JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 12, Number 6, 2005

Pp. 847–861

Designing Multiple Simultaneous Seeds for DNA Similarity Search

YANNI SUN and JEREMY BUHLER

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 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.

Key words: DNA sequence comparison, sequence alignment, database search, seed design, Mandala.

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 nongene features such as transposable elements (Smit and Green, 1999) and putative regulatory regions (Hardison *et al.*, 1997; Li and Miller, 2002). Pairwise comparison of entire mammalian genomes has been used to support gene-structure prediction (Korf *et al.*, 2001), long-range orthology detection (Schwartz *et al.*, 2003), and inference of genome rearrangements (Pevzner and Tesler, 2003).

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 (National Center for Biological Information, 2002), have revived innovation in heuristics for fast similarity search in DNA (Schwartz *et al.*, 2000; Kent, 2002; Ma *et al.*, 2002; Buhler *et al.*, 2003). At the same time,

Department of Computer Science and Engineering, Washington University, St. Louis, MO 63130.

interest in whole-genome alignment (Bray *et al.*, 2003; Brudno *et al.*, 2003; Lipper *et al.*, 2004) has led to tools that perform local similarity search as a way to "anchor" global alignments. All of these search methods, like their ancestors BLASTN (Altschul *et al.*, 1990) and FASTN (Pearson and Lipman, 1988), 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 $\{x_1 \ldots x_w\}$. 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 $\pi_c = \{0, 1, \ldots w - 1\}$, but more recent work (Ma *et al.*, 2002; Buhler *et al.*, 2003; Brejova *et al.*, 2004) has shown the desirability of utilizing seeds which inspect a discontiguous 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 (Califano and Rigoutsos, 1993) and BLAT (Kent, 2002) 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 (Kent, 2002) 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) (Compton and Hauck, 2002) or network processor (Shah and Keutzer, 2002). 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 $\Pi = \{\pi_1 \ldots \pi_n\}$ 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 (Ma *et al.*, 2002) and our own work on the Mandala seed design software (Buhler *et al.*, 2003) have shown the utility of designing a seed to optimize sensitivity in a probabilistic model $\mathcal{M}$ of the alignments being sought. However, these earlier methods, while applicable to multi-seed design, scale poorly with the size n of the seed set. We have therefore sought a robust, efficient approach to multi-seed design that leverages the power of existing local-search methods (Buhler *et al.*, 2003) 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 Buhler *et al.* (2003). 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. After reviewing related work, 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.

1.1. Related work

Optimized design of multiple simultaneous seeds builds on previous work in database search generally and biosequence similarity search in particular. One closely related thread of work is the use of *randomized* seed sets, that is, sets of seeds chosen independently at random from some distribution. Califano and Rigoutsos (1993), in their FLASH similarity search tool, chose up to tens of seeds that inspect sequence positions chosen independently at random, creating a separate index of the database on disk for each seed. More recently, random seed sets have been applied to geometric nearest neighbor search (Indyk and Motwani, 1998) and to general database search (Gionis *et al.*, 1999) under the term *locality-sensitive hashing*. This approach seeks objects in a database that are close to a query object under some metric, such as ℓ_k distance for real-valued vectors or Hausdorff distance for geometric shapes. They first construct a low-distortion embedding mapping objects to bit strings under the Hamming metric, then use enough randomly chosen seeds to discover with high probability all pairs of vectors that match in sufficiently many positions.

Buhler (2001) adapted ideas from locality-sensitive hashing to biosequence comparison in the LSH-ALL-PAIRS software. Like FLASH, this tool uses multiple seeds; unlike FLASH, it chooses the number of seeds to guarantee a high probability of discovering sufficiently conserved ungapped alignments (i.e., pairs of strings at small Hamming distance) *regardless* of how their substitutions are distributed. LSH-ALL-PAIRS is computationally expensive because it requires a large number of seeds to bound the probability of missing a well-conserved alignment. In contrast, the present work seeks a much smaller number of seeds designed to optimize performance given a probabilistic model that describes the alignments being sought.

The use of an alignment model to design seeds appears in FLASH's performance analysis, which assumes that alignments contain uniformly distributed mutations. Ma *et al.* (2002) recognized that this assumption strongly impacts the choice of even a single seed and exploited it to pick the seed used by their PatternHunter search tool. Keich *et al.* (2004) gave a dynamic programming algorithm to estimate the probability that a seed discovers an alignment with mutations distributed uniformly at random; an equivalent algorithm using finite automata is described by Nicodéme *et al.* (1999). Buhler, Keich, and Sun gave a faster algorithm that extends easily to more complex alignment models in Buhler *et al.* (2003), where it was used in the Mandala seed design software. These methods extend in principle to multiple seed design, but attempting to optimize a set Π of seeds simultaneously remains computationally challenging.

Recently, Xu *et al.* (2004) found an LP-rounding approximation algorithm for Problem 1.1. The approximation ratio depends in a nontrivial way on the properties of the seeds being designed, but in practice seed sets can be designed whose probability of detecting a match is at least 70% of the highest possible for any seed set of the same size. The heuristic methods described in this work do not provide such guarantees, but the seed sets we produce empirically exhibit sensitivity comparable to that reported by Xu *et al.*

Other users of seed sets in similarity search include Pevzner and Waterman (1995) and Kucherov *et al.* (2004). Unlike the probabilistic approach described here, their methods use carefully designed seed sets to guarantee perfect sensitivity to alignments with a certain level of conservation while minimizing the number of false positives. Also, Pevzner and Waterman's pairs of seeds are used *conjunctively* (that is, all seeds must match to discover an alignment), while this work uses seeds *disjunctively* (at least one seed must match).

Alternative extensions of single seeds that do not explicitly use seed sets include the BLASTZ algorithm (Schwartz *et al.*, 2003), which combines seed matching with differentiation between transition and transversion mutations in DNA, and vector seeds (Brejova *et al.*, 2003), which sample alignments like ordinary seeds but then add together the scores of the sampled alignment positions under some substitution matrix.

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 Buhler *et al.* (2003) 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. Previously (Buhler *et al.*, 2003), we reported 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) kth-order Markov process, can be trained on biologically meaningful alignments obtained from real genomic sequence comparisons.

We say that a seed $\pi = \{x_1 \ldots x_w\}$ *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

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

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 nonfixed positions $x_2 \ldots x_w$ 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. A greedy covering algorithm for multi-seed design

Finding an optimal set of seeds for a model $\mathcal{M}$ is computationally challenging because seed design space contains numerous local maxima of sensitivity that may be substantially below the global maximum. The algorithm of Buhler *et al.* (2003) addresses this challenge using a hill-climbing local search algorithm with random restart. 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 ith 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. This entire procedure is then repeated for some number of different starting points chosen at random. To design a single optimized seed, we simply run the above algorithm with $n = 1$.

Empirically, two factors increase the running time of local search precipitously with the set size n. Firstly, when evaluating match probabilities using the fast dynamic programming algorithm of Buhler *et al.* (2003), 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 empirically 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* $\Pr(E_\pi \mid \overline{E_{\Pi_0}})$. 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 the local search of Buhler *et al.* (2003) 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 that of 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 computation-intensive strategy uses a beam search (Bisiani, 1992) 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 $\Pr(E_\pi \mid \overline{E_{\pi_0}})$. 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 $\Pr(E_\pi \mid \overline{E_\Pi})$ for an alignment model $\mathcal{M}$. In this section, we describe enhancements to the methods of Buhler *et al.* (2003) to efficiently compute conditional match probabilities. A single evaluation of $\Pr(E_\pi \mid \overline{E_\Pi})$ 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 than their weight ($s - w \leq 15$), a highly efficient method (Buhler *et al.*, 2003) to compute $\Pr(E_\pi)$ exactly 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 kth-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 sub-quadratic-time method such as Hopcroft's algorithm (Hopcroft, 1971).

We can compute $\Pr(E_\pi \mid \overline{E_\Pi})$ for a seed π and a set of seeds Π as

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

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 $w 2^{s-w}$. 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 1,000-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), $\Pr(E_\pi \mid \overline{E_\Pi})$) 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 of Buhler *et al.* [2003]) 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 kth-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

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

where event q_t (resp., r_{t+1}^b) denotes 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

$$\Pr(r_{t+1}^b \mid q_t, h_t, \overline{f_\ell}) = \frac{\Pr(\overline{f_\ell} \mid r_{t+1}^b, q_t, h_t)\Pr(r_{t+1}^b \mid q_t, h_t)}{\Pr(\overline{f_\ell} \mid q_t, h_t)}. \tag{3}$$

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 $\Pr(b \mid h_t)$, 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

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

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

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

It remains to compute $\Pr(f_\ell \mid q_t, h_t)$ for any state q of A_Π and k-bit history h and any $t \le \ell$. 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:

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

The base cases of the recurrence are that $\Pr(f_\ell \mid f_\ell, h_\ell) = 1$, while $\Pr(f_\ell \mid q_\ell, h_\ell) = 0$ 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 $p(q, h, t) = \Pr(r_{t+1}^1 \mid q_t, h_t, \overline{f_\ell})$ for each state q, each k-bit history h, and each length $t \le \ell$. 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 Buhler *et al.* (2003) 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 Buhler *et al.* (2003). 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. Sequence data and alignment models

The experiments of this section were carried out on pairs of orthologous DNA sequence blocks drawn from the human and mouse genomes. We obtained NCBI Build 31 of the human genome and Release 2 of the mouse genome, along with the annotated coordinates of 1,262 pairs of orthologous blocks, from the UCSC Genome Browser (Kent *et al.*, 2002). These blocks comprised 2.65 gigabases of total DNA sequence. Sequences were divided into annotated coding exons and the remaining noncoding DNA based on the coding exon predictions of the Twinscan gene finder (Korf *et al.*, 2001). Soft-masked regions of the sequences, representing interspersed repeats and low-complexity DNA, were not used either in training or in testing.

We mined orthologous pairs of regions for high-scoring local sequence alignments, which were used to train probabilistic models for seed design. The noncoding training set was produced by randomly sampling ungapped alignments of length 64 with 70–75% identity from these pairs of regions, as described by Buhler *et al.* (2003). 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. These procedures generated 1.4 million noncoding and 357,000 coding alignments, respectively.

Using our sampled alignments, we trained two probabilistic models—a noncoding model $\mathcal{M}_{nc}$ and a coding model $\mathcal{M}_c$—for use in seed design. Model $\mathcal{M}_{nc}$ is a simple first-order Markov process. The coding model $\mathcal{M}_c$ is a fifth-order nonstationary Markov model consisting of three stationary models, which predict the rates of substitution at first, second, and third codon positions, respectively. Each submodel looks at the previous five alignment positions in making its prediction. A nonstationary model more accurately predicts the relative sensitivity of seeds in coding DNA than does a stationary model because it can capture the three-periodic structure of codons: the second base of a codon is the best conserved, followed closely by the first base, while the third or "wobble" base typically evolves nearly neutrally. A more comprehensive treatment of this phenomenon appears in Brejova *et al.* (2004).

4.2. Performance of greedy covering

We first evaluated the performance of the greedy covering algorithm, along with the more aggressive beam search strategy, relative to the original, more expensive local search of Buhler *et al.* (2003). These experiments used the noncoding alignment model $\mathcal{M}_{nc}$.

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 Buhler *et al.* (2003). 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.

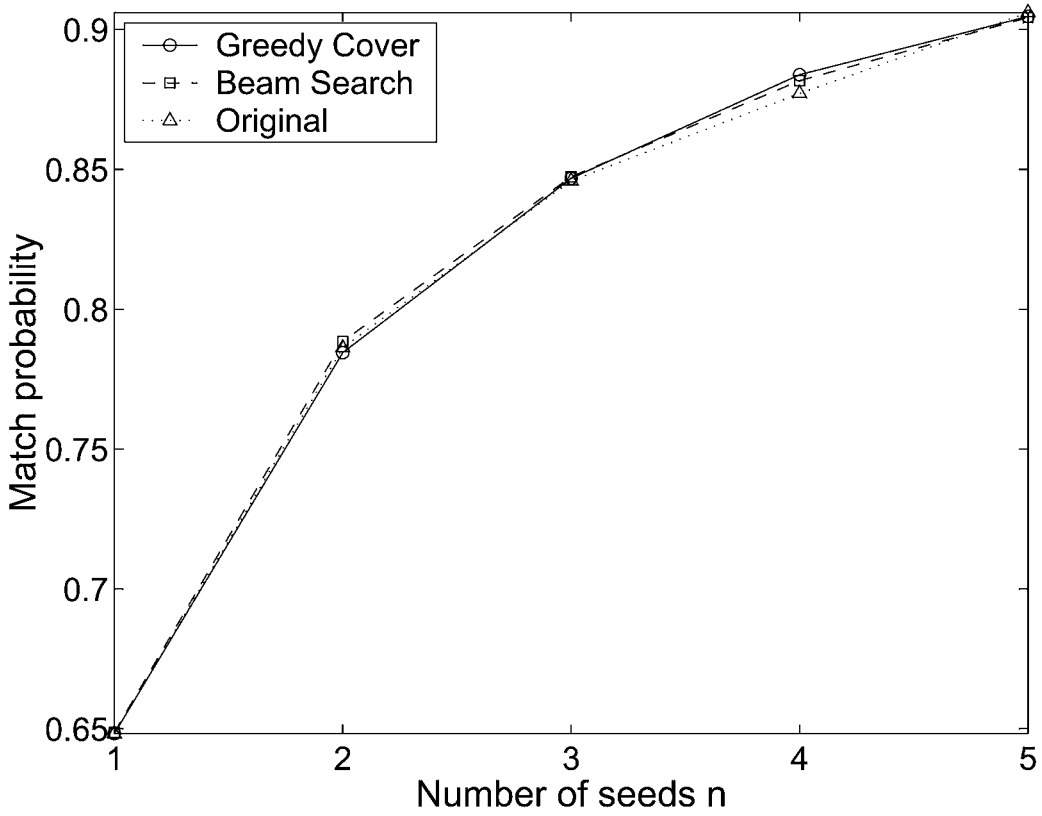

FIG. 1. Match probabilities of seed sets derived by greedy covering, beam search ($b = 12$), and Mandala's original local search in the first-order model $\mathcal{M}_{nc}$.

While greedy covering yields seed sets of quality similar to those found by the original local search of Buhler *et al.* (2003), 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 yielded a seven-fold speedup.

Given that the three methods in Fig. 1 produced nearly identical results, the reader may wonder whether there is anything to optimize, that is, whether random seeds (as suggested by Ma *et al.* [2002]) are just as sensitive as the best seeds found. 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 Buhler *et al.* (2003) indicate that gaps of this magnitude significantly affect the sensitivity of a seed set in real-world sequence comparisons.

4.3. Mammalian genomic comparison

We next 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.

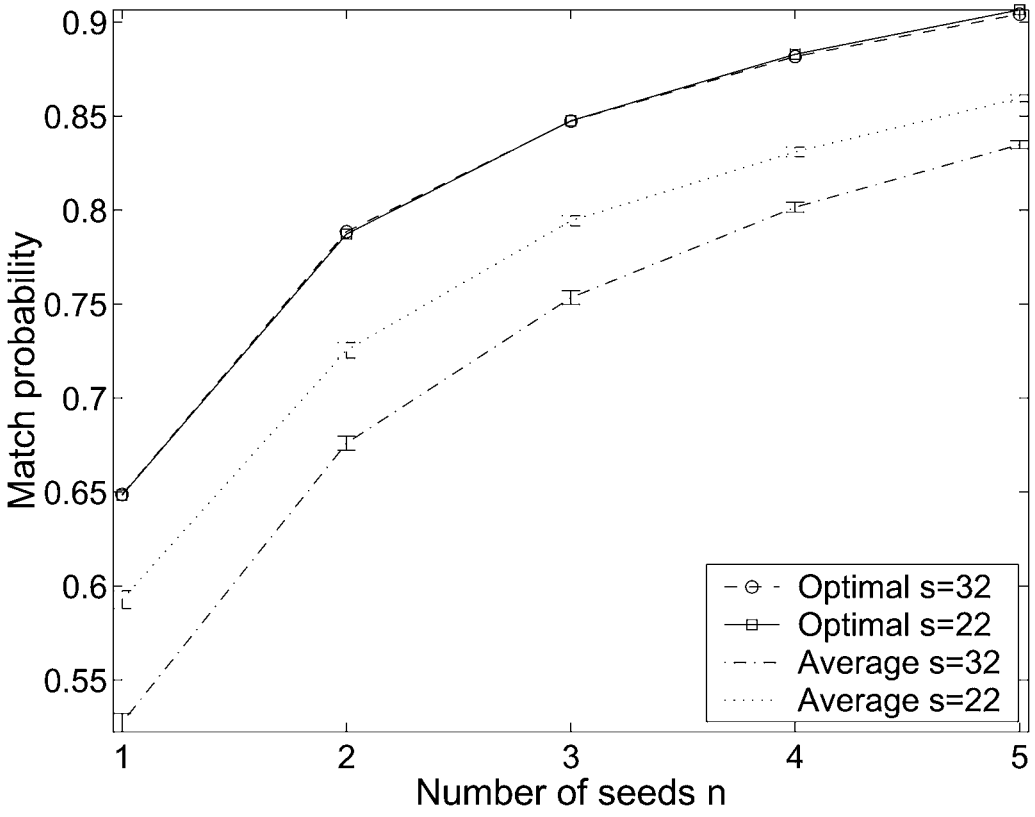

FIG. 2. Comparison of seed sets produced by greedy covering with random sets of same weight $w = 11$ and maximum span s. "Optimal" curves for $n \leq 22$ and $n \leq 32$ nearly coincide. Error bars give 95% confidence intervals of mean over 100 trials.

Our testing procedure used the masked pairs of orthologous human and mouse regions described above. To test a set Π of seeds, we compared each orthologous pair of sequences with a BLAST-like seeded alignment algorithm using Π. 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 (Chiaromonte *et al.*, 2002) and affine gap penalties of -400 to open and -30 to extend. We kept only non-overlapping alignments that scored above 3,000. These scoring parameters were taken from the popular PipMaker comparison software (Schwartz *et al.*, 2000). Finally, the sensitivity of each seed set Π was quantified by counting the total number of nonoverlapping gapped alignments found between orthologous regions.

4.3.1. Noncoding DNA sequence. Our first experiments on noncoding DNA sequence used seed sets designed with the model $\mathcal{M}_{nc}$. 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 Fig. 3. In the interest of time, evaluation used only 449 pairs of orthologous 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

$\{0, 1, 2, 3, 6, 7, 10, 11, 12, 13, 14\}$

$\{0, 1, 3, 4, 5, 12, 13, 18, 19, 20, 21\}$

$\{0, 1, 2, 3, 6, 7, 10, 11, 12, 13, 14\}$

$\{0, 1, 3, 4, 5, 12, 13, 18, 19, 20, 21\}$

$\{0, 1, 2, 3, 7, 8, 16, 17, 19, 20, 21\}$

$\{0, 1, 2, 3, 6, 7, 10, 11, 12, 13, 14\}$

$\{0, 1, 3, 4, 5, 12, 13, 18, 19, 20, 21\}$

$\{0, 1, 2, 3, 7, 8, 16, 17, 19, 20, 21\}$

$\{0, 1, 2, 3, 5, 6, 11, 12, 15, 16, 17\}$

$\{0, 1, 2, 3, 6, 7, 10, 11, 12, 13, 14\}$

$\{0, 1, 3, 4, 5, 13, 14, 18, 19, 20, 21\}$

$\{0, 1, 2, 3, 8, 9, 16, 17, 19, 20, 21\}$

$\{0, 1, 2, 3, 4, 6, 8, 9, 10, 14, 15\}$

$\{0, 1, 2, 5, 6, 12, 13, 14, 15, 16, 17\}$

FIG. 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.

TABLE 1. SENSITIVITY AND SPECIFICITY OF SEEDS IN MODEL $\mathcal{M}_{nc}$[a]

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

[a]All seeds and seed sets shown are optimized, with span ≤ 22. Sensitivity improvement is measured versus one optimized seed of weight 11; *w*: seed weight; *n*: number of simultaneous seeds.

$$\{0, 1, 3, 4, 9, 10, 12, 13, 15, 16, 18\}$$

$$\{0, 1, 3, 4, 6, 7, 9, 10, 18, 19, 21\}$$

$$\{0, 1, 3, 4, 9, 10, 12, 13, 15, 16, 18\}$$

$$\{0, 1, 3, 4, 6, 7, 9, 10, 18, 19, 21\}$$

$$\{0, 1, 2, 3, 8, 9, 14, 15, 18, 20, 21\}$$

$$\{0, 1, 3, 4, 9, 10, 12, 13, 15, 16, 18\}$$

$$\{0, 1, 3, 4, 6, 7, 9, 10, 18, 19, 21\}$$

$$\{0, 1, 2, 3, 8, 9, 14, 15, 18, 20, 21\}$$

$$\{0, 1, 4, 10, 11, 12, 13, 16, 19, 20, 21\}$$

$$\{0, 1, 3, 4, 9, 10, 12, 13, 15, 16, 18\}$$

$$\{0, 1, 3, 4, 6, 7, 9, 10, 18, 19, 21\}$$

$$\{0, 1, 2, 3, 8, 9, 14, 15, 18, 20, 21\}$$

$$\{0, 1, 4, 10, 11, 12, 13, 16, 19, 20, 21\}$$

$$\{0, 1, 3, 4, 6, 7, 12, 13, 14, 16, 17\}$$

FIG. 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.

TABLE 2. SENSITIVITY AND SPECIFICITY OF SEEDS IN MODEL $\mathcal{M}_c$[a]

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

[a]All seeds and seed sets shown are optimized, with span ≤ 22. Sensitivity improvement is measured versus one optimized seed of weight 11; w: seed weight; n: number of simultaneous seeds.

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 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 Fig. 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.

4.3.2. Coding DNA sequence. For our trials with coding DNA, 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 the coding model $\mathcal{M}_c$. The seed sets are given in Fig. 4. Each seed set was tested on the coding portions of all 1262 orthologous region pairs.

Table 2 shows the sensitivity of our coding 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 AND FUTURE WORK

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 Buhler *et al.* (2003). 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 methods described in this work are part of release 1.1 of the Mandala software, available at *http://www.cse.wustl.edu/∼jbuhler/mandala/*.

We plan to construct and evaluate alternative platforms for seeded alignment that can take advantage of multi-seed designs. The Mercury system (Chamberlain *et al.*, 2003) is a hardware platform designed to stream large quantities of data from an array of disks through a controller containing an embedded FPGA processor. This system can sustain a data rate of 800 Mbytes/second into the FPGA. We are constructing an implementation of DNA similarity search for this system (Krishnamurthy *et al.*, 2004) that performs seeded alignment on the FPGA. The inherent parallelism of the FPGA makes feasible the use of several simultaneous seeds without increasing the latency of the computation.

The ability to efficiently design simultaneous seeds raises interesting possibilities for future improvements in seeded alignment. One application of multi-seed design is the exploitation of the score matrix embeddings described by Buhler (2003). In that work, it is shown 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. Because embedding-based seeded alignment (unlike vector seeds) would preserve the ability to use ordinary hashing to discover seed matches between query and database, this approach should be compatible with BLAST and similar alignment tools.

ACKNOWLEDGMENTS

This work was supported by NSF career grant DBI-0237903.

REFERENCES

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., *et al.* 1990. Basic local alignment search tool. *J. Mol. Biol.* 215, 403–410.

Bisiani, R. 1992. Search, beam. *In* Shapiro, S.C., ed., *Encyclopedia of Artificial Intelligence*, 2nd ed., 1467–1468, Wiley-Interscience, New York.

Bray, N., Dubchak, I., and Pachter, L. 2003. AVID: A global alignment program. *Genome Res.* 13, 97–102.

Brejova, B., Brown, D.G., and Vinar, T. 2003. Vector seeds: An extension to spaced seeds allows substantial improvements in sensitivity and specificity. *In* Benson, G., and Page, R., eds., *Algorithms and Bioinformatics: 3rd International Workshop (WABI)*, vol. 2812 of Lecture Notes in Bioinformatics, 39–54.

Brejova, B., Brown, D.G., and Vinar, T. 2004. Optimal spaced seeds for homologous coding regions. *J. Bioinformatics and Comp. Biol.* 1, 595–610.

Brudno, M., Do, C.B., Cooper, G.M., Kim, M.F., Davydov, E., Green, E.D., Sidow, A., and Batzoglou, S. 2003. LAGAN and Multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.* 13, 721–731.

Buhler, J. 2001. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics* 17, 419–428.

Buhler, J. 2003. Provably sensitive indexing strategies for biosequence similarity search. *J. Comp. Biol.* 10, 399–418.

Buhler, J., Keich, U., and Sun, Y. 2003. Designing seeds for similarity search in genomic DNA. *Proc. 7th Ann. Int. Conf. on Computational Molecular Biology (RECOMB '03)*, 67–75.

Califano, A., and Rigoutsos, I. 1993. FLASH: A fast look-up algorithm for string homology. *Proc. 1st Int. Conf. on Intelligent Systems for Molecular Biology (ISMB '93)*, 56–64.

Chamberlain, R.D., Cytron, R.K., Franklin, M.A., and Indeck, R.S. 2003. The Mercury system: Exploiting truly fast hardware for data search. *Proc. Intl. Workshop on Storage Network Architecture and Parallel I/Os*, 65–72.

Chiaromonte, F., Yap, V.B., and Miller, W. 2002. Scoring pairwise genomic sequence alignments. *Pac. Symp. Biocomputing*, 115–126.

Compton, K., and Hauck, S. 2002. Reconfigurable computing: A survey of systems and software. *ACM Computing Surveys* 34, 171–210.

Gionis, A., Indyk, P., and Motwani, R. 1999. Similarity search in high dimensions via hashing. *Proc. 25th Int. Conf. Very Large Databases.*

Hardison, R.C., Oeltjen, J., and Miller, W. 1997. Long human–mouse sequence alignments reveal novel regulatory elements: A reason to sequence the mouse genome. *Genome Res.* 7, 959–966.

Hopcroft, J. 1971. An $n \log n$ algorithm for minimizing states in a finite automaton. In *Theory of Machines and Computations*, 189–196, Academic Press, New York.

Indyk, P., and Motwani, R. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. *Proc. 30th Ann. ACM Symp. on Theory of Computing*, 604–613.

Keich, U., Li, M., Ma, B., and Tromp, J. 2004. On spaced seeds for similarity search. *Disc. Appl. Math.* 138, 253–263.

Kent, W.J. 2002. BLAT: The BLAST-like alignment tool. *Genome Res.* 12, 656–664.

Kent, W.J., Sugnet, C.W., Furey, T.S., Roskin, K.M., Pringle, T.H., Zahler, A.M., and Haussler, D. 2002. The human genome browser at UCSC. *Genome Res.* 12, 996–1006.

Korf, I., Flicek, P., Duan, D., and Brent, M.R. 2001. Integrating genomic homology into gene structure prediction. *Bioinformatics* 17(Suppl 1), S140–S148.

Krishnamurthy, P., Buhler, J., Chamberlain, R., Franklin, M., Gyang, K., and Lancaster, J. 2004. Biosequence similarity search on the Mercury system. *Proc. IEEE 15th Int. Conf. on Application-Specific Systems, Architectures, and Processors (ASAP '04)*, 365–375.

Kucherov, G., Noe, L., and Roytberg, M. 2004. Multi-seed lossless filtration. *Proc. 15th Ann. Combinatorial Pattern Matching Symposium (CPM '04)*, 297–310.

Li, J., and Miller, W. 2002. Significance of inter-species matches when evolutionary rate varies. *Proc. 6th Ann. Int. Conf. on Computational Molecular Biology (RECOMB '02)*, 216–224.

Lipper, R.A., Zhao, X., Florea, L., Mobarry, C., and Istrail, S. 2004. Finding anchors for genomic sequence comparison. *Proc. 8th Ann. Int. Conf. on Computational Molecular Biology (RECOMB '04)*, 233–241.

Ma, B., Tromp, J., and Li, M. 2002. PatternHunter—Faster and more sensitive homology search. *Bioinformatics* 18, 440–445.

National Center for Biological Information. 2002. Growth of GenBank. *http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html.*

Nicodéme, P., Salvy, B., and Flajolet, P. 1999. *Motif Statistics*, 194–211, Number 1643 in Lecture Notes in Computer Science, Springer, Berlin.

Pearson, W.R., and Lipman, D.J. 1988. Improved tools for biological sequence analysis. *Proc. Natl. Acad. Sci. USA* 85, 2444–2448.

Pevzner, P., and Tesler, G. 2003. Transforming men into mice: The Nadeau-Taylor chromosomal breakage model revisited. *Proc. 7th Ann. Int. Conf. on Computational Molecular Biology (RECOMB '03)*, 247–256.

Pevzner, P., and Waterman, M.S. 1995. Multiple filtration and approximate pattern matching. *Algorithmica* 13, 135–154.

Schwartz, S., Kent, W.J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R.C., Haussler, D., and Miller, W. 2003. Human–mouse alignments with BLASTZ. *Genome Res.* 13, 103–107.

Schwartz, S., Zhang, Z., Frazer, K.A., Smit, A.F., *et al.* 2000. PipMaker—A web server for aligning two genomic DNA sequences. *Genome Res.* 10, 577–586.

Shah, N., and Keutzer, K. 2002. Network processors: Origin of species. *Proc. ISCIS XVII, 17th Int. Symp. on Computer and Information Sciences*.

Smit, A.F., and Green, P. 1999. RepeatMasker, *http://www.ftp.genome.washington.edu/RM/RepeatMasker.html*.

Xu, J., Brown, D., Li, M., and Ma, B. 2004. Optimizing multiple spaced seeds for homology search. *Proc. 15th Ann. Combinatorial Pattern Matching Symposium (CPM '04)*, 47–58.

Address correspondence to:
Yanni Sun
Dept. of Computer Science and Engineering
Washington University
St. Louis, MO 63130

E-mail: yanni@cse.wustl.edu