Designing Seeds for Similarity Search in Genomic DNA

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

Uri Keich
keich@cs.ucsd.edu
Department of Computer Science and Engineering
University of California
San Diego
La Jolla, CA 92093

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

ABSTRACT

Large-scale comparison of genomic DNA is of fundamental importance in annotating functional elements of genomes. To perform large comparisons efficiently, BLAST [3, 2] and other widely used tools use seeded alignment, which compares only sequences that can be shown to share a common pattern or "seed" of matching bases. The literature suggests that the choice of seed substantially affects the sensitivity of seeded alignment, but designing and evaluating seeds is computationally challenging.

This work addresses problems arising in seed design. We give the fastest known algorithm for evaluating the sensitivity of a seed in a Markov model of ungapped alignments, as well as theoretical results on which seeds are good choices. We also describe **Mandala**, a software tool for seed design, and show that it can be used to improve the sensitivity of alignment in practice.

Categories and Subject Descriptors

J.3 [**Life and Medical Sciences**]: biology and genetics; 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, Theory

Keywords

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

1. INTRODUCTION

Genomes and genomic sequence databases provide a fundamental reference tool for molecular biologists. Searching these databases for DNA sequences similar to (differing by few mutations from) a query sequence, or for pairs of sequences similar to each other, remains important to detect repetitive elements [22] and noncoding parts of genes, to augment the power of gene-structure prediction [14], to compare whole genomes [9], and to identify sequences of unknown origin or function. Public genomic DNA sequence databases such as GenBank are growing exponentially [18], driving demand for fast comparison algorithms and heuristics that nonetheless are as sensitive as possible to biologically meaningful sequence conservation.

Seeded alignment is the dominant paradigm for accelerating large-scale genomic sequence comparison. BLAST [3, 2] and other widely used tools [21, 13] apply alignment algorithms like Smith-Waterman [23] only to pairs of sequences that exhibit prior evidence of similarity in the form of a shared *seed*, typically a common short substring or *word* of matching bases[1]. All matching words between two sequences can be found quickly, so seeded alignment efficiently directs computational resources toward pairs of sequence regions most likely to exhibit high similarity. The words in a sequence database can also be statically indexed [7, 13] to accelerate subsequent searches for word matches.

While words are the most popular type of seed for seeded alignment, discontiguous patterns of matching bases have seen considerable use in the sequence comparison literature. A discontiguous pattern spanning s bases, unlike a word of length s, requires matching pairs of bases at only a subset of the positions $\{0, 1, \dots s-1\}$. Califano and Rigoutsos, in their FLASH comparison tool [7], found that randomly chosen discontiguous patterns in practice yielded the highest sensitivity to pairs of similar sequences when used to index a database. Buhler [4] formally established the sensitivity of random patterns in developing the randomized LSH-ALL-PAIRS comparison algorithm. Discontiguous patterns have also been used to accelerate seeded alignment algorithms including that of Pevzner and Waterman [20] and, more recently, the BLASTZ algorithm underlying the PipMaker program [21] and Ma, Tromp, and Li's PatternHunter [17].

PatternHunter introduced an important formal innovation to seeded alignment: the *resource-constrained paradigm* of seed design. This paradigm fixes the computational cost of seeded alignment *a priori* by fixing the number of different seeds to be used and the approximate false positive rate

[1]BLASTN, unlike BLASTP, does *not* compute a neighborhood of each word in the query because typical word lengths are much longer for DNA than for protein.

for each seed. It then asks how to choose seeds that maximize the probability of detecting biosequence similarities described by a probabilistic model.

The resource-constrained paradigm of seed design is well-suited to BLAST-like tools, in which the cost of using more than one or two seeds to search a database is unacceptably high, as well as to static indexing schemes in which the number of indices, and hence of seeds, can be larger but is constrained by storage and disk access costs. However, actually designing seeds in the resource-constrained paradigm, even for simple probabilistic models of biosequence similarities, is computationally challenging. Moreover, previous work on resource-constrained seed design neither exploits more informative models of biosequence similarities nor extends to designing two or more seeds to be used simultaneously.

This work describes new tools for resource-constrained seed design. We address the following generalized design problem:

> *Given a collection of ungapped genomic sequence similarities of fixed length ℓ, whose distribution of matching base pairs is described by a kth-order Markov model $\mathcal{M}$, and resource limits w and n, find n seeds $\pi_1 \ldots \pi_n$, each inspecting w bases, such that the* ***sensitivity****, or probability that at least one seed detects a random similarity from $\mathcal{M}$, is maximized.*

The target application, detection of ungapped alignments between ℓ-mers with few substitutions, abstracts the initial filtering phase of seeded alignment. The cost of filtering is controlled by fixing the *weight* of, or number of positions inspected by, each seed (which largely controls its false positive rate, the chance of seeing a seed match in the absence of an ℓ-mer similarity) and by fixing the total number of seeds permitted. The Markov model $\mathcal{M}$ and the length ℓ describe a model of "interesting" ungapped alignments, which may be adjusted to match similarities between sequences of a particular type (e.g. protein-coding).

In the following sections, we present theoretical and practical results on solving resource-constrained seed design and show that the generalized design problem is of practical interest in improving algorithms for biosequence similarity search. Section 2 gives an exact algorithm to compute detection probabilities for sets of seeds in Markov models. This algorithm, which uses dynamic programming on a finite automaton in the style of [19], generalizes and accelerates the algorithm devised for the design of PatternHunter's seed. Section 3 investigates the relative detection probabilities of different seeds for ℓ-mer similarities and elucidates the structure of seed space by proving the existence of seeds that are asymptotically optimal with increasing length ℓ. Section 4 describes **Mandala**, a tool to design near-optimal seeds for similarity models derived empirically from a collection of biosequences. Section 5 presents controlled trials using human-mouse genomic sequence comparisons to illustrate the practical utility of discontiguous seeds and of our design methods. Finally, Section 6 concludes and indicates directions for future work.

2. AN EXACT ALGORITHM FOR SEED DETECTION PROBABILITIES

To design seeds for the resource-constrained similarity search paradigm, we must first define a measure of goodness for seeds and show how to evaluate it. Although the measure of goodness – sensitivity to interesting biosequence similarities – is straightforward, evaluating it efficiently is computationally challenging.

2.1 Problem Definition

Let C be a collection of genomic sequences. We seek all "interesting" ungapped alignments of some fixed length ℓ between pairs of substrings of C. In a BLAST-like algorithm, each such alignment serves as a starting point for gapped extension. A *similarity*, or aligned pair of ℓ-mers, consists of ℓ pairs of bases, each of which may be a match or a mismatch. Alternatively, it may be viewed as a string of ℓ bits, with a 1 wherever two bases match and a 0 where they fail to match.

A similarity is modeled by a kth-order Markov process $\mathcal{M}$ that gives the probability that the next bit seen will be 1 (i.e. a matching pair of bases) given the values of the previous k bits. The zeroth-order marginal probabilities of $\mathcal{M}$ correspond to the similarity's overall degree of conservation, while its higher-order marginals can reflect specific patterns of conservation. For example, similarities in coding sequence often exhibit a pattern of two matches followed by a mismatch, corresponding to conservation of the underlying protein with silent mutations at third base positions of codons.

Our goal is to devise a *seed* π, which is an ordered list of w positions $\{x_1 \ldots x_w\}$. We call w the *weight* of π, also denoted $|\pi|$, while its *span* is the distance $x_w - x_1 + 1$. We say that π *detects* a similarity S if, for at least one offset j, $S[j + x_i] = 1$ for $1 \leq i \leq w$. That is, every position of S inspected by π at offset j must contain matching bases. Seeded alignment algorithms enumerate all pairs of locations in the input C that exhibit matching bases in the pattern prescribed by π, so they are guaranteed to detect every similarity that π detects. If we instead devise a set Π of $n > 1$ patterns, then Π detects S if at least one of its component patterns detects S.

The computational cost of seeded alignment is controlled by the seed weight w, which largely determines its false positive rate in the absence of a similarity, and the number of seeds n to be used. Following [17], we assume that the investigator sets these parameters *a priori* to match available computing resources. The problem, then, is to find a set Π of n seeds of weight w that maximizes sensitivity to ℓ-mer similarities S from model $\mathcal{M}$. That is, we want the set Π that maximizes the *detection probability*

$$\Pr_{S \sim \mathcal{M}} [\Pi \text{ detects } S] .$$

In practice, the similarity length ℓ is less than 100 bases; it represents the typical distance between indels in the alignments of interest. BLAST-like algorithms typically use a seed weight w of 10-15.

2.2 Computing Detection Probabilities

Figure 1 illustrates the key difficulty in computing detection probabilities accurately enough to differentiate among seeds of a given weight. A seed is applied at all possible offsets into a similarity, and we wish to compute the probability that it matches at *at least one* such offset. While the expected number of matches is identical for all seeds of the same weight and span, the probability of at least one match varies because the probabilities of matches at different off-

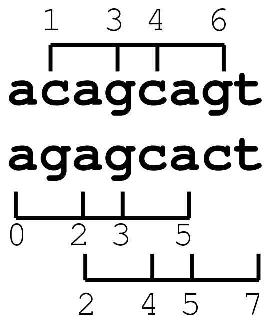

Figure 1: applying a seed, in this case $\{0,2,3,5\}$, at multiple offsets can cause it to inspect overlapping sets of sequence positions.

sets are not independent. For example, the similarity in the figure has two matches, at offsets 0 and 2, which share two of four positions in common. Even in an i.i.d. random model $\mathcal{M}$, the chance of seeing a match at one offset depends on whether a match is observed at the other. The precise pattern of overlap, and hence of dependence, varies with the pattern of positions inspected by the seed.

We cannot simply ignore overlap in analyzing seeds. Seed matches in biosequence similarities are not (and should not be!) rare events, so Poisson approximation does not accurately estimate detection probabilities. We could instead consider *part* of a seed's overlap structure using an inclusion-exclusion-based approximation to the detection probability that considers only pairwise overlap. We have explored this approximation and have found that it yields a simple criterion for approximately optimal seeds when the model $\mathcal{M}$ is of order zero. However, it is not clear how to extend the criterion to Markov models or to sets of seeds.

Our algorithm to compute detection probabilities encodes the overlap structure of a seed or set of seeds into a deterministic finite automaton (DFA). Considering similarities as bit strings, a seed π describes a (possibly degenerate) pattern of 1's in those strings. We compute the probability of seeing π by first constructing a DFA $\mathcal{A}_\pi$ that accepts strings containing it, then computing the probability that $\mathcal{A}_\pi$ accepts a string chosen at random from the model $\mathcal{M}$. This approach has been used by Nicodéme et al. [19] and by Tompa [24] to compute occurrence probabilities for degenerate words in sequences. Previous algorithms to compute detection probabilities, including the one devised for PatternHunter and a later improvement by Keich et al. [12], also use dynamic programming, though not on a DFA; it is not clear whether they generalize to multiple seeds or to Markov models.

Let π be a seed with weight w and span s, and let Q_π be the set of all 2^{s-w} s-bit strings that match π. Following [19], we could construct a regular expression for Q_π and compile it into the desired DFA; however, the following explicit construction gives better intuition for the shape and size of $\mathcal{A}_\pi$. First, construct a trie T_π from the strings of Q_π. T_π may be viewed as a DFA that accepts precisely Q_π.

We next convert the trie T_π to a DFA $\mathcal{A}^0_\pi$ that accepts any input containing a string from Q_π as a suffix. $\mathcal{A}^0_\pi$ can be built efficiently from T_π using the Aho-Corasick algorithm [1], which adds *failure links* to T_π that indicate, for any state corresponding to an input string α, the state corresponding to the longest proper suffix of α. If we find that we cannot follow a trie edge out of state s on a given input, we instead follow the path of failure links out of s until we *can* continue with a trie edge, or until we reach the start state. Once the failure link for state s is known, we follow it if needed to determine the proper transition out of s on a 0 bit. (Note that for non-accepting states, we never fail on seeing a 1 bit.) If we add failure links to T_π in breadth-first order from the start state, the full path of such links out of state s is well-defined when s is first processed.

The desired DFA $\mathcal{A}_\pi$ must accept every input containing a string of Q_π as a *substring*, not just as a suffix. To form $\mathcal{A}_\pi$ from $\mathcal{A}^0_\pi$, simply make each accepting state of the latter an absorbing state, so that the DFA accepts forever once it sees a seed match. The accepting states may now be collapsed into one state q_a.

By construction, $\mathcal{A}_\pi$ accepts an input similarity S, represented as a bit string, iff π detects S. The following dynamic programming algorithm now computes the probability that the DFA accepts a random similarity of length ℓ from a kth-order Markov model $\mathcal{M}$. Let δ be a bit string of length k. For a state q, let $\Phi_b(q)$ be the set of all states that transition to q on bit b. Define $P(q,t,\delta\cdot b)$ to be the probability of reaching state q after reading t bits of an input S, *the last* $k+1$ *of which are* $\delta\cdot b$. Then with $k'=\min\{k,t\}$ we have

$$P(q,t,\delta\cdot b)=\Pr(S[t]=b\mid S[t-k'\ldots t-1]=\delta)\times\sum_{q'\in\Phi_b(q)}\sum_{b_0\in\{0,1\}}P(q',t-1,b_0\cdot\delta).$$

Note that $\Pr(S[t]=b\mid S[t-k'\ldots t-1]=\delta)$ is given by the model $\mathcal{M}$; for $t<k$, δ is of length t, and the probability is given by $\mathcal{M}$'s lower-order marginals. We initialize the recurrence with $P(q_0,0,0)=1$ for the start state q_0 and set all other probabilities to 0. Finally, after ℓ steps, we return the sum over all $k+1$-mer bit strings $\delta\cdot 1$ of $P(q_a,\ell,\delta\cdot 1)$.

The size of T_π, and hence of $\mathcal{A}_\pi$, is at most $s2^{s-w}$, the total length of Q_π; for seeds, the trie-size bound can be improved to $w2^{s-w}$. The Aho-Corasick construction runs in time linear in the trie size, so $\mathcal{A}_\pi$ can be built in worst-case time $\Theta(w2^{s-w})$. Although some states of $\mathcal{A}_\pi$ may have many parents, the total number of parents over the entire DFA is simply its total number of transitions, which is at most twice the number of states. Hence, each step of dynamic programming takes time $\Theta(w2^{s-w}2^k)$ for a kth-order model. The full computation therefore runs in time $\Theta(w2^{s-w+k}\ell)$, which is faster by a factor of s^2/w than the best previous algorithm for detection probabilities [12] (which works only for $k=0$). A C++ implementation of our algorithm, applied with $\ell=64$, a seed π with $w=11$ and span $s=18$ (the same ℓ, w, and s as for PatternHunter) and order $k=5$, runs in a few tens of milliseconds on a 2.5 GHz Intel Pentium IV workstation.

We make three further observations about our algorithm. Firstly, we may extend it to work with a set of seeds Π by beginning with the set $Q_\Pi=\cup_{\pi\in\Pi}Q_\pi$. Secondly, the DFA $\mathcal{A}_\pi$ need not be minimal; indeed, it is usually 3-30 times larger than its minimal equivalent. The majority of time is in practice spent computing probabilities, so minimizing $\mathcal{A}_\pi$ by Hopcroft's algorithm [10] prior to dynamic programming improves overall speed. Finally, the algorithm's cost can be reduced to $\Theta((2^k+w2^{s-w})\ell)$ (see Appendix A for details); however, we did not implement this last speedup because it appears incompatible with standard DFA minimization.

3. STRUCTURE IN SEED SPACE

When is one seed more sensitive than another? The answer seems to depend in a complicated way on both the parameters of $\mathcal{M}$ *and* the similarity length ℓ. For example, while PatternHunter's seed outperforms a contiguous word π_c of the same weight in a zeroth-order model with $\ell = 64$ and 70% identity, it can be shown that for short enough ℓ or sufficiently low identity[2], π_c becomes optimal. Such parameter-dependent irregularities complicate comparisons among seeds. In this section, we consider which if any general statements we can make about the superiority of some classes of seed over others.

For a similarity S of length ℓ, define $E_\ell(\pi)$ to be the event that π detects S at *some* offset and $E^c_\ell(\pi)$ to be the complementary event. (We drop the π, writing E_ℓ, when π is clear from context.) Consider a zeroth-order similarity model $\mathcal{M}$ with p denoting its degree of conservation, i.e. the probability that a pair of bases match.

The following claim (See Appendix A for proof) indicates that at least *some* general comparisons are possible among classes of seeds. Call a seed *uniformly spaced* if its positions form an arithmetic progression with difference > 1, e.g. $\{0, 2, 4, 6, \dots\}$.

Claim 1. *If π is a uniformly spaced seed, then, for any ℓ and any zeroth-order model $\mathcal{M}$,* $\Pr[E_\ell(\pi)] < \Pr[E_\ell(\pi_c)]$.

3.1 Asymptotic Structure

While some comparisons among seeds hold for any fixed similarity length ℓ, others, like the apparent optimality of the contiguous seed π_c, hold only for small ℓ. To avoid irregularities that occur for small ℓ, we turn to sensitivity measures that hold *asymptotically*, that is, as ℓ grows large. Asymptotic results can elucidate properties of seed space that are not apparent for small ℓ, and we may hope that they apply at least approximately to the range of ℓ assumed in practice.

Claim 2. *For any seed π there exist $\beta > 0$ and $\lambda > 0$ (both of which depend on π) such that $\Pr[E^c_\ell]/\lambda^\ell \longrightarrow \beta$ as $\ell \to \infty$.*

Proof. For this and following results, we need the following *matrix representation* of the DFA $\mathcal{A}_\pi$, after [19]. We assume here that the model $\mathcal{M}$ is zeroth-order, relegating the proof of the kth-order case to Appendix A. Let $N+1$ be the number of states of the DFA, and let A_π be an $N \times N$ matrix indexed by all states of $\mathcal{A}_\pi$ except its accepting state, such that $A_\pi(s,t)$ is the probability of transitioning from state s to state t of the DFA upon reading a base pair generated from $\mathcal{M}$. Note that A_π is sparse, entrywise non-negative, and sub-stochastic.

Let $\boldsymbol{e}_1 = (1, 0, 0, \dots, 0) \in \mathbb{R}^N$. Then $\boldsymbol{e}_1 A$ yields the distribution of the automaton states one step (or one bit) after starting from the entry state q_0. More generally, $\boldsymbol{e}_1 A^\ell$ yields the distribution on the states after ℓ steps[3]. This non-negative distribution has total mass less than 1; indeed, the difference is exactly $\Pr[E_\ell]$.

[2]The latter result follows by inspecting the inclusion-exclusion form of the detection probability.

[3]This formulation introduces another way to compute $\Pr[E_\ell]$ in time $O(w 2^{s-w} \ell)$ by noting that $\boldsymbol{e}_1 A^n = (\boldsymbol{e}_1 A^{n-1}) A$ [24].

Assume for a moment that A is primitive in the sense that there exists a j such that A^j is an entrywise strictly positive matrix. Since A^j is a sub-stochastic positive matrix, by the Perron-Frobenius theorem it has a multiplicity-1, positive eigenvalue $\lambda^j < 1$ such that any other eigenvalue of A^j satisfies $|\mu| < \lambda^j$ [15]. Moreover, the left eigenvector $\boldsymbol{u}$ corresponding to λ^j is positive, and without loss of generality $\boldsymbol{u}$'s entries sum to 1. Note that $\boldsymbol{u}$ is also a left eigenvector of A with $\boldsymbol{u}A = \lambda \boldsymbol{u}$. Similarly, there exists a corresponding positive right eigenvector $\boldsymbol{w}$ for which $A\boldsymbol{w} = \lambda \boldsymbol{w}$). Let $\boldsymbol{e}_1 = \beta \boldsymbol{u} + \boldsymbol{v}$, where $\boldsymbol{v}$ is a linear combination of the other (possibly generalized) left eigenvectors of A^j. Then

$$0 < \boldsymbol{e}_1 \boldsymbol{w} = \beta \boldsymbol{u}\boldsymbol{w} + \boldsymbol{v}\boldsymbol{w} = \beta \boldsymbol{u}\boldsymbol{w},$$

and, since $\boldsymbol{u}\boldsymbol{w} > 0$, $\beta > 0$. Thus, assuming A is primitive we have

$$\boldsymbol{e}_1 A^\ell = \beta \lambda^\ell \boldsymbol{u} + \boldsymbol{v} A^\ell = \beta \lambda^\ell \boldsymbol{u} + o(\lambda^\ell),$$

which immediately implies the claim.

Finally, notice that for a seed π of span s, any non-terminal state of $\mathcal{A}_\pi$ is accessible from the entry state by at most s steps and vice versa. Because we can stay at the entry state for an arbitrary number of steps, A^{2s} is entrywise positive, whence A is primitive. $\square$

Remark 1. *A version of this claim can be traced back to [19] with many missing details which are provided here for completeness.*

Definition 1. *A seed π is asymptotically worse than a seed π', denoted $\pi \prec \pi'$, if $\lim_\ell \Pr[E^c_\ell(\pi)]/\Pr[E^c_\ell(\pi')] > 1$. Similarly, π and π' are asymptotically equivalent, denoted $\pi \simeq \pi'$, if $\lim_\ell \Pr[E^c_\ell(\pi)]/\Pr[E^c_\ell(\pi')] = 1$, and $\pi \preceq \pi'$ if $\pi \prec \pi'$ or $\pi \simeq \pi'$.*

Remark 2. *Using Claim 2, it is easy to verify that this relation defines a linear order on the set of all seeds. A seed π's asymptotic performance is entirely determined by $\lambda(\pi)$ and $\beta(\pi)$.*

The following claim indicates that the contiguous seed is not very promising for large ℓ. From now on we assume S is generated by a zeroth-order model.

Claim 3. *Let π_c be the contiguous seed of weight w and π any seed with $|\pi| = w$. Then*

$$\lambda(\pi_c) \ge \lambda(\pi). \tag{1}$$

Proof. In [12, Claim 1] the authors prove that

$$\Pr[E_\ell(\pi_c)] \le \Pr[E_{\ell+s-w}(\pi)].$$

It follows from Claim 2 that

$$\lim_\ell \frac{\beta(\pi_c)\lambda(\pi_c)^\ell}{\beta(\pi)\lambda(\pi)^{\ell+s-w}} \ge 1,$$

which proves the claim. $\square$

We conjecture that equality holds in (1) if and only if π is a uniformly spaced seed. The "if" part follows immediately from Claim 1 and the preceding claim[4]. If this conjecture is

[4]It is not hard to show, in the spirit of Claim 1, that for any uniformly spaced seed π, $\pi \prec \pi_c$; that is, although $\lambda(\pi) = \lambda(\pi_c)$, $\beta(\pi) > \beta(\pi_c)$.

true, than any non-uniformly spaced seed is asymptotically better than the contiguous seed. While we cannot yet prove this conjecture, we *can* show that π_c cannot be asymptotically optimal (see Appendix A for proof):

CLAIM 4. *Let* $\pi = \{0, 1, \ldots, w-2, w\}$ *where* $|\pi| = |\pi_c| = w > 2$. *Then* $\lambda(\pi_c) > \lambda(\pi)$, *and in particular* $\pi_c \prec \pi$.

REMARK 3. *While one might be suspicious of the practicality of asymptotic results, in this case one can prove that if* $w \geq 4$ *then for* $\ell = w+3$, $p > \frac{1}{2}$, *and the seed* π *given in Claim 4,* $\Pr[E_\ell(\pi)] > \Pr[E_\ell(\pi_c)]$. *The same inequality holds if* $w \geq 5$, $\ell = w+4$, *and* $p > \frac{1}{3}$.

4. MANDALA: FAST, PRACTICAL SEED DESIGN

The results of Section 3 yield clues to the structure of seed space, suggesting that contiguous and uniformly spaced seeds are likely to be among the least sensitive choices for seeded alignment. However, to maximize the impact of seed design in practice, we must still solve the original optimization problem: find the *best* seeds for similarities of some finite length ℓ from a model $\mathcal{M}$. Moreover, we must define the model $\mathcal{M}$ so as to capture specific properties of the similarities being sought. We have therefore constructed **Mandala**, a software tool that addresses both seed selection and model design.

4.1 Seed Selection

Other than brute-force enumeration and evaluation, no procedure is known to find the optimum seed of fixed weight w and span $\leq s$ for a model $\mathcal{M}$. Enumeration is fast enough to derive single seeds in simple similarity models but rapidly becomes impractical as the number of simultaneous seeds, the model complexity, or the maximum span increases. Mandala therefore sacrifices global optimality for much faster design using the following *local* search method.

Let $\pi = \{x_1 \ldots x_w\}$ be the current seed, with all $x_i \leq s$. To avoid generating shifted versions of the same seed, we fix $x_1 = 0$. The local neighborhood of π is the set of all seeds π' that differ from π in exactly one of $x_2 \ldots x_w$, with the differing position chosen from among the unused set $\{1 \ldots s-1\} - \pi$. Using this neighborhood definition and the probability calculation of Section 2 as an evaluation function, we perform hill climbing with random restart in seed space to find a near-optimal seed. To design a set of simultaneous seeds Π, we simply extend the neighborhood definition to encompass all sets Π' in which one seed $\pi'_i \in \Pi'$ differs from the corresponding $\pi_i \in \Pi$ in a single position.

The speedup of Mandala over exhaustive enumeration is dramatic, even for problems in which enumeration is feasible. Using the same design constraints as Ma, Tromp, and Li ($\ell = 64$, $w = 11$, zeroth-order model with 70% identity [17]), we sought the best seed of span $s \leq 22$. Mandala with ten random restarts finished in under 20 seconds on a 2.5 GHz Intel Pentium IV workstation, versus over an hour for exhaustive enumeration. In ten trials of ten restarts each, Mandala found the globally optimal seed three times; the seven suboptimal seeds found differed from the optimum in their detection probabilities by less than 1%.

The evaluation algorithm of Section 2 is fast for small spans, but its cost grows exponentially as the span increases for fixed weight. Mandala can instead evaluate seeds on a large set of similarities generated at random from $\mathcal{M}$. A usefully precise Monte Carlo estimate of a seed's detection probability requires upwards of 10^6 trials, which can take several seconds. However, Mandala accelerates the evaluation of multiple moves in local search using Wald's Sequential Probability Ratio Test [25] to rapidly terminate evaluation of moves with a high probability of being downhill.

4.2 Training a Similarity Model

Mandala is intended to optimize seeded alignment for particular applications of similarity search, such as detection of coding similarities or specific repeat families. Properties of the similarities of interest are captured by the model $\mathcal{M}$. A key question, therefore, is how to train this model.

We train the model $\mathcal{M}$ by example using an empirically derived set of ℓ-mer similarities. These similarities could be obtained using an existing seeded alignment tool, but such tools would bias the training set toward whichever seed they use. We instead sample the set of ℓ-mer similarities in a collection of sequences using a variant of Buhler's LSH-ALL-PAIRS algorithm [4], which uses a large number of random seeds to sample ℓ-mer similarities with probability proportional to their percent identity regardless of how their substitutions are arranged. To avoid overloading the training set with highly conserved similarities (which are easy to find no matter which seed is used), we limit range of identities returned (e.g. to 70-75%) and use rejection sampling within this range to equalize the sampling rates for different identity levels. Our sampling procedure is efficient enough to work with 10^7-10^8 bases of genomic sequence; hence, producing large training sets for $\mathcal{M}$ has not been a problem in practice.

5. EXPERIMENTAL RESULTS

Mandala's design criteria produce seeds that work well in theory, but we sought further evidence that these seeds are also useful in practice. We investigated the following questions. First, do discontiguous seeds, and Mandala's seeds in particular, *by themselves* benefit seeded alignment? Second, does empirical evidence support our conjecture in Section 3 that contiguous seeds are among the *least* sensitive choices? Third, how well do our Markov similarity models $\mathcal{M}$ predict seeds' actual performance?

Previously published experiments [17] have not controlled for the effect of extraneous algorithm changes, e.g. in gapped extension, on alignment performance. To isolate the contribution of seed choice, we therefore devised the following test setup. Starting with assemblies of the human and mouse genomes obtained from Ensembl [11], we extracted 428 pairs of regions that had been annotated as homologous with no major rearrangements[5]. These regions, spanning roughly 2.5 gigabases, were masked to remove repeats and low-complexity DNA, then divided into coding and noncoding parts on the basis of annotated exons. Our experiments measured the number of nonoverlapping gapped alignments found when comparing homologous pairs of regions.

Sequences were compared using BLAST-like seeded alignment. We modified the `lsh` program from the Projection Genomics Toolkit [5] to use externally specified seeds. The program performed gapped extension using banded Smith-

[5]Homology mappings were kindly provided by Michele Clamp of the Sanger Center.

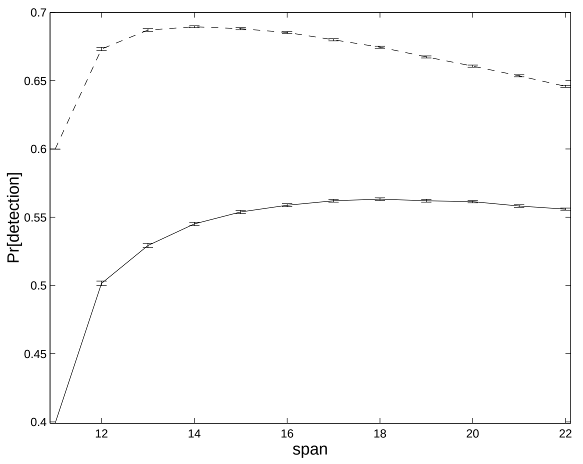

Figure 2: average detection probabilities given by theoretical models for random seeds of weight 11 and various spans. Solid line: zeroth-order noncoding model $\mathcal{M}_0$; dashed line: fifth-order model $\mathcal{M}_5$. Error bars are 95% confidence intervals for each average over 1000 trials.

Waterman with the default scoring function and significance threshold used by PipMaker [21], whose parameters are optimized for human-mouse comparison. Other than the choice of seed, no property of the alignment tool was changed between experiments. Appendix B gives further details of our experimental setup.

5.1 Noncoding DNA Sequence

In the interest of time, we tested seed performance in noncoding DNA using 43 of our 428 regions, spanning roughly 500 megabases. We extracted 1136000 similarities with $\ell = 64$ and 70-75% identity from these regions and used them to infer a fifth-order Markov similarity model $\mathcal{M}_5$. The zeroth-order approximation to $\mathcal{M}_5$, denoted $\mathcal{M}_0$, is identical to the model of [17] except with $p \approx 0.727$ instead of 0.7.

We concentrated on weight-11 seeds, since this weight is commonly used in NCBI BLASTN. Figure 2 illustrates the theoretical average detection probability for such seeds as a function of their span. The figure supports our conjecture that nearly *any* seed is more sensitive than the contiguous seed. Because the similarity length ℓ is finite, there are preferred spans for each model that optimize the tradeoff between maximizing the number of offsets available for detection and minimizing seed overlap between offsets. For $\mathcal{M}_0$ (solid line), the best spans are 17-19, consistent with the results of [17]; however, the full $\mathcal{M}_5$ (dashed line) prefers shorter spans of 13-15, suggesting that its similarities exhibit tighter grouping of their matching base pairs.

We tested the performance of five seeds, which are given in Table 1: the contiguous 11-mer π_c, the contiguous 10-mer π_{c10}, the PatternHunter seed π_{ph}, and the best seeds π_{N_0} and π_{N_5} found by Mandala in ten restarts for models $\mathcal{M}_0$ and $\mathcal{M}_5$, respectively, with span ≤ 22 (by which point sensitivity declines according to Figure 2). We note that the best seeds found are predicted to be noticeably more sensitive than average seeds of the same span.

All discontiguous seeds found substantially more alignments in practice than did π_c; in particular, π_{ph} found 13.7% more alignments than π_c and 2.7% more than even π_{c10}. The extra alignments found were among the most difficult to detect, with scores up to a few times times the threshold score but far below the most "obvious" high-scoring alignments. While the higher-order model did not confer significant extra sensitivity in practice, it did result in an equally good seed of shorter span (15 vs 18 for π_{ph}), which is desirable because it reduces the chance of missing similarities with frequent indels. All discontiguous seeds incurred a slight speed penalty versus π_c, largely because their false positive rates (i.e. the number of seed matches that did not lead to a significant gapped alignment after extension) were up to 33% higher. However, the worst slowdown observed was less than 6.5%, considerably less than the $> 50\%$ slowdown observed going from π_c to the shorter π_{c10}. Indeed, the false positive rate for π_{c10} may be expected to be around 400% higher than for π_c, which (as observed in [17]) makes seed design a more attractive option to increase sensitivity than simply reducing a seed's weight.

Finally, we tested Mandala's ability to design simultaneous seeds by finding two seeds of weight 12, which should have sensitivity comparable to a seed of weight 11 while exhibiting roughly half the false positive rate. Mandala produced the pair of seeds π_1 and π_2, which together were roughly 2.2% more sensitive than the best single seed found. The cost of finding seed matches dominates the practical cost of seeded alignment, so simultaneous seeds are most appropriate when indexing a sequence database offline or for hardware that can use multiple seeds in parallel.

5.2 Coding DNA Sequence

We tested seeds for coding DNA using annotated coding exons from all 428 homologous region pairs. We extracted 2231000 64-mer similarities with 70-75% identity and again trained a fifth-order model, $\mathcal{M}_5^c$.

A key benefit of Mandala is that it exploits structure present in its set of training similarities. For coding similarities, the natural structure is that of codons. Biological intuition therefore suggests that the best seed design should be a repeating "110" pattern that ignores every third position; given the right offset into a similarity, such a seed would consider only first and second base positions of each codon. We investigated whether Mandala could automatically exploit this structure in $\mathcal{M}_5^c$.

Table 2 lists the seeds tested on our coding data set. Besides π_c and π_{ph}, we tried the intuitive coding seed π_{110} and the best weight-11 seed π_{C_5} found by Mandala in ten restarts on model $\mathcal{M}_5^c$ with span $s \leq 22$. The seed π_{C_5} is structured similarly but not identically to π_{110}, starting with three matches in a row and skipping the interval 3-7 entirely.

Our tests of the above seeds on real coding DNA reveal both strengths and limitations of Mandala's approach. On the one hand, Mandala's chosen pattern π_{C_5} was slightly more sensitive than π_{110} and *did* capture the intuitive codon structure of the training similarities. On the other hand, the model $\mathcal{M}_5^c$ falsely favored π_{ph} over π_{110} and predicted a large advantage for π_{C_5} over π_{110} that was not actually observed. Mandala penalized seeds with highly regular structures, as might be expected given the result of Claim 1; in this instance, that penalty proved unjustified.

Despite Mandala's limitations, we hypothesized that it had still made an important observation: the seed π_{C_5} skips

Seed	Pattern	Pr[detection]	Alignments Found	Time (s)
π_c	$\{0,1,2,3,4,5,6,7,8,9,10\}$	0.600	66419	15802
π_{c10}	$\{0,1,2,3,4,5,6,7,8,9\}$	0.707	73539	24129
π_{ph}	$\{0,1,2,4,7,9,12,13,15,16,17\}$	0.691	75518	16717
π_{N_0}	$\{0,1,2,4,7,8,11,13,16,17,18\}$	0.683	75231	16225
π_{N_5}	$\{0,1,2,3,5,6,7,10,12,13,14\}$	0.709	75547	16817
$\pi_1+\pi_2$	$\{0,1,2,4,5,9,14,16,17,18,19,20\}+$ $\{0,1,2,3,4,6,7,8,10,11,12,13\}$	0.744	77211	22033

Table 1: performance of seeds on noncoding DNA. Pr[detection] **is detection probability in our fifth-order noncoding model. Gapped alignments found and running times are on 500 megabases of homologous noncoding regions from human and mouse.**

Seed	Pattern	Pr[detection]	Alignments Found	Time (s)
π_c	$\{0,1,2,3,4,5,6,7,8,9,10\}$	0.565	199760	9229
π_{ph}	$\{0,1,2,4,7,9,12,13,15,16,17\}$	0.719	216713	10001
π_{110}	$\{0,1,3,4,6,7,9,10,12,13,15\}$	0.699	220563	10233
π_{C_5}	$\{0,1,2,8,9,11,12,14,15,17,18\}$	0.744	221413	10202
π_{sp}	$\{0,1,3,4,9,10,12,13,15,16,18\}$	0.738	225680	10486

Table 2: performance of seeds on coding DNA. Pr[detection] **is detection probability in our fifth-order coding model. Gapped alignments found and running times are on coding portions of 2.5 gigabases of homologous regions from human and mouse. The sensitivity of 0.744 for π_{C_5} is only slightly below the optimum of 0.747 found by exhaustive enumeration.**

an entire codon (positions 5-7). Figure 2 shows that skipping even one position in noncoding DNA (span 12) can substantially improve sensitivity, so we guessed that the same observation applied to skipping codons in coding DNA. We therefore manually designed a new "sparse 110" seed π_{sp} that kept both the sparsity of π_{C_5} and the regularity of π_{110}. The new π_{sp} proved 2.3% more sensitive than π_{110}, a respectable fraction of the latter's 10% improvement over π_c. We conclude that, even when the truly optimal seed eludes Mandala, it still provides useful insight into seed design.

5.3 Influence of Model Order

Our experiments show that Mandala's choice of seed is significantly influenced by the order of its Markov similarity model. We quantified the utility of higher-order models for improving seed sensitivity by testing whether seeds trained using more complex models were measurably superior to those trained using less complex models.

Starting with the Markov models $\mathcal{M}_5$ and $\mathcal{M}_5^c$, we constructed simplified models $\{\mathcal{M}_0 \ldots \mathcal{M}_4\}$ and $\{\mathcal{M}_0^c \ldots \mathcal{M}_4^c\}$ of orders 0 through 4 by using only the lower-order marginal probabilities of the original fifth-order models. We applied Mandala to these simplified models to find a near-optimal seed of weight 11 and span at most 22 for each model with $\ell = 64$. We then evaluated the detection probabilities of the seeds trained on the lower-order models against the full fifth-order models $\mathcal{M}_5$ and $\mathcal{M}_5^c$. If the fifth-order models capture useful information about the distribution of similarities that is absent from their lower-order approximations, then seeds trained on the full models should prove more sensitive than those trained on the approximations.

Figure 3 shows the results of running the above experiment 100 times for each lower-order model. For noncoding models (solid line), seeds trained on the second-order approximation $\mathcal{M}_2$ behave nearly identically to those trained on the full $\mathcal{M}_5$, indicating that the latter model contains relatively little useful information above second order. In

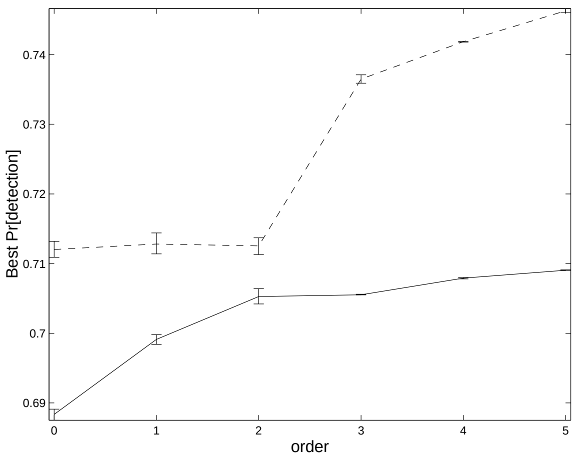

Figure 3: detection probabilities for best seeds found by Mandala when trained on lower-order approximations to our fifth-order noncoding model $\mathcal{M}_5$ (solid line) and coding model $\mathcal{M}_5^c$ (dashed line). Error bars indicate 95% confidence intervals over 100 experiments.

contrast, seeds trained on the second-order coding model $\mathcal{M}_2^c$ perform substantially worse than those trained on $\mathcal{M}_5^c$, and indeed about equally to seeds trained on the *zeroth*-order $\mathcal{M}_0^c$ (dashed line). The first and largest information gain for coding models occurs between second and third order, when the model can "look back" one complete codon length.

Overall, the results of Figure 3 suggest that our noncoding similarity model primarily exploits near-neighbor correlations in the occurrence of matching bases, while the coding model indeed exploits correlation arising from codon structure. Such observations could prove helpful in designing similarity models based on formalisms other than simple Markov processes, such as the PIP-based models of Li and Miller [16].

6. CONCLUSIONS AND DIRECTIONS

Seed design materially affects the sensitivity of seeded alignment algorithms. Sensitivity improvements of even a few percent are worthwhile, given the heavy use of these techniques in practice (over 10^5 queries/day to the NCBI BLAST server). We have described algorithms and software tools to optimize the choice of seeds for particular alignment problems and have demonstrated that the right seed choice confers practical benefits. The Mandala software, with source code, may be obtained from *http://www.cs.wustl.edu/∼jbuhler/mandala/*.

We plan to pursue several directions to improve our understanding and exploitation of seed design. First, we will continue the work of Section 3 to gain insight into the structure of seed space. Second, we will seek to overcome current limits on our ability to design several seeds at once, so as to fully address the question of how to design multiple seeds for static indexing. Currently, such designs are limited to at most 3-4 seeds of span ≤ 25, both because neighborhood size scales as maximum span times the number of seeds and because the optimizer converges much more slowly when designing several seeds at once. We will investigate alternate optimization strategies, as well as restrictions on which seeds can be used simultaneously (e.g. forbidding excessive overlap) to shrink the neighborhood size.

We also plan to extend the models used to evaluate seeds. Our results on coding DNA suggest that Markov models alone do not capture the interesting properties of real biosequence similarities. Other forms of model might better describe the underlying processes of conservation, such as the different codon positions of coding DNA, thereby enabling more informed seed choice. If these models become highly parametric, we may need to apply complexity corrections to decide which and how many parameters to infer. We also plan to extend our similarity models to distinguish between different classes of substitution, in particular transitions and transversions, since previous work [8, 6] indicates that exploiting these signals can increase sensitivity and reduce noise. Finally, we will construct models of *multiple* alignments to help design seeded alignment tools to compare three or more genomes at once.

Acknowledgments

The authors wish to thank Tom Joseph, for help with the infrastructure to run the experiments; Ming Li, Tomas Vinar, and Brona Brejova, for encouragement to work on finding asymptotically optimal seeds; and Steven Altschul and Alejandro Schäffer, for helpful suggestions on Mandala's design.

7. REFERENCES

[1] A. Aho and M. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18:333–40, 1975.

[2] S. F. Altschul and W. Gish. Local alignment statistics. *Methods: a Companion to Methods in Enzymology*, 266:460–80, 1996.

[3] 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.

[4] J. Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17:419–28, 2001.

[5] J. Buhler. *Search Algorithms for Biosequences Using Random Projection*. PhD thesis, University of Washington, Seattle, WA, 2001.

[6] J. Buhler. Provably sensitive indexing strategies for biosequence similarity search. In *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB02)*, pages 69–76, Washington, DC, 2002.

[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] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, et al. Alignment of whole genomes. *Nucleic Acids Research*, 27(11):2369–76, 1999.

[10] J. Hopcroft. *An $n \log n$ algorithm for minimizing states in a finite automaton*, pages 189–96. Academic Press, 1971.

[11] T. Hubbard, D. Barker, E. Birney, G. Cameron, et al. The Ensembl genome database project. *Nucleic Acids Research*, 30:138–41, 2002.

[12] U. Keich, B. Ma, M. Li, and J. Tromp. On spaced seeds for similarity search. 2002. Preprint.

[13] W. J. Kent. BLAT: the BLAST-like alignment tool. *Genome Research*, 12:656–64, 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] P. D. Lax. *Linear Algebra*. Pure and Applied Mathematics. John Wiley & Sons, Inc., New York, 1997.

[16] 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.

[17] B. Ma, J. Tromp, and M. Li. PatternHunter – faster and more sensitive homology search. *Bioinformatics*, 18:440–5, 2002.

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

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

[20] P. Pevzner and M. S. Waterman. Multiple filtration and approximate pattern matching. *Algorithmica*, 13:135–54, 1995.

[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] A. F. Smit and P. Green. Repeatmasker, 1999. http://ftp.genome.washington.edu/RM/RepeatMasker.html.

[23] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–97, Mar. 1981.

[24] M. Tompa. An exact method for finding short motifs in sequences, with applications to the ribosome binding site problem. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB '99)*, pages 262–71, Heidelberg, Germany, 1999. AAAI Press.

[25] A. Wald. *Sequential Analysis*. Wiley, New York, 1947.

APPENDIX

A. PROOFS

A.1 Smaller Automaton Construction for Markov Case

The following construction builds a new DFA $\mathcal{B}_\pi$ equivalent to $\mathcal{A}_\pi$ such that each state can be reached by at most one possible k-bit history. First, note that every state of $\mathcal{A}_\pi$ of depth $\geq k$ in the trie T_π already has this property. For each state s of $\mathcal{A}_\pi$, let the *label* $\alpha(s)$ be the string labeling the path from the trie root to s.

Define new states s_j for each k-bit string j such that j is not $\alpha(s)$ for any state at depth k of T_π. Although these states are not in T_π, we define their labels $\alpha(s_j) = j$. On bit b, each new state s_j transitions to the state whose label $j' = j[2 \dots k] \cdot b$ (either a new state $s_{j'}$ or an existing state of $\mathcal{A}_\pi$ whose label is j'). Moreover, if an existing state t of $\mathcal{A}_\pi$ with depth $\geq k$ transitions on bit b to a state at depth $< k$, that transition is changed to target the state $s_{j''}$, where the label j'' is the last $k-1$ bits of $\alpha(t)$ plus b. Now delete the (unused) old states of $\mathcal{A}_\pi$ with depth $< k$. Finally, augment the new DFA with a full tree of 2^{k-1} *initial* states, the root of which is the new start state, such that the leaf of the tree reached on input string α transitions on bit b to the state labeled $\alpha \cdot b$ (which may be an old or a new state).

We can divide the non-initial states s of the new $\mathcal{B}_\pi$ into equivalence classes $\Phi(t)$, where t indexes the states of $\mathcal{A}_\pi$. The relation is as follows: if s was a preexisting state t of $\mathcal{A}_\pi$, $s \in \Phi(t)$. For all new states s_j, $s_j \in \Phi(t)$ iff t is the state of $\mathcal{A}_\pi$ with depth $< k$ whose label forms the longest suffix of j. It can be shown that for any string S of length $\geq k$, if the original DFA $\mathcal{A}_\pi$ ends up in state s after reading S, then the new DFA $\mathcal{B}_\pi$ ends up in some state in $\Phi(s)$. Provided that $\mathcal{A}_\pi$ does not accept on any strings shorter than k bits, this proves equivalence of $\mathcal{A}_\pi$ and $\mathcal{B}_\pi$. Moreover, each state of the new DFA, except for the initial tree, is by construction associated with exactly one history of k bits.

Because each transition of $\mathcal{B}_\pi$ is associated with only one k-bit history, each transition has fixed probability in a kth-order Markov model. We can therefore define a (sparse) matrix A_π corresponding to $\mathcal{B}_\pi$ as described in the proof of Claim 2. The cost of computing sensitivity in the kth-order Markov model is therefore $O((2^k + w2^{s-w})\ell)$, the cost of ℓ matrix multiplications with the sparse A_π.

A.2 Proof of Claim 1

Denote by $B_j(\pi)$ the event that the uniformly spaced seed π matches a similarity S at offset j, i.e. $B_j(\pi) = \{S[j + x_i] = 1\ ,\ \forall x_i \in \pi\}$. To simplify notation, assume that $n = 2m$ and that $\pi = \{0, 2, 4, \dots, 2(w-1)\}$. Let $D_\pi = \cup_{j=0}^{m-w} B_{2j}(\pi)$ and $F_\pi = \cup_{j=0}^{m-w} B_{2j+1}(\pi)$. Similarly, define $D_{\pi_c} = \cup_{j=0}^{m-w} B_j(\pi_c)$, define $F_{\pi_c} = \cup_{j=m}^{2m-W} B_j(\pi_c)$, and define $G_{\pi_c} = \cup_{j=m-w+1}^{m-1} B_j(\pi_c)$.

We have that $\Pr[D_\pi] = \Pr[D_{\pi_c}]$ and $\Pr[F_\pi] = \Pr[F_{\pi_c}]$. Moreover, D_π is independent of F_π, and the same holds for π_c, so

$$\begin{aligned}\Pr[E_\ell(\pi)] &= \Pr[D_\pi] + \Pr[F_\pi] - \Pr[D_\pi]\Pr[F_\pi] \\ &= \Pr[D_{\pi_c} \cup F_{\pi_c}] \\ &< \Pr[D_{\pi_c} \cup F_{\pi_c}] + \Pr[G_{\pi_c} \setminus (D_{\pi_c} \cup F_{\pi_c})] \\ &= \Pr[E_\ell(\pi_c)].\end{aligned}$$

A.3 Proof of Claim 2 in the Markovian case

When the string S is governed by a kth-order Markov chain, we define the $N' \times N'$ matrix A' as the submatrix of the A from the construction of A.1 that corresponds to all non-initial states $\mathcal{B}_\pi$. A' is primitive since, as in the proof of the zeroth-order case, we can move between any two states in $2s$ steps. We can complete the proof as before upon replacing $\boldsymbol{e}_1$ with $\boldsymbol{\mu} \in \mathbb{R}^{N'}$, the distribution of the states after the first k bits. Note that the probability distribution, $\boldsymbol{\mu}$, is supported only on the states of $\mathcal{B}_\pi$ whose labels have length k and is readily obtainable from $\boldsymbol{e}_1 A^k$.

A.4 Proof of Claim 4

Proof. Let $\boldsymbol{x} = (x_1, \dots, x_N)$ be the unique unit-mass positive eigenvector for which $\boldsymbol{x}A(\pi) = \lambda(\pi)\boldsymbol{x}$, where $A(\pi)$ is the nonnegative automaton matrix associated with π and $N = w + 2$ is its dimension. Without loss of generality, the automaton of π consists of states F_i, $i = 1 \dots w$, which correspond to a prefix of $i-1$ 1 bits and states F_{w+1} and F_{w+2} to a prefix of $w-1$ consecutive 1 bits followed by a 0 and a 1 respectively. For example, with $w = 2$ and $q = 1-p$, $A(\pi_{c_2}) = \left(\begin{smallmatrix} q & p \\ q & 0 \end{smallmatrix}\right)$, while $A(\pi) = \left(\begin{smallmatrix} q & p & 0 & 0 \\ 0 & 0 & q & p \\ q & 0 & 0 & 0 \\ 0 & 0 & q & 0 \end{smallmatrix}\right)$. Thus, with $\hat{\boldsymbol{x}} = (x_1, \dots, x_w)$ ($\boldsymbol{x}$ without its last two digits),

$$\begin{aligned}\hat{\boldsymbol{x}}A(\pi_c) = \left(q\sum_{i=1}^{w} x_i, px_1, px_2, \dots, px_{w-1} \right) = \\ (\lambda(\pi)x_1 + q(x_w - x_{w+1}), \lambda(\pi)x_2, \dots, \lambda(\pi)x_w)\,,\end{aligned}$$

since $\boldsymbol{x}A(\pi) = \lambda(\pi)\boldsymbol{x}$. It follows that for $i = 1, 2, \dots, w-1$, $px_i = \lambda(\pi)x_{i+1}$ and $q(x_{w+1} + \sum_{i=1}^{w-1} x_i) = \lambda(\pi)x_1$. If we can show that $q(x_w - x_{w+1}) > 0$ then it follows exactly as in the proof of the Perron-Frobenius theorem [15] that $\lambda(\pi_c) > \lambda(\pi)$ as stated. Invoking $\boldsymbol{x}A(\pi) = \lambda(\pi)\boldsymbol{x}$ once again, we obtain the following system of equations:

$$q(x_w + x_{w+2}) = \lambda(\pi)x_{w+1} \qquad px_w = \lambda(\pi)x_{w+2},$$

from which we deduce that $\lambda(\pi)x_{w+1} = q(1 + p/\lambda(\pi))x_w$. Hence, $x_w > x_{w+1}$ if and only if $\lambda(\pi)^2 - q(\lambda(\pi) + p) > 0$. Consider the polynomial $f(\lambda) = \lambda^2 - q(\lambda + p)$. For $p \in (0,1)$ (and $q = 1-p$) it has a unique positive root. It happens that $f(\lambda)$ is the characteristic polynomial of the matrix $A(\pi_{c_2})$ above. Since $w > 2$, $\pi_{c_2} \subset \pi$, and so the unique positive root of $f(\lambda)$, $\lambda(\pi_{c_2})$, satisfies $\lambda(\pi_{c_2}) \leq \lambda(\pi) \leq \lambda(\pi_c)$. If we can show that $\lambda(\pi_{c_2}) < \lambda(\pi_c)$ then we are done, since in that case either $\lambda(\pi) < \lambda(\pi_c)$, which is exactly our claim, or else $\lambda(\pi_{c_2}) < \lambda(\pi)$ which implies that $f(\lambda(\pi_{c_2})) > 0$ and therefore proves the claim. To show that $\lambda(\pi_{c_2}) < \lambda(\pi_c)$, we note that since $w \geq 3$ it suffices to check that $\lambda(\pi_{c_2}) < \lambda(\pi_{c_3})$, which can readily be verified using Maple or the like. $\square$

B. DETAILS OF EXPERIMENTAL SETUP

We obtained NCBI build 28 of the human genome and Release 3 of the mouse genome, along with their annotations, from Ensembl. Sequence regions were masked using the existing annotated list of interspersed repeats along with a standalone version of Dust taken from NCBI BLAST. Sequences were divided into coding exons and the remaining noncoding DNA.

Comparisons were performed with a modified version of the `lsh` program. 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.