bits*. Since \mathcal{M} is a k th-order Markov process, the next bit of α also depends on the history h of the last k bits generated. Hence, for $b \in \{0, 1\}$, we must compute

$$\Pr(r_{t+1}^b | q_t, h_t, \overline{f_\ell}),$$

where event q_t (resp. r_{t+1}^b) denotes that being in state q (resp. r^b) after t (resp. $t + 1$) bits, and the history h_t lists the last k of the first t generated bits.

We have by Bayes' theorem that

$$\begin{aligned} \Pr(r_{t+1}^b | q_t, h_t, \overline{f_\ell}) &= \frac{\Pr(\overline{f_\ell} | r_{t+1}^b, q_t, h_t) \Pr(r_{t+1}^b | q_t, h_t)}{\Pr(\overline{f_\ell} | q_t, h_t)}. \end{aligned} \quad (3)$$

For every state q with a transition to r^b on bit b , the right-hand probability in the numerator is identical and is equivalent to $\Pr(b | h_t)$, the probability that model \mathcal{M} generates a b bit given history h_t . Moreover, knowing that q_t and r_{t+1}^b occur implies that the $t + 1$ st bit is b . Let h_{t+1}^b be the concatenation of b with the $k - 1$ most recent bits of h_t ; then

$$\Pr(\overline{f_\ell} | r_{t+1}^b, q_t, h_t) = \Pr(\overline{f_\ell} | r_{t+1}^b, h_{t+1}^b).$$

Finally, the denominator of (3) is a normalizing constant γ that ensures $\sum_b \Pr(r_{t+1}^b | q_t, h_t, \overline{f_\ell}) = 1$. Conclude that

$$\begin{aligned} \Pr(r_{t+1}^b | q_t, h_t, \overline{f_\ell}) &= \frac{[1 - \Pr(f_\ell | r_{t+1}^b, h_{t+1}^b)] \Pr(b | h_t)}{\gamma}. \end{aligned}$$

It remains to compute $\Pr(f_\ell | q_t, h_t)$ for any state q of A_Π and k -bit history h and any $t \leq \ell$. Once again, let r^0 and r^1 be the targets of transitions from q on 0 and 1. We can compute the desired probability via the following backward recurrence:

$$\begin{aligned} \Pr(f_\ell | q_t, h_t) &= \sum_b \Pr(f_\ell | r_{t+1}^b, h_{t+1}^b) \Pr(r_{t+1}^b | q_t, h_t) \\ &= \sum_b \Pr(f_\ell | r_{t+1}^b, h_{t+1}^b) \Pr(b | h_t). \end{aligned}$$

The base cases of the recurrence are that $\Pr(f_\ell | f_\ell, h_\ell) = 1$, while $\Pr(f_\ell | q_\ell, h_\ell) = 0$ for $q \neq f$.

We can now generate samples from \mathcal{M} conditioned on $\overline{E_\Pi}$ as follows. We first compute A_Π , then precompute and store $p(q, h, t) = \Pr(r_{t+1}^1 | q_t, h_t, \overline{f_\ell})$ for each state q , each k -bit history h , and each length $t \leq \ell$. We generate successive bits of α while tracing its path in A_Π . If we reach state q after t bits with history h , the next bit of α is 1 with probability $p(q, h, t)$. Running this procedure for ℓ steps generates one random alignment.

As noted above, the automaton A_Π will be expensive to compute if use of Monte Carlo is warranted. However, given

that even accelerated Monte Carlo with $T = 2 \times 10^6$ requires at least several seconds per evaluation, even an up-front cost of several minutes to compute A_{Π} can be amortized over the hundreds of evaluations needed for one local search. We note that this optimization works best on machines with high memory bandwidth, as the sampling procedure requires nearly random access to the table $p()$ of worst-case size $\Theta(\ell 2^k |A_{\Pi}|)$, which by [6] is $\Theta(\ell w 2^{s-w+k})$.

4. RESULTS

In this section, we characterize the behavior of the greedy covering algorithm and evaluate its performance relative to the local search of [6]. We then apply our new methods to design seed sets for a large-scale mammalian genomic DNA comparison to investigate the sensitivity and specificity of multi-seed designs.

4.1 Performance of Greedy Covering

We evaluated the performance of the greedy covering algorithm, along with the more aggressive beam search strategy, using a model \mathcal{M}_{nc} derived from pairs of large orthologous blocks of the human and mouse genomes. Ungapped alignments of length $\ell = 64$ and identity between 70 and 75% were sampled from the noncoding portions of these sequences using the randomized procedure described in [6], then used to train a first-order Markov model. Further details of model construction are given in Appendix A.

To validate the greedy covering approach, we designed sets of n seeds, for $1 \leq n \leq 5$. Seeds were constrained to have weight 11 and span at most 22, so that candidate seed sets could be efficiently evaluated by dynamic programming. A second set of design experiments produced seeds of weight 11 and span at most 32 using Monte Carlo evaluation. Seed sets were derived by three different methods: greedy covering, beam search with a beam size $b = 12$, and the original local search of [6]. The original local search was run with 10 random restarts, while greedy covering and beam search used 10 restarts in optimizing each seed added to the set.

Figure 1 compares the match probabilities of the seed sets with span up to 22 derived by each of the three search algorithms versus model \mathcal{M}_{nc} . Despite the fact that greedy covering does not simultaneously optimize multiple seeds, the seed sets it produced were in three of five cases ($n = 1, 3, 4$) at least as sensitive as those found by the original method. Differences in match probability were generally slight (within 0.01). Beam search, despite its more thorough exploration of the search space, produced seed sets with nearly the same probabilities as did greedy covering, giving us additional confidence that the latter does not unduly sacrifice the quality of its results.

While greedy covering yields seed sets of similar quality to those found by Mandala’s original local search, it has a clear advantage in computational cost. On a 2.8 GHz Intel Pentium 4 workstation, greedy covering with the above parameters produced a set of $n = 5$ seeds in about 20 minutes. The same computation with the original local search algorithm required 2.4 hours. The combined effects of fewer evaluations ($\frac{1}{2}$ to $\frac{2}{3}$ as many) for smaller n and more efficient computations per evaluation thus yielded a seven-fold speedup.

Given that the three methods tested in Figure 1 produced nearly identical results, the reader may wonder whether there is anything to optimize, that is, whether random seeds

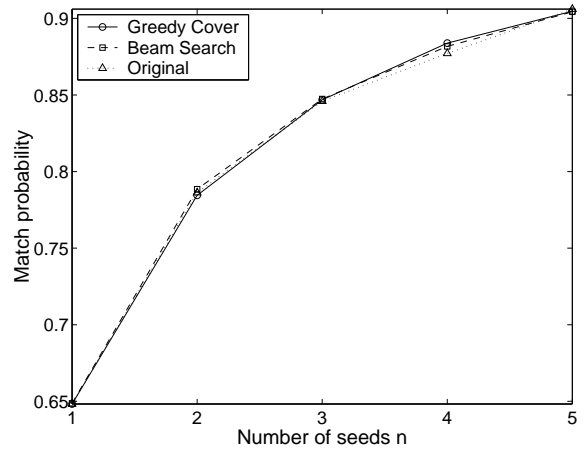


Figure 1: Match probabilities of seed sets derived by greedy covering, beam search ($b = 12$), and Mandala’s original local search in the first-order model \mathcal{M}_{nc} .

(as suggested in [16]) are just as sensitive as the best seeds found. While [6] refutes this possibility for $n = 1$, we sought to test it for $n > 1$. Figure 2 compares the match probability of locally optimal seed sets to the average probability over 100 random seed sets of the same weight and maximum span. There is a consistent gap of around 0.05 between the match probability of “average” seed sets and that of locally optimal sets with span up to 22, and an even larger gap of > 0.1 for maximum span 32. The experiments of [6] indicate that gaps of this magnitude significantly affect the sensitivity of a seed set in real-world sequence comparisons.

4.2 Mammalian Genomic Comparison

We evaluated whether the sensitivity gains estimated for multi-seed designs over single seeds translate into practical improvement by applying these designs to a large-scale comparison of the human and mouse genomes. In particular, we sought to determine whether the sensitivity conferred by multiple seeds of weight w was at least that obtained with single seeds of lower weight. Moreover, the qualitative argument of Section 2.1 suggests that sets of up to three well-designed seeds of weight w could exhibit greater sensitivity *and* specificity than one seed of weight $w - 1$. We sought to quantitatively validate this hypothesis.

The testing procedure was similar to that described in [6]. Briefly, we used 1262 pairs of orthologous human and mouse regions of total length 2.65 gigabases, masked to remove repeats and low-complexity DNA. The regions were divided into putative coding and noncoding portions using the predictions of the Twinscan gene finder [14]. Alignments sampled from each type of sequence were used to train a type-specific probabilistic alignment model \mathcal{M} , which in turn was used to pick optimized seed sets.

To test a set Π of seeds, we compared each pair of sequences with a BLAST-like seeded alignment algorithm using Π , followed by ungapped and gapped extension. Sensitivity of each seed set was evaluated by counting the total number of nonoverlapping gapped alignments found between orthologous regions, using the default scoring parameters and significance threshold of the PipMaker software [21].

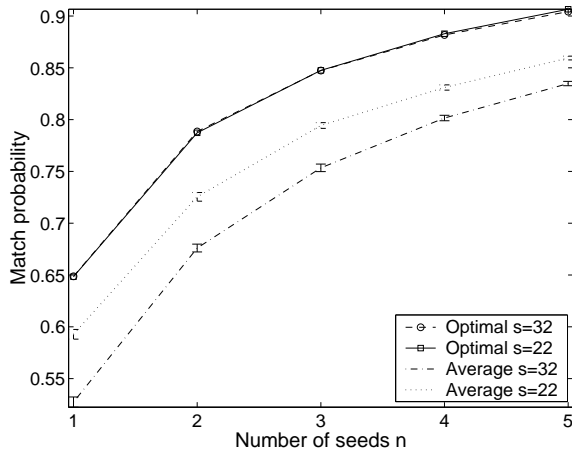


Figure 2: Comparison of seed sets produced by greedy covering with random sets of same weight $w = 11$ and maximum span s . “Optimal” curves for $n \leq 22$ and $n \leq 32$ nearly coincide. Error bars give 95% confidence intervals of mean over 100 trials.

Further details about the sequences used are given in Appendix A.

4.2.1 Noncoding DNA Sequence

A first-order noncoding model \mathcal{M}_{nc} was trained on approximately 1.4 million randomly sampled alignments with 70–75% identity from noncoding orthologous regions. We designed sets of up to five seeds with weight 11 and span at most 22, as well as locally optimal single seeds of weights 10 and 9. The multi-seed sets are given in Figure 3. In the interest of time, evaluation used only 449 pairs of regions spanning about 500 megabases of total unmasked sequence.

Table 1 shows the percent increases in sensitivity observed over a single optimized seed of weight 11, either by decreasing seed weight or by increasing the set size n . We found that a pair of seeds with weight 11 outperformed a single optimized seed of weight 10, and that four seeds of weight 11 outperformed an optimized seed of weight 9. Note that the percent increase in sensitivity observed between one weight-11 seed and three such seeds is of comparable magnitude to that obtained by switching from a contiguous 11-mer seed to the optimal single seed of the same weight.

These experiments provided an opportunity to investigate how quickly adding seeds to a set reaches the point of diminishing returns in sensitivity. While successive improvements in sensitivity do diminish with increasing n , they remain above 1% even as the fifth seed is added to the set. Provided that an indexing or hardware-accelerated search strategy can support $n > 2$, we find no compelling reason to stop at two or even three seeds in designs for these platforms.

Table 1 also gives, for each experiment, the total number of seed matches investigated by the algorithm, which is closely related to specificity because nearly all seed matches are false positives. The total seed match rates indeed exhibit a roughly linear increase with the number of seeds n . Moreover, as predicted, four seeds of weight 11 give roughly the same total number of matches as one seed of weight 10, though the total match rate increases somewhat less than four-fold as w is decreased by one. These results, along

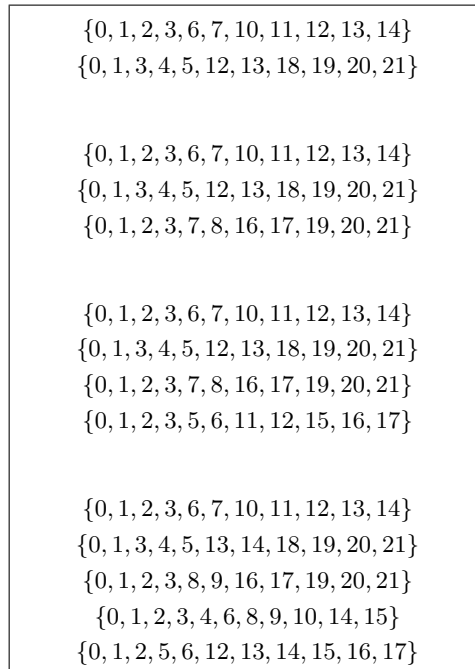


Figure 3: Seed sets of sizes $n = 2$ through 5 for noncoding DNA comparison. Each set of seeds was obtained in an independent run of optimization, so similarities between sets reflect convergence of different runs to the same local optimum.

with the substantial gains in sensitivity from multiple seeds, demonstrate that multi-seed designs can outperform single-seed designs in *both* sensitivity and specificity.

We note that the total seed match figures of Table 1 count matches between a given pair of locations i, j in query and database only once, even if multiple seeds matched there. The observed linear trend therefore shows that the different seeds of our optimized sets generate largely non-overlapping sets of false positives.

Finally, we repeated the experiments of Table 1, this time using seed sets of weight 11 and span up to 32, which were designed with our optimized Monte Carlo method. Designs with a longer maximum span can better avoid overlap between seeds if doing so improves the overall match probability. However, as Figure 2 suggests, the predicted improvement in sensitivity obtained from optimized seed sets of $s \leq 32$ is small compared to sets of $s \leq 22$. In practice, we found that the seed sets of longer span yielded a small but systematic improvement of 0.3–0.5% in the number of alignments found, compared to seed sets of the same size with maximum span 22.

4.2.2 Coding DNA Sequence

A fifth-order nonstationary Markov model \mathcal{M}_c was trained using approximately 357,000 alignments of 70–75% identity extracted from TBLASTX alignments of putative coding DNA in 1262 pairs of orthologous human and mouse sequences. The model \mathcal{M}_c consists of three stationary models predicting the rates of substitution at first, second, and third codon positions, respectively. Each submodel looks at the last five alignment positions in making its prediction. A nonstationary model more accurately predicts the relative

w	n	# alignments found	% improvement	total seed matches
11	1	251941	–	1.57×10^9
10	1	273831	8.7	5.88×10^9
9	1	293670	16.6	1.72×10^{10}
11	2	279902	11.1	3.10×10^9
11	3	292093	15.9	4.56×10^9
11	4	298968	18.7	6.05×10^9
11	5	303197	20.3	7.61×10^9

Table 1: Sensitivity and specificity of optimized seeds and seed sets of weight w and size n in model \mathcal{M}_{nc} . All seeds had span ≤ 22 . Sensitivity improvement is measured versus one optimized seed of weight 11.

sensitivity of seeds in coding DNA than does a stationary model; a more comprehensive treatment of this phenomenon appears in [3].

As before, we designed seed sets with weight 11 and span at most 22, as well as single optimized seeds of weights 10 and 9, to optimize sensitivity in model \mathcal{M}_c . The seed sets are given in Figure 4. Table 2 shows the sensitivity of these seed sets, again relative to a single optimized seed of weight 11, and the total number of seed matches observed in each experiment. While the overall magnitude of sensitivity improvement was lower than in noncoding DNA for both multi-seed designs and lower-weight seeds, the multi-seed designs again showed sensitivity at least comparable to that of single-seed designs producing considerably more false positives. The total numbers of seed matches continue to increase nearly linearly with the number of seeds, albeit with a slope of about 0.6 rather than 1.0, suggesting that alignments of real coding DNA incur substantially more matches by two or more seeds at the same location than those produced by the simple i.i.d. model of Section 2.1.

5. CONCLUSIONS

The design of multi-seed sets for nontraditional seeded alignment technologies presents a computational challenge, even compared to the already nontrivial problem of designing single seeds, because the space of possible designs increases exponentially with the set size. We have proposed new methods to navigate this larger design space either by dynamic programming or by Monte Carlo, obtaining significant improvements in computing time over the local search scheme of [6]. The resulting seed sets, all of moderate size, demonstrate markedly improved sensitivity on a large mammalian genomic DNA comparison, even relative to single seeds that inspect fewer alignment positions. Our multi-seed designs are good candidates for inclusion in indexing- and hardware-based similarity search tools.

The ability to efficiently design simultaneous seeds raises interesting possibilities for future improvements in seeded alignment. One direction for improvement is combining optimized seeds for different types of alignments in a single set. For example, seeds optimized to detect coding alignments have a structure quite different from those trained on alignments of less rigidly organized noncoding DNA. Combining both types of seed in a single set could improve sensitivity for both types of sequence. However, simply combining separately derived optimized seeds will likely leave considerable overlap between the similarities detected by each. A better approach, which would yield more orthogonal seeds, might be to design a single seed set using an appropriately weighted mixture of models for all sequence types of interest.

$\{0, 1, 3, 4, 9, 10, 12, 13, 15, 16, 18\}$
$\{0, 1, 3, 4, 6, 7, 9, 10, 18, 19, 21\}$
$\{0, 1, 3, 4, 9, 10, 12, 13, 15, 16, 18\}$
$\{0, 1, 3, 4, 6, 7, 9, 10, 18, 19, 21\}$
$\{0, 1, 2, 3, 8, 9, 14, 15, 18, 20, 21\}$
$\{0, 1, 3, 4, 9, 10, 12, 13, 15, 16, 18\}$
$\{0, 1, 3, 4, 6, 7, 9, 10, 18, 19, 21\}$
$\{0, 1, 2, 3, 8, 9, 14, 15, 18, 20, 21\}$
$\{0, 1, 4, 10, 11, 12, 13, 16, 19, 20, 21\}$
$\{0, 1, 3, 4, 9, 10, 12, 13, 15, 16, 18\}$
$\{0, 1, 3, 4, 6, 7, 9, 10, 18, 19, 21\}$
$\{0, 1, 2, 3, 8, 9, 14, 15, 18, 20, 21\}$
$\{0, 1, 4, 10, 11, 12, 13, 16, 19, 20, 21\}$
$\{0, 1, 3, 4, 6, 7, 12, 13, 14, 16, 17\}$

Figure 4: Seed sets of sizes $n = 2$ through 5 for coding DNA comparison. Each set of seeds was obtained in an independent run of optimization, so similarities between sets reflect convergence of different runs to the same local optimum.

A second application of multi-seed design is the exploitation of the score matrix embeddings described in [5]. In that work, we show that a score matrix M can be mapped to a finite metric δ that embeds isometrically in Hamming space, so that a random seed detects a match between the embedded representations of two sequences with probability proportional to their ungapped alignment score under M . While incorporating score matrices into seeded alignment appears to improve the sensitivity of DNA similarity search, our current implementation of this idea requires tens to hundreds of random seeds. The present work raises the possibility that a few carefully designed seeds for the embedded representations of sequences might achieve the same sensitivity at much reduced cost.

Finally, we plan to construct and evaluate alternative platforms for seeded alignment that can take advantage of multi-seed designs. An indexed search test bed could be built by, e.g., modification of the existing code base for BLAT [12], while hardware-based acceleration is a greater challenge that

w	n	# alignments found	% improvement	total seed matches
11	1	94109	–	1.28×10^6
10	1	95804	1.8	3.27×10^6
9	1	97255	3.3	9.93×10^6
11	2	95758	1.7	1.98×10^6
11	3	96789	2.8	2.66×10^6
11	4	97229	3.3	3.30×10^6
11	5	97521	3.6	3.94×10^6

Table 2: Sensitivity and specificity of optimized seeds and seed sets of weight w and size n in model \mathcal{M}_c . All seeds had span ≤ 22 . Sensitivity improvement is measured versus one optimized seed of weight 11.

must be addressed through collaboration with experts in FPGAs and network processors. By adapting seeded alignment to play to the particular strengths of these platforms, we will pursue quantum improvements in the efficiency of large-scale DNA sequence comparison.

An upcoming release of the Mandala software, incorporating our improvements for multi-seed design, will be made available at <http://www.cse.wustl.edu/~jbuhler/mandala/>.

Acknowledgments

This work was supported by NSF Career Grant DBI-0237903.

6. REFERENCES

- [1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, et al. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–10, 1990.
- [2] R. Bisiani. Search, beam. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 1467–1468. Wiley-Interscience, New York, NY, 2nd edition, 1992.
- [3] B. Brejova, D. G. Brown, and T. Vinar. Optimal spaced seeds for hidden Markov models, with application to homologous coding regions. In *Fourteenth Annual Symposium on Combinatorial Pattern Matching (CPM '03)*, number 2676 in Lecture Notes in Computer Science, pages 42–54, Berlin, Germany, 2003. Springer Verlag.
- [4] J. Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17:419–28, 2001.
- [5] J. Buhler. Provably sensitive indexing strategies for biosequence similarity search. *Journal of Computational Biology*, 10(3/4):399–418, 2003.
- [6] J. Buhler, U. Keich, and Y. Sun. Designing seeds for similarity search in genomic DNA. In *Proceedings of the Seventh Annual International Conference on Computational Molecular Biology (RECOMB '03)*, pages 67–75, Berlin, Germany, 2003.
- [7] A. Califano and I. Rigoutsos. FLASH: a fast look-up algorithm for string homology. In *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology (ISMB '93)*, pages 56–64, 1993.
- [8] F. Chiaromonte, V. B. Yap, and W. Miller. Scoring pairwise genomic sequence alignments. In *Pacific Symposium on Biocomputing*, pages 115–26, 2002.
- [9] K. Compton and S. Hauck. Reconfigurable computing: a survey of systems and software. *ACM Computing Surveys*, 34(2):171–210, 2002.
- [10] R. C. Hardison, J. Oeltjen, and W. Miller. Long human-mouse sequence alignments reveal novel regulatory elements: a reason to sequence the mouse genome. *Genome Research*, 7:959–66, 1997.
- [11] J. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In *Theory of Machines and Computations*, pages 189–96. Academic Press, 1971.
- [12] W. J. Kent. BLAT: the BLAST-like alignment tool. *Genome Research*, 12:656–64, 2002.
- [13] W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler, and D. Haussler. The human genome browser at UCSC. *Genome Research*, 12:996–1006, 2002.
- [14] I. Korf, P. Flicek, D. Duan, and M. R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17 Suppl 1:S140–8, 2001.
- [15] J. Li and W. Miller. Significance of inter-species matches when evolutionary rate varies. In *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB02)*, pages 216–24, Washington, DC, 2002.
- [16] B. Ma, J. Tromp, and M. Li. PatternHunter – faster and more sensitive homology search. *Bioinformatics*, 18:440–5, 2002.
- [17] National Center for Biological Information. Growth of GenBank, 2002. <http://www.ncbi.nlm.nih.gov/genbank/genbankstats.html>.
- [18] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence analysis. *Proceedings of the National Academy of Sciences USA*, 85:2444–8, 1988.
- [19] P. Pevzner and G. Tesler. Transforming men into mice: the Nadeau-Taylor chromosomal breakage model revisited. In *Proceedings of the Seventh Annual International Conference on Computational Molecular Biology (RECOMB '03)*, pages 247–56, Berlin, Germany, 2003.
- [20] S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, D. Haussler, and W. Miller. Human-mouse alignments with BLASTZ. *Genome Research*, 13:103–7, 2003.
- [21] S. Schwartz, Z. Zhang, K. A. Frazer, A. F. Smit, et al. PipMaker – a web server for aligning two genomic DNA sequences. *Genome Research*, 10:577–86, 2000.
- [22] N. Shah and K. Keutzer. Network processors: origin of species. In *Proceedings of ISICIS XVII, the Seventeenth International Symposium on Computer and Information Sciences*, October 2002.
- [23] A. F. Smit and P. Green. RepeatMasker, 1999. <http://ftp.genome.washington.edu/RM/RepeatMasker.html>.

APPENDIX

A. DETAILS OF EXPERIMENTS

We obtained NCBI Build 31 of the human genome and Release 2 of the mouse genome, along with their annotations, from the UCSC Genome Browser [13]. Sequences were divided into annotated coding exons and the remaining noncoding DNA based on Twinscan’s coding exon predictions. Soft-masked regions of the sequences, representing interspersed repeats and low-complexity DNA, were ignored for training and testing purposes.

We mined pairs of orthologous human and mouse regions for the alignments needed to train the Markov models \mathcal{M}_{nc} and \mathcal{M}_c . The noncoding training set was produced by randomly sampling ungapped alignments of length 64 with 70–75% identity from pairs of orthologous sequences, as described in [6].

To generate the coding training set, TBLASTX was run on the coding portions of the orthologous region pairs with the BLOSUM62 scoring function, modified so as to heavily penalize stop codons. Translated alignments with E-values at most 10^{-5} were mapped back to their underlying genomic sequences, and aligned segments with 70–75% identity were extracted for training.

Test comparisons were performed with a modified version of `1sh`, a seeded alignment search tool developed by Buhler [4]. The program was modified to use hashing rather than sorting for detecting seed matches and to read a list of seeds from a file. Each seed match was subjected to ungapped extension using NCBI BLAST’s linear-time dynamic programming algorithm, followed by gapped extension using banded Smith-Waterman. We scored alignments with the HOXD-70 score matrix [8] and affine gap penalties of -400 to open and -30 to extend. We kept only non-overlapping alignments that scored above 3000, the default cutoff used by PipMaker.