

Efficient Large-Scale Sequence Comparison by Locality-Sensitive Hashing

Jeremy Buhler

Department of Computer Science and Engineering

University of Washington

Seattle, WA 98195-2350, USA

Appearing in *Bioinformatics* 17(5) 419–428, 2001

Abstract

Motivation: Comparison of multimegabase genomic DNA sequences is a popular technique for finding and annotating conserved genome features. Performing such comparisons entails finding many short local alignments between sequences up to tens of megabases in length. To process such long sequences efficiently, existing algorithms find alignments by expanding around short runs of matching bases with no substitutions or other differences. Unfortunately, exact matches that are short enough to occur often in significant alignments also occur frequently by chance in the background sequence. Thus, these algorithms must trade off between efficiency and sensitivity to features without long exact matches.

Results: We introduce a new algorithm, LSH-ALL-PAIRS, to find ungapped local alignments in genomic sequence with up to a specified fraction of substitutions. The length and substitution rate of these alignments can be chosen so that they appear frequently in significant similarities yet still remain rare in the background sequence. The algorithm finds ungapped alignments efficiently using a randomized search technique, *locality-sensitive hashing*. We have found LSH-ALL-PAIRS to be both efficient and sensitive for finding local similarities with as little as 63% identity in mammalian genomic sequences up to tens of megabases in length.

Availability: Contact the author at the address below.

Contact: jbuhler@cs.washington.edu

Supplementary Information: the sequences and local alignment data described in this work are available at <http://bio.cs.washington.edu/jbuhler-bioinformatics-2001/>.

Keywords: local alignment, genome annotation, locality-sensitive hashing

Introduction

Comparisons between long DNA sequences are useful in annotating conserved genomic features. Recent work with orthologous mammalian sequences (Hardison *et al.*, 1997; Loots *et al.*, 2000) has demonstrated the value of comparative analysis for detecting intronic and intergenic regions with potential functional significance. Comparing long segments from even a single organism’s genome, e.g. an entire human chromosome (Dunham *et al.*, 1999; Hattori *et al.*, 2000), can uncover novel repeated structures, including new transposable elements and members of multigene families. The rapidly increasing amount of sequence available from eukaryotic genomes will provide even more opportunities for informative comparisons.

Algorithms for long-range genomic sequence comparison must separate short local alignments of high similarity from long stretches of unrelated or diverged background sequence. Linear-space Smith-Waterman variants like Sim (Huang and Miller, 1991) directly discover alignments with both substitutions and gaps, but they are too computationally demanding to analyze multi-megabase sequences without specialized hardware. For this reason, practical sequence comparison algorithms seek only those alignments that can be found by extending a *seed*, usually a sufficiently long run of contiguous matching bases between sequences. Efficient algorithms for finding all such *exact matches* above some minimum length form a key part of local alignment search tools such as BLAST (Altschul *et al.*, 1997), Cross-Match (Green, 1994), and PipMaker (Schwartz *et al.*, 2000). Exact matches also provide the basic features used to construct fast global alignments between long sequences in tools such as MUMmer (Delcher *et al.*, 1999) and GLASS (Batzoglou *et al.*, 2000).

An exact-match-based comparison algorithm must choose a minimum seed length that balances sensitivity to weak similarities against efficiency on long sequences. High sensitivity favors a shorter seed length because similarities are more likely to contain shorter exact matches. In practice, programs like BLAST achieve adequate sensitivity to functional genome features using exact matches of 10–15 bases, though finding weak homologies such as ancient interspersed repeats may require exact matches as short as seven bases (Smit and Green, 1999). Unfortunately, such short exact matches often arise purely by chance in the background sequence, even in the absence of significant conservation. Exact matches therefore exhibit poor specificity to the alignments of interest, so that comparison algorithms must filter out a majority of spurious seeds to find the minority that extend into significant similarities. In comparisons between multimegabase genomic sequences, the number

of chance matches can range from tens of thousands for fifteen-base seeds up to hundreds of millions for ten-base seeds.

This work describes LSH-ALL-PAIRS, a new, scalable similarity search algorithm whose seeds are ungapped alignments with up to a specified fraction of substitutions. These seeds, unlike exact matches, can span substitutions; thus, the algorithm can use a much longer seed length (typically 60–80 bases) to improve its specificity, and therefore its efficiency, while maintaining sensitivity to weak similarities. The algorithm is particularly useful for finding similarities with frequent substitutions, such as orthologous exons with many changes in third codon positions. We have used our implementation of LSH-ALL-PAIRS to compare genomic sequences of more than ten megabases within a few hours, detecting ungapped similarities tens or hundreds of bases in length with as little as 63% nucleotide identity.

LSH-ALL-PAIRS finds its seed alignments by *locality-sensitive hashing* (Indyk and Motwani, 1998), an efficient randomized search technique. Because the algorithm is randomized, it is *incomplete*; that is, it may by chance miss some seeds, and therefore some significant similarities, in the input. Because every seed is found with nonzero probability, we can reduce the chance of missing any given seed by the standard approach of iterating the algorithm multiple times (Motwani and Raghavan, 1995, Section 1.5). The iterative strategy, along with some simple implementation heuristics, ensures that the number of missed similarities due to randomization can be made arbitrarily small.

Previous work in approximate string matching has produced deterministic, complete algorithms for finding ungapped alignments, such as multiple filtration (Pevzner and Waterman, 1995). Deterministic algorithms can efficiently detect ungapped alignments containing few substitutions (substantially fewer than $d/\log_4 d$ for alignments of length d (Gusfield, 1997, Section 12.3)); however, for the range of ungapped alignment lengths that are useful as seeds, the algorithms’ performance degrades rapidly as the allowed substitution rate rises to 25–40%. The randomized approach of LSH-ALL-PAIRS is substantially more efficient at such high substitution rates, even when our algorithm is iterated tens or hundreds of times to minimize the number of missed similarities.

In the remainder of this work, we describe the theory and practice of LSH-ALL-PAIRS. We first formally describe and analyze the algorithm, then discuss how to implement it efficiently. Finally, we give evidence of our method’s efficiency and sensitivity by finding conserved features in several genomic sequences from human and mouse, including the β -globin locus control region, the

T-cell receptor α/δ locus, and human chromosome 22.

Algorithm

The LSH-ALL-PAIRS algorithm uses locality-sensitive hashing (LSH) to reduce the problem of string matching with substitutions to a more tractable exact matching problem. LSH was developed by Indyk and Motwani (Indyk and Motwani, 1998) and later refined by Gionis and coworkers (Gionis *et al.*, 1999) for solving high-dimensional computational geometry problems such as finding nearest neighbors. In this section, we review the principle of LSH, describe how to apply it to find local alignments between sequences, and finally analyze its expected sensitivity and specificity using an independent, identically distributed (i.i.d.) background sequence model.

Locality-Sensitive Hashing

Consider two strings s_1 and s_2 of common length d over an alphabet Σ . Fix $r < d$; we call s_1 and s_2 *similar* if they differ by at most r single-character substitutions.

To detect similarity between s_1 and s_2 , we construct the following randomized filter. Choose k indices i_1, \dots, i_k uniformly at random from the set $\{1, \dots, d\}$; for ease of analysis, assume that the indices are sampled with replacement, so that an index can be chosen multiple times. Define the function $f : \Sigma^d \rightarrow \Sigma^k$ by

$$f(s) = \langle s[i_1], s[i_2], \dots, s[i_k] \rangle$$

The function f is called a *locality-sensitive hash function*. It concatenates characters from at most k distinct positions of s to form a length- k string called an *LSH value*. Our filter accepts the pair (s_1, s_2) if and only if $f(s_1) = f(s_2)$.

Indyk and Motwani call f “locality-sensitive” because the probability that two strings hash to the same LSH value under f varies directly with their similarity. More precisely, if s_1 and s_2 are similar,

$$\Pr[f(s_1) = f(s_2)] \geq \left(1 - \frac{r}{d}\right)^k \quad (1)$$

where the probability is computed over all random choices of i_1, \dots, i_k for f . Our filter is therefore likely to accept pairs of strings that differ by only a few substitutions while rejecting pairs that differ by many substitutions. Pairs that match exactly are always accepted.

The filter can commit both false positive and false negative errors. A false positive occurs if strings s_1 and s_2 are dissimilar but $f(s_1) = f(s_2)$, i.e. if f samples only positions at which the two strings agree. False positives can easily be distinguished from true positives by counting the number of positions at which s_1 and s_2 disagree.

Conversely, a false negative occurs if s_1 and s_2 are similar but $f(s_1) \neq f(s_2)$, i.e. if f samples any position at which the strings disagree. We cannot detect false negatives efficiently, but we can make them arbitrarily unlikely by repeated filtering with multiple independent random hash functions.

LSH for All-Pairs Local Alignment

We use LSH to compute an approximate solution to the following local alignment problem:

Given a collection C of sequences with total length N , find all pairs of d -mers (optionally from distinct sequences) in C that differ by no more than r substitutions.

When $r = 0$, this problem reduces to the well-studied task of finding all pairs of exactly matching d -mers.

The LSH-ALL-PAIRS algorithm iterates the following four steps:

1. Choose a random locality-sensitive hash function f by picking k indices i_1, \dots, i_k from $\{1, \dots, d\}$.
2. For every d -mer s of every sequence in C , store a tuple $\langle f(s), \pi(s) \rangle$, where $\pi(s)$ is the position (sequence number and starting offset) of s in C .
3. Partition the tuples into classes C_j , such that all tuples in a class have the same LSH value $f(s)$.
4. For each class C_j , compare all pairs of d -mers whose tuples are in C_j . Store the positions of those pairs that differ by at most r substitutions.

Steps 1–4 are iterated ℓ times, after which LSH-ALL-PAIRS outputs the union of the sets of similar pairs found in each iteration. The constants ℓ and k are fixed at the beginning of the algorithm based on the parameters d and r and on efficiency and sensitivity considerations described in the next section.

LSH-ALL-PAIRS is sound because it outputs only similar pairs of d -mers. However, the algorithm is only guaranteed to find all pairs that match exactly; it will miss similar pairs that happen to be false negatives for every one of the ℓ hash functions chosen. The number of missed pairs can be controlled by performing more iterations or by allowing more d -mers to hash together in each iteration.

The running time for each iteration of LSH-ALL-PAIRS is dominated by two factors: the cost of creating and partitioning tuples for each d -mer in C , and the cost of comparing the d -mer pairs in each class C_j . Creating the tuples clearly takes time $\Theta(kN)$, but we can also partition them in time $\Theta(kN)$ by a variety of methods.

For example, we can partition tuples by ordinary (non-locality-sensitive) hashing keyed on their LSH values or, for small alphabets like $\{A, C, G, T\}$, by radix sorting by LSH value.

The time spent comparing d -mer pairs depends critically on the sizes of the classes C_j . In the worst case, $\Theta(N)$ d -mers might hash to the same LSH value, either because they are all pairwise similar or because the hash functions f yield many false positives. The number of string comparisons performed can therefore in theory be as large as $\Theta(\ell N^2)$. For practical sequence comparisons, however, we can usually choose ℓ and k to make the C_j 's small – at most some tens of d -mers each – so that the number of string comparisons performed is orders of magnitude fewer than N^2 . The next section characterizes the expected number of comparisons more precisely in terms of ℓ and k .

Performance Analysis

We now analyze the sensitivity and specificity of LSH-ALL-PAIRS. Sensitivity is expressed by the false negative rate ρ_{fn} , which measures the fraction of all similar pairs missed by the algorithm. Specificity is expressed by the false positive rate ρ_{fp} , which measures the fraction of all *dissimilar* pairs that the algorithm must check and reject. In the usual case of sequences of total length N with $\Theta(N^2)$ dissimilar pairs, the time spent rejecting false positives is proportional to $\rho_{fp}dN^2$.

The performance of LSH-ALL-PAIRS depends on the number of iterations ℓ and the number of positions k sampled per iteration. Intuitively, increasing k increases specificity (and so reduces running time) because dissimilar d -mers become less likely to hash to the same LSH value. However, increasing k also decreases sensitivity because *similar* d -mers also hash together less often. We can boost sensitivity by increasing ℓ , giving each similar pair more opportunities to hash together, but this in turn gives each dissimilar pair more chances to hash together as well. The following analysis makes these tradeoffs precise.

We first compute the expected false negative rate ρ_{fn} . This rate is typically specified by the user as a parameter of the algorithm; for example, setting $\rho_{fn} = 0.05$ would on average recover at least 95% of all similar pairs of d -mers. Let s_1 and s_2 be one such similar pair. As shown in Inequality (1), a single randomly chosen hash function sampling k positions will hash s_1 and s_2 together with probability at least $(1 - r/d)^k$. Hence, the probability q_{fn} that s_1 and s_2 *never* hash together under any of ℓ independent hash functions is bounded by

$$q_{fn} \leq \left[1 - \left(1 - \frac{r}{d}\right)^k\right]^\ell \quad (2)$$

q_{fn} is computed for one similar d -mer pair, but by linearity of expectation, $q_{fn} = \rho_{fn}$, the expected fraction of *all* similar pairs missed by the algorithm after ℓ iterations. Note that Inequality (2) holds with equality only for pairs of d -mers that differ by exactly r substitutions; pairs differing by fewer substitutions are *more* likely to be detected.

Rearranging Inequality (2) to isolate k , we find that

$$k \leq \frac{\log\left(1 - \rho_{fn}^{1/\ell}\right)}{\log\left(1 - \frac{r}{d}\right)} \quad (3)$$

This bound determines the largest k that can be used without compromising our desired sensitivity. For a given sensitivity, we can trade an increase in k , which lowers the expected work per iteration, for a larger number of iterations ℓ . The false positive rate ρ_{fp} determines when this tradeoff is advantageous.

To estimate our algorithm's expected false positive rate, we must assume a specific background sequence model. We assume an i.i.d. model, in which characters are chosen independently at random for each sequence position such that any two positions match with probability ϕ . In practice, ϕ ranges between 0.25 and 0.30. In this model, the chance that two random, independently-chosen d -mers differ by exactly t substitutions is the binomial probability

$$\beta_{1-\phi,d}[t] = \binom{d}{t} (1 - \phi)^t \phi^{d-t}$$

We will compute the probability q_{fp} that two randomly chosen *dissimilar* d -mers hash together in one iteration of the algorithm. If the d -mers differ by exactly t substitutions, they hash together with probability $(1 - t/d)^k$. Applying the prior distribution on t induced by our sequence model and summing over all t large enough to be considered dissimilar, we have that

$$q_{fp} = \sum_{t=r+1}^d \beta_{1-\phi,d}[t] \left(1 - \frac{t}{d}\right)^k \quad (4)$$

The probability q_{fp} is also the expected rate of false positives in one iteration of the algorithm; hence, the overall false positive rate ρ_{fp} is $\ell \cdot q_{fp}$.

For fixed ℓ and ρ_{fn} , we can minimize the false positive rate by choosing k as large as possible, subject to the upper bound of Inequality (3) and the fact that k must be an integer. Figure 1 illustrates how ρ_{fp} depends on ℓ for typical values of d , r , and ρ_{fn} . As ℓ becomes larger, the maximum allowable k increases, which lets us lower ρ_{fp} . The decreases in ρ_{fp} are discontinuous because k grows in integer steps; between these steps, the false positive

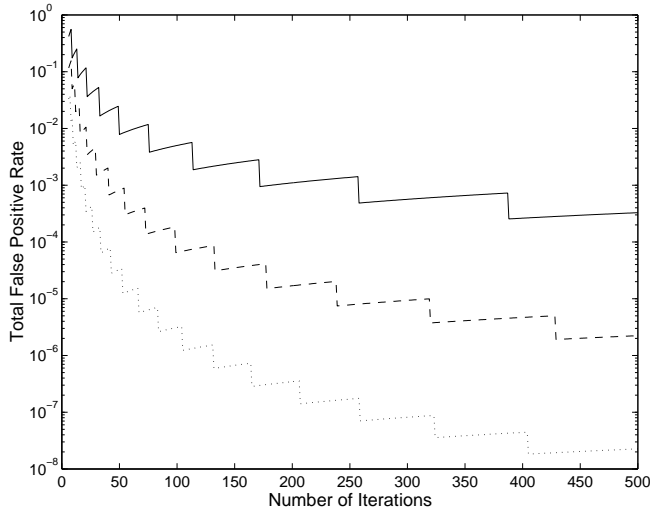


Figure 1: false positive rates as a function of ℓ for $d = 75$, $\rho_{fn} = 0.05$, and various levels of similarity. From top to bottom, the lines represent $r = 25$ (67% identity), $r = 19$ (75% identity), and $r = 15$ (80% identity). For each value of ℓ , we used the largest possible k as determined by Inequality (3) to compute the false positive rate.

rate actually increases because we perform more iterations without changing k .

In summary, the LSH-ALL-PAIRS algorithm may be parameterized as follows. The user specifies a desired level of match identity r/d and a target false negative rate ρ_{fn} . We first fix d long enough to achieve a low rate of chance matches under the background sequence model. We then compute a curve for ρ_{fp} versus ℓ as shown in Figure 1 using d , r , and ρ_{fn} . Finally, we choose the number of iterations ℓ , which determines a point on the curve and therefore fixes both k and the expected false positive rate.

The choice of ℓ should minimize the total running time $\Theta(\ell k N) + \Theta(\rho_{fp} d N^2)$. Because ρ_{fp} decreases with increasing k , which in turn depends on ℓ , we must trade off the time spent discarding false positives against the time to partition $\Theta(N)$ tuples in each iteration. To choose ℓ optimally, we scale the two terms in the running time by their respective implementation constants, which are the costs for one partitioning step and one pairwise d -mer comparison. We measure these costs empirically. Total work is typically minimized when ℓ is large, on the order of several hundred iterations for match identities of 60 to 70%.

Our parameter estimates conservatively assume that we can find similar d -mer pairs only if the core LSH-ALL-PAIRS algorithm hashes them together. However, we may relax this assumption by adding inexpensive algorithmic extensions to recover similarities that the core algorithm

would otherwise miss. We consider several such extensions, along with other implementation details, in the next section. In practice, extending the core algorithm allows us to perform substantially fewer iterations while still achieving a low false negative rate.

Implementation

We have implemented the LSH-ALL-PAIRS algorithm in the C++ language as part of a program to compute local alignments between long genomic sequences. Our implementation focuses on memory efficiency, which is necessary to analyze very long sequences, and on heuristic extensions to reduce the probability of missing similarities even when the core algorithm has a high false negative rate.

Partitioning Tuples

We partition the array of tuples $\langle f(s), \pi(s) \rangle$ for the input sequences using an in-place quicksort on the tuples' LSH values. Although quicksort requires time $\Omega(N \log N)$ rather than the optimal time $O(N)$, it allows us to sort twice as many tuples in memory, and therefore to analyze sequences twice as long, as would be possible using an $O(N)$ sort that requires a temporary array. After sorting, the classes C_j correspond to contiguous runs of tuples with the same LSH value.

Our sorting-based partitioning scheme differs from that used by Gionis *et al.* in their geometric application of LSH. They chose to build a hash table of tuples keyed on LSH value, primarily because their application required constant-time access to arbitrary C_j 's as new tuples became available online. In contrast, LSH-ALL-PAIRS has all d -mers available at the start of computation, so the online property is not required. We prefer sorting over hashing because, although slightly slower, it is more memory-efficient. Our implementation requires 8–12 bytes per base, depending on how compactly sequence positions are stored, while even a highly optimized hashing implementation requires 16–20 bytes per base.

The sorting approach to partitioning extends naturally to the case when the array of tuples to be sorted does not fit in main memory. We can partition such large arrays efficiently using a merge sort to disk, which accesses disk blocks sequentially and so minimizes the number of expensive seek operations. In contrast, partitioning by disk-based hashing requires essentially random access to the disk, which can be quite slow.

Finding and Extending Similar Pairs

Step four of the LSH-ALL-PAIRS algorithm compares every pair of d -mers with the same LSH value to find those

pairs that are actually similar. We reduce memory usage in this phase of the computation by storing only a representative subset of all similar d -mer pairs found. Specifically, we require that each pair of d -mers stored begin with a residue match and that it be preceded by a residue mismatch. We call d -mer pairs that meet these criteria *canonical*. Given any d -mer pair, we can find a nearby canonical pair on the same diagonal using the following rule: if the pair begins with a residue match, search backwards to find a canonical pair; otherwise, search forwards¹. Typically, the search terminates after inspecting only a few characters, so it adds little to the comparison cost.

Conversion to canonical pairs provides an opportunity to increase our algorithm's sensitivity: instead of converting only similar d -mer pairs, we convert every pair *before* it is checked for similarity. It is not hard to see that shifting the starting indices of a d -mer pair according to our search rule can never reduce and may increase the number of matching residues it contains. Converting to canonical pairs before comparison therefore detects some similar d -mer pairs that would otherwise be missed.

After LSH-ALL-PAIRS completes, we are left with a list of seeds (the canonical similar d -mer pairs) that must be extended into local alignments suitable for reporting to the user. We first expand the list of seeds by searching the region around each canonical pair (currently 500 bases to 5' and 3') to find nearby similar pairs on the same diagonal. This expansion step recovers noncanonical pairs that were not stored and may also discover similar pairs that were missed by the core algorithm. We then assemble overlapping seeds on the same diagonal into longer, disjoint ungapped local alignments. Our initial implementation of seed extension does not attempt to further extend seeds into gapped alignments, but we intend to implement gapped extension in the future.

Before we report a local alignment to the user, we verify that it is sufficiently well conserved so as to be unlikely to have arisen by chance alone. We estimate the significance of each local alignment using the statistical theory of Karlin and Altschul (Karlin and Altschul, 1990) for ungapped high-scoring segment pairs (HSP's). To apply the theory, we trim each local alignment to remove regions of low similarity at its ends, keeping only the central HSP that maximizes the score ($\# \text{ matches} - \# \text{ mismatches}$). We report only HSP's whose scores are significant ($p < 0.05$) given the background nucleotide frequencies of the sequences being compared.

Our three extensions to the core LSH-ALL-PAIRS algorithm – conversion to canonical pairs, seed list expansion,

and seed assembly – each improve the algorithm's ability to find significant local alignments for any fixed values of ℓ and k . Each canonical pair can be found by checking any one of several nearby d -mer pairs; similarly, many d -mer pairs can be recovered by seed list expansion if the core algorithm discovers at least one nearby canonical pair. The same local alignment can often be assembled from multiple sets of d -mer pairs, so assembly provides additional redundancy in recovering alignments. As our results will show, these extensions can uncover significant similarities between sequences much faster than would be predicted from the recovery rate for d -mer pairs alone.

Results

We tested the sensitivity and efficiency of LSH-ALL-PAIRS by analyzing genomic sequences from human and mouse in an attempt to recover known homologies. The sequences tested ranged in length from 10 kilobases to 17 megabases and were searched for similarities with substitution rates of up to 33–50%. Details of our testing procedure and parameter values for each experiment are described in the Appendix. We found that LSH-ALL-PAIRS efficiently recovered known homologies between our test sequences, including many similarities with less than 70% nucleotide identity and some that contained only very short exact matches.

The sequences and alignments described in this section are also available on the author's supplementary web site at <http://bio.cs.washington.edu/jbuhler-bioinformatics-2001/>.

β -Globin Locus Control Region

To test the performance of LSH-ALL-PAIRS on distantly related noncoding nucleotide sequences, we compared the locus control regions (LCRs) of the human and mouse β -globin loci². The LCRs contain no genes but do show significant conservation over short segments, called *DNAse-I hypersensitive sites*, which are known to regulate transcription at the locus (Fraser and Grosfeld, 1998). The regions between hypersensitive sites in mouse are less conserved versus human – particularly so in comparison to other mammals such as goat and rabbit – but do contain some weak similarities.

The human and mouse LCR sequences were respectively 20 and 12 kilobases in length and contained two conserved hypersensitive sites. We searched for ungapped similarities with up to 50% mismatches, which required roughly seventeen seconds.

Figure 2 shows the results of our search. LSH-ALL-PAIRS yielded eleven significant similarities with a median length of 82 bases and a median identity of 67.1%.

¹We must still accept similar but noncanonical pairs in which one d -mer occurs at the end of a sequence.

²GenBank accessions: U01317 (human), Z13985 (mouse); only the first 20 kb of human sequence were used.

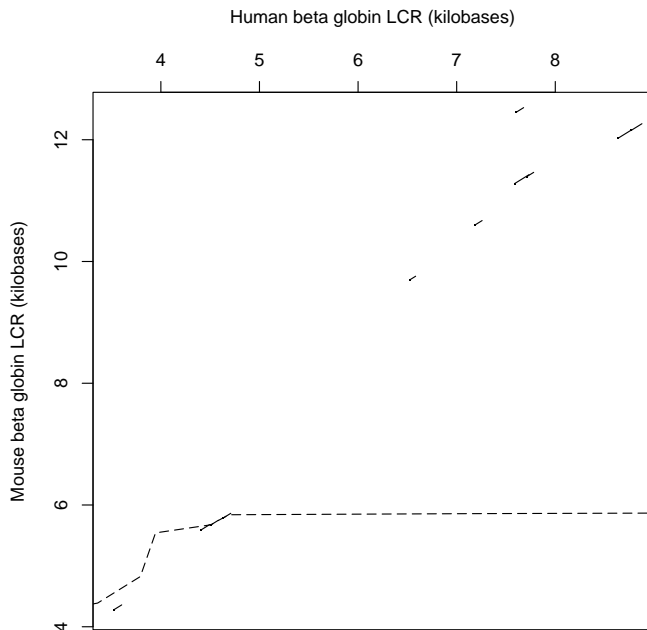


Figure 2: comparison of β -globin locus control regions in human and mouse. The solid lines are ungapped similarities detected by LSH-ALL-PAIRS, with a median identity of 67.1%. The dashed line is an alignment path proposed by MUMmer when run with a match length of ten bases.

These similarities cover the two hypersensitive sites, located at approximately 4.5 kb and 8.6 kb in the human sequence, as well as several regions in between. Similarities at the hypersensitive sites were 70–80% identical, while those found between them were 63–67% identical. All similarities were detected within the algorithm’s first ten iterations (out of 382 total).

Although LSH-ALL-PAIRS easily identified known homologies between the human and mouse LCR’s, these sequences appear difficult for alignment algorithms that rely on long exact matches. Eight of the eleven similarities we found contained no exact match longer than ten bases, yet we observed over 160 exact matches of 10–13 bases between the two LCR sequences, most of which are expected to be random noise. We tried to separate matches near the hypersensitive sites from the noise by globally aligning the two sequences with TIGR’s MUMmer software (Delcher *et al.*, 1999), using exact matches of length at least ten to construct an alignment path. Although MUMmer uses several techniques to separate chance matches from those on the correct alignment path, in this case it chose a path, also shown in Figure 2, which missed the known hypersensitive site at 8.6 kb. MUMmer successfully found both hypersensitive sites when we increased the minimum exact match length to eleven, but this length is too long to detect most of

the other observed similarities.

T-cell Receptor α/δ Locus

The T-cell receptor (TCR) α/δ locus is one of several loci that determine the shapes of antigen-specific T-cell receptors in the vertebrate immune system. This locus contains two families of paralogous features – *V-segments* and *J-segments* – each of which codes for part of the receptor protein (Wilson *et al.*, 1988). Sequence divergence among V- and J-segments is a key source of diversity between lineages of antigen-specific T-cells: each lineage’s progenitor cell edits its genome to create a new TCR gene containing a randomly chosen V- and J-segment, along with a fixed constant or C-segment.

To test the performance of LSH-ALL-PAIRS in identifying diverged coding sequences, we searched for similarities between gene segments in the human and mouse TCR α/δ loci³. The human and mouse sequences spanned 1.07 Mb and 1.67 Mb respectively, of which 587 kb in human and 774 kb in mouse remained after removing common interspersed repeats. Searching these sequences for ungapped similarities with at least 67% identity required about 16 minutes.

Our analysis produced 2879 significant similarities, shown in Figure 3, with a median length of 93 bases (compared to the 300–500 bases in each V-segment) and a median identity of 72%. The median exact match length found in these similarities was twelve bases, though more than a quarter had no exact match longer than eight bases. Similarity between paralogous V- and J-segments caused each segment from one sequence to match multiple segments in the other, giving rise to the large number of off-diagonal matches present in the figure. Over 99.5% of similarities were found within the algorithm’s first thirty iterations (out of 258 total).

We evaluated the sensitivity of our analysis by comparing the similarities found to the annotated positions of V-segments in the two sequences⁴. LSH-ALL-PAIRS’ matches intersected 56 of 57 annotated V-segments in human and 104 of 106 V-segments in mouse, primarily by matching the 3’ ends of the segments’ second exons along with a conserved noncoding signal just downstream of these exons. Our analysis also detected similarities involving roughly 90% of annotated J-segments in each sequence, as well as additional features including the four exons of each sequence’s C-segment, several noncoding

³GenBank accessions for human sequence (Boysen *et al.*, 1997): AE000658–62. Accessions for mouse sequence: AC003057, AC003993–7, AC004096, AC004101–2, AC004399, AC004404–7, AC005240–1, AC005402–3, AC005835, AC005855, AC005938, AC005964, AC006119, AF259071–4, M64239.

⁴Annotations were kindly provided by C. Boysen for human (Boysen, 2000) and J. Roach for mouse (Roach, 2000).

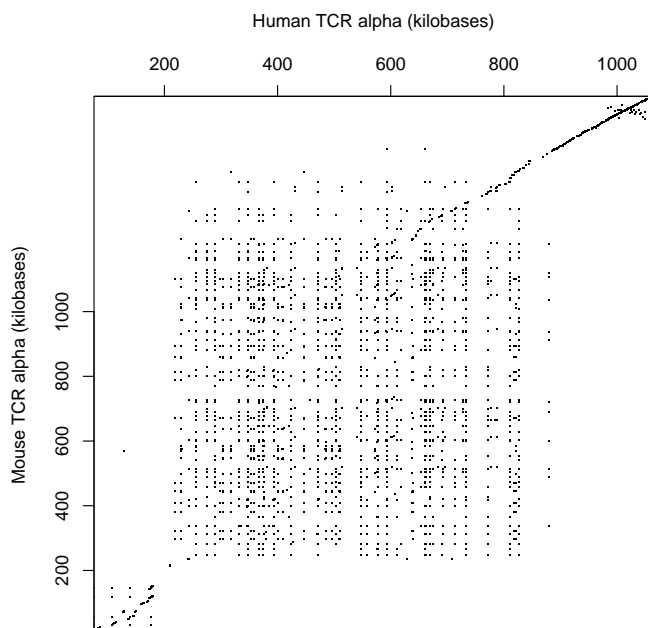


Figure 3: similarities detected between the human and mouse TCR α/δ loci. The median identity of all similarities shown is 72%. The central group of similarities are V-segments, while the smaller group at upper right are J-segments.

enhancer sequences, and a cluster of olfactory receptor genes near the 5' ends of the sequences.

We detected several matches between annotated V-segments in one genome and regions with no annotation in the other. These matches appear to reflect additional V-segments, probably no longer transcribed, that were not detected by the investigators who originally annotated these loci.

Human Chromosome 22

To test the performance of LSH-ALL-PAIRS on a large sequence, we searched for significant self-similarities in the sequence of human chromosome 22. We used the Sanger Center's May 2000 assembly of the chromosome, though we obtained similar results with the original assembly described in (Dunham *et al.*, 1999).

Human chromosome 22 has 34.6 megabases of euchromatic sequence, of which 18 megabases remained after masking interspersed repeats. Searching for ungapped self-similarities with at least 67% identity required 15.2 hours and roughly 285 megabytes of memory.

Our analysis produced 38514 significant similarities, plotted in Figure 4. The similarities had a median length of 111 bases, a median identity of 79.6%, and a median exact match length of eighteen bases. 6274 similarities, or approximately 16% of those reported, had identities

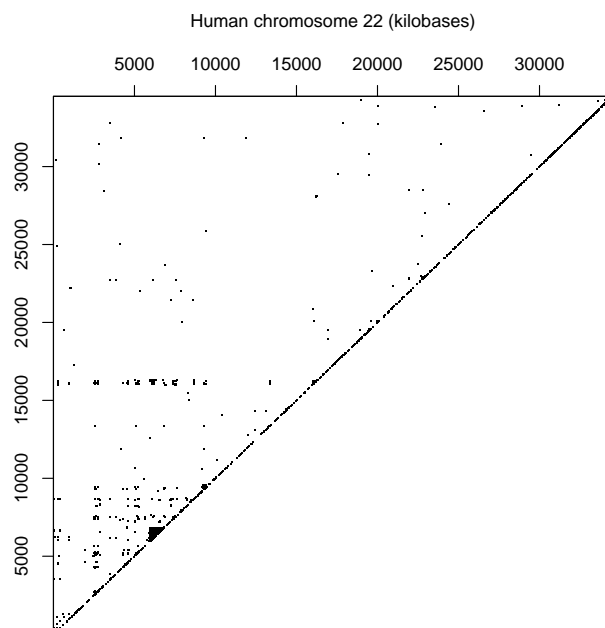


Figure 4: similarities detected in the self-alignment of chromosome 22. The median identity of all similarities shown is 79.6%. Compare Figure 3 of (Dunham *et al.*, 1999).

of at most 70%; among these, the median exact match length was only ten bases. Over 3×10^8 exact 10-mer matches would be expected in chromosome 22 by chance alone, so we expect that many of the weak similarities we found would be difficult to detect efficiently by conventional methods that filter exact matches.

LSH-ALL-PAIRS was able to identify a number of repetitive gene families on chromosome 22. The most striking group of similarities was the large immunoglobulin λ locus, located between 5.9 and 6.9 Mb in the figure, which has a family of paralogous gene segments similar to the TCR loci. Close inspection of our plot revealed additional clusters of similarities near the diagonal; using the annotated dot plot from (Dunham *et al.*, 1999), we identified these clusters as the glutathione *S*-transferase, β -crystallin, Ret-finger-protein-like, apolipoprotein, and APOBEC gene families. Vertical and horizontal bands in the figure correspond to LCR-22 chromosome-specific repeats. We did not report matches on the main diagonal, so the central line apparent in the figure actually consists of approximate tandem repeats and self-overlapping matches on nearby diagonals.

Over 99% of all similarities we found were detected after only fifty iterations (out of 583 total), or slightly more than one hour of computing time. The quick recovery of most of these similarities can be explained by the fact that their component *d*-mer pairs had consider-

ably higher identity than the 67% used to compute the false negative rate ρ_{fn} . However, even the weakest similarities – those with at most 70% identity – were over 99% complete after only fifty iterations.

Table 1 illustrates the effects of our enhancements to LSH-ALL-PAIRS on the recovery rates for weak similarities. Our analysis used a match length of $d = 81$ bases. Inequality (2) predicts that of all 81-mer pairs with at most 70% identity found in 583 iterations, fewer than half should be detected after only fifty iterations. However, canonical pairs and extended similarities were recovered much faster because they could be detected by checking any one of several 81-mer pairs. As a result, the analysis was substantially complete in less than one tenth the time required for all 583 iterations.

As the table shows, Karlin-Altschul significance testing unexpectedly contributed to the high recovery rate for distant similarities. Some similarities that were found only after our algorithm’s first fifty iterations were never reported at all because they were not long or highly-conserved enough to be considered significant. It is debatable whether this fact constitutes a feature of our method because at least some of the discarded similarities were likely parts of longer gapped matches that we could not detect. Even if we discount the effect of significance testing, the number of assembled similarities reported after only fifty iterations is still a respectable 95.6% of the total.

Discussion

We have demonstrated the ability of LSH-ALL-PAIRS to detect known homologies efficiently in several biosequences. The algorithm proved capable of detecting significant similarities with as little as 63% identity in both coding and noncoding sequences. It efficiently found weak ungapped similarities without discarding large numbers of seeds, even when the similarities found contained no long exact matches. We have confirmed these results in a number of other analyses, including whole-genome comparisons among three *Pyrococcus* species and additional orthologous regions from human and mouse. We plan to use LSH-ALL-PAIRS for comparative analyses of the very large genome fragments that are now or will soon become available from organisms such as human, mouse, *C. elegans*, *Drosophila*, and *Arabidopsis*.

The evidence strongly suggests that our enhancements to the core LSH-ALL-PAIRS algorithm – canonical d -mer pairs, seed list expansion, and seed assembly – substantially improve its recovery rate for weak similarities over the pessimistic rate predicted for raw d -mer pairs. In our chromosome 22 analysis, taking full advantage of these enhancements when choosing ℓ and k could have avoided

over 90% of the computation at a minimal cost in missed similarities; however, it was not clear *a priori* how small we could safely make ℓ . Future work on LSH-ALL-PAIRS must include more careful empirical techniques for estimating safe yet efficient parameters for large-scale sequence comparisons.

Improvements to LSH-ALL-PAIRS will circumvent some of its current limitations and expand its utility to more annotation problems. Four important areas in which the algorithm can be enhanced are detecting short similarities, finding similarities with gaps, use of more general scoring functions, and extension to multiple alignment.

Short, highly conserved similarities were problematic in our experiments because we chose d -mer lengths long enough to avoid processing many chance matches. For example, our analysis of chromosome 22 used seed alignments of length 81 with at most 27 mismatches; it therefore failed to detect perfect matches of length up to 53 if they were surrounded by regions with no matching residues.

To detect shorter regions of high conservation, we may reduce d and, like existing algorithms, rely on seed extension to filter out the majority of chance seed alignments. However, this simple, computationally intensive approach does not take advantage of the fact that shorter similarities must be more highly conserved to be considered significant. A more efficient solution would perform multiple searches, gradually decreasing d while simultaneously increasing the similarity threshold to keep the number of false positives as small as possible.

Gapped similarities are difficult for LSH-ALL-PAIRS for two reasons. First, the segments between gaps may be so short as to be missed by the initial search for ungapped d -mer pairs. If each ungapped segment is well conserved, we can use the above strategy for reducing d -mer length to find all such segments. Second, our seed extension phase does not yet attempt to link nearby similarities on different diagonals. As a result, long gapped similarities may be broken into ungapped fragments that by themselves are not considered significant. Future work will use a gap-aware method such as banded Smith-Waterman or a fragment assembly algorithm (Eppstein *et al.*, 1992) to perform seed extension.

LSH-ALL-PAIRS can be generalized to use more biologically sophisticated similarity measures. In particular, we wish to extend the core locality-sensitive hashing technique to find ungapped alignments that score highly under more general alignment scoring functions than a simple count of mismatches. LSH has previously been generalized to work with Euclidean and Manhattan metric distances (Gionis *et al.*, 1999), and we have extended it to simple alignment scoring functions such as the DNA-PAM-TT family (States *et al.*, 1991). A more ambitious

Table 1: Features With At Most 70% Identity Found in Chromosome 22

Feature Type	# after 50 Iters	# after 583 Iters	Fraction after 50 Iters
81-mer pairs	–	–	≈ 0.4 (predicted)
Canonical 81-mer pairs	82207	128470	0.640
Assembled Similarities	8267	8648	0.956
Significant Similarities	6249	6274	0.996

extension to arbitrary scoring functions will, for example, allow our algorithm to process AT-rich genomes more efficiently by hashing apart d -mers with many A and T matches while hashing together those with many G and C matches.

Finally, it would be useful to extend LSH-ALL-PAIRS to find multiple alignments – similar substrings occurring simultaneously in three or more sequences. Locating orthologous regions in several widely diverged genomes increases the likelihood that the regions have functional importance, while multiple alignment within a single genome can discover shorter conserved elements that would not be significant if only two copies were known. The ideas of LSH-ALL-PAIRS extend straightforwardly to finding multiple matches: if a sequence occurs several times, a random locality-sensitive hash function is likely to hash *all* of its copies together. We have successfully applied this idea to improve the sensitivity of motif-finding algorithms (Buhler and Tompa, 2001). Future work will determine if the same idea produces an efficient and sensitive algorithm for the much longer sequences and larger feature sizes of multiple genome alignment problems.

Acknowledgements

The author wishes to thank Cecilie Boysen, Inyoul Lee, and Jared Roach for providing TCR assemblies and annotations, and Piotr Indyk, Max Robinson, and Martin Tompa for helpful discussions. This work was supported in part by a Fannie and John Hertz Fellowship, and in part by National Science Foundation grant DBI-9974498.

Appendix: Details of Experimental Alignments

We performed our tests on a 550 MHz Intel Pentium III computer running the Linux operating system. All computations performed fit in the machine’s memory without swapping. Sequences were preprocessed with RepeatMasker (Smit and Green, 1999) to remove interspersed repeats, which would otherwise have comprised the majority of all detected similarities. Only similarities occurring on the forward strand of each sequence were computed.

We searched for ungapped seed alignments with minimal identities of 50 to 67%. We chose the match length d in each test long enough that the number of chance seeds was small.

More specifically, we chose d such that the expected number of chance occurrences of d -mer pairs with identity at least $1 - r/d$ was no more than two. The expectations were computed using i.i.d. random sequence models whose nucleotide frequencies matched the sequences being compared.

We conservatively parameterized all experiments to obtain an expected false negative rate $\rho_{fn} \leq 0.05$ for similar d -mer pairs. To choose among the numerous values of ℓ and k , we performed empirical running time measurements as described above, which, along with our theoretical performance analysis, yielded ℓ and k that minimized each experiment’s expected running time.

Specific parameter choices for each experiment were as follows. *β -globin*: $d = 130$, $r = 65$, $k = 7$, $\ell = 382$; *TCR*: $d = 69$, $r = 23$, $k = 11$, $\ell = 258$; *chromosome 22*: $d = 81$, $r = 27$, $k = 13$, $\ell = 583$.

References

- Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acids Res.*, **25**, 3389–402.
- Batzoglou, S., Pachter, L., Mesirov, J. P., Berger, B. and Lander, E. S. (2000) Human and mouse gene structure: comparative analysis and application to exon prediction. *Genome Res.*, 950–8.
- Boysen, C. (2000) Personal communication.
- Boysen, C., Simon, M. and Hood, L. (1997) Analysis of the 1.1-Mb human alpha/delta T-cell receptor locus with bacterial artificial chromosome clones. *Genome Res.*, **7**, 330–8.
- Buhler, J. and Tompa, M. (2001) Finding motifs using random projections. In *RECOMB 01: Proc. 5th Ann. Intl. Conf. on Computational Mol. Bio.*, Montreal, CA.
- Delcher, A. L., Kasif, S., Fleischmann, R. D., Peterson, J., White, O. and Salzberg, S. L. (1999) Alignment of whole genomes. *Nucl. Acids Res.*, **27**, 2369–76.
- Dunham, I., Shimizu, N., Roe, B. A., Chisoe, S. *et al.* (1999) The DNA sequence of human chromosome 22. *Nature*, **402**, 489–95.
- Eppstein, D., Galil, Z., Giancarlo, R. and Italiano, G. F. (1992) Sparse dynamic programming I: Linear cost functions. *J. ACM*, **39**, 519–45.
- Fraser, P. and Grosveld, F. (1998) Locus control regions, chromatin activation, and transcription. *Curr. Op. Cell Biol.*, **10**, 361–5.

- Gionis, A., Indyk, P. and Motwani, R. (1999) Similarity search in high dimensions via hashing. In *Proc. 25th Intl. Conf. on Very Large Databases*, Edinburgh, Scotland.
- Green, P. (1994) Crossmatch. At <http://bozeman.mbt.washington.edu/phrap.docs/swat.html>.
- Gusfield, D. (1997) *Algorithms on strings, trees, and sequences*. Cambridge University Press, New York, NY.
- 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**, 959–66.
- Hattori, M., Fujiyama, A., Taylor, T. D., Watanabe, H. *et al.* (2000) The DNA sequence of human chromosome 21. *Nature*, **405**, 311–9.
- Huang, X. and Miller, W. (1991) A time-efficient, linear-space local similarity algorithm. *Adv. Appl. Math.*, **12**, 337–57.
- Indyk, P. and Motwani, R. (1998) Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Ann. ACM Symp. on Theory of Computing*, Dallas, TX.
- Karlin, S. and Altschul, S. F. (1990) Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci.*, **87**, 2264–8.
- Loots, G. G., Locksley, R. M., Blankespoor, C. M., Wang, Z. E., Miller, W., Rubin, E. M. and Frazer, K. A. (2000) Identification of a coordinate regulator of interleukins 4, 13, and 5 by cross-species sequence comparisons. *Science*, **288**, 136–40.
- Motwani, R. and Raghavan, P. (1995) *Randomized Algorithms*. Cambridge University Press, New York, NY.
- Pevzner, P. and Waterman, M. S. (1995) Multiple filtration and approximate pattern matching. *Algorithmica*, **13**, 135–54.
- Roach, J. (2000) Personal communication.
- Schwartz, S., Zhang, Z., Frazer, K. A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R. and Miller, W. (2000) PipMaker – a web server for aligning two genomic DNA sequences. *Genome Res.*, **10**, 577–86.
- Smit, A. and Green, P. (1999) Repeatmasker. At <http://ftp.genome.washington.edu/RM/RepeatMasker.html>.
- States, D. J., Gish, W. and Altschul, S. F. (1991) Improved sensitivity of nucleic acid database searches using application-specific scoring matrices. *Methods: a Companion to Methods in Enzymology*, **3**, 66–70.
- Wilson, R. K., Lai, E., Concannon, P., Barth, R. K. and Hood, L. E. (1988) Structure, organization, and polymorphism of murine and human T-cell receptor α and β chain gene families. *Immunol. Rev.*, **101**, 149–72.