In Proceedings, Fourth International Conference on Intelligent Systems for Molecular Biology, AAAI press, 1996.

Compact Encoding Strategies for DNA Sequence Similarity Search

David J. States and **Pankaj Agarwal**
Institute for Biomedical Computing
Washington University, Box 8036
700 S. Euclid, St. Louis, MO 63110
{states, agarwal}@ibc.wustl.edu

Abstract

Determining whether two DNA sequences are similar is an essential component of DNA sequence analysis. Dynamic programming is the algorithm of choice if computational time is not the most important consideration. Heuristic search tools, such as BLAST, are computationally more efficient, but they may miss some of the sequence similarities (Altschul et al., 1990). These tools often use common k-tuples (words) between the two sequences to determine anchor points for the alignment, and spend most of their computational time extending the alignment beyond these anchor points. We discuss and provide a DNA sequence similarity search implementation (called SENSEI) that improves upon the performance of BLASTN by almost an order of magnitude for comparable sensitivity. This improvement is a result of using compactly encoded scoring tables for k-tuples, encoding bases with a single bit, filtering the sequence to remove the simple sequence repeats using XNUN, and masking the known species-specific repeats in the query sequence. To reduce memory requirements, especially for large genomic DNA query sequences, we recommend generating the neighborhood words from the target sequence at run-time, instead of generating them by preprocessing the query sequence.

Introduction

Discovering homologs to a new DNA sequence is often the first step in establishing its possible origin and function. Statistically significant sequence similarity is often used to infer homology. Dynamic programming is the best known algorithm for establishing sequence similarity (Smith and Waterman, 1981; Needleman and Wunsch, 1970; Myers and Miller, 1988). Unfortunately, dynamic programming is computationally expensive, as it takes time proportional to the product of the lengths of the query sequence and the target sequence. There are numerous heuristic sequence similarity search programs, which are faster but perhaps not as sensitive. Most of these have been optimized for amino acid sequence comparisons and perform remarkably well. These programs include BLAST (Altschul et al., 1990) and FASTA (Pearson and Lipman, 1988). Though the heuristic techniques apply to nucleic acid sequence similarity searches, they have not been optimized for these searches.

Nucleic acid sequences have special properties that should make it possible to develop better tools for specifically searching nucleic acid sequence databases. These properties include a small four-nucleotide alphabet and simple evolutionary models for base mutations (States, Gish, and Altschul, 1991). Moreover, nucleotide query sequence are often much longer in length than protein sequences. The nucleotide sequences range from 40,000 bases for a cosmid to a few million bases for bacterial genomes and yeast chromosomes. Soon, contigs of almost a hundred million bases will become available as the *C. elegans* and human chromosomes are sequenced.

SENsitive **SE**arch **I**mplementation (SENSEI) is a tool for the computationally efficient identification of nucleic acid sequence similarities, and it is particularly optimized for the analysis of large sequences. The search engine is based on a heuristic word search similar to that of BLASTN (Altschul et al., 1990). BLASTN, a component of the BLAST suite of programs, is used for searching DNA query sequences against a DNA sequence database or a DNA target sequence. For comparable sensitivity, SENSEI is almost an order of magnitude faster than BLASTN. Several features have been incorporated to facilitate genomic sequence analysis, including automated masking of short and long period repeats, assembly of ungapped aligned segments (HSPs) into gapped alignments, a memory-efficient algorithm, and increased flexibility in scoring schemes.

Methods

The SENSEI search engine

Both the BLAST and SENSEI search engines are based on a word search algorithm in which words generated from the query sequence are indexed by the location of their occurrence in the query. Thus, for each word or k-tuple, a list of all the locations in the query sequence containing that word is generated. The target sequence is then scanned sequentially to identify potential matches by finding words in common with the query. When a word hit occurs, the program attempts to extend it on both the left and the right by checking if additional matching nucleotides can be found. If this extended word forms a significant ungapped segment

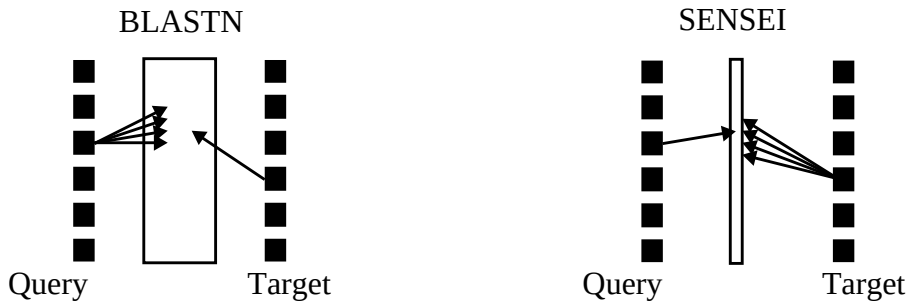


Figure 1. In BLASTN, multiple words for each query address are stored in the word look-up table. In SENSEI, only a single word for each query address is stored in the word table, and multiple words are generated for each target location at run-time.

```
Ungapped segment score: 39 (42 unmasked)
Expected frequency in random sequence 5.86e-08, unmasked 7.36e-09
Expected frequency using HSP composition 4.23e-07

Query:     25019 ttgacctcaggtgatccgccctcattggcctttcaaagtgctgggattac 25068
                 ..|.x.|||.|||||.|x||.x|||..||||..|.|..xlllll.|.|||
Target:    29879 ccgggttcaagtgattctcctgcatcagcctcccgagaagctggaactac 29830

Query:     25069 aagcatgagccaccacacccacccagtgctgttatttttagtg 25112
                 |.||.|xx|.|||||.||||.x|.|.mmmmmmmmmmmmmmmmmmm
Target:    29829 aggcgtctgtcaccatacccggctaatttttttgtatttttagt 29786
```

Figure 2. An example of a significant sequence repeat in the human G6PD locus that is discovered using the 1 bit/base encoding, but cannot be discovered using the 2 bit/base encoding. The bold face alignment denotes the initial 16-base (purine/ pyrimidine) word hit, matches are denoted by "|", transitions by ".", transversions by "x", and masked regions by "m". In this alignment, identities score +1, transitions 0, and transversions –1 ; this corresponds to a PAM 86 nucleotide scoring matrix with a transition to transversion ratio of 3.

(in the BLAST nomenclature, high-scoring pair or HSP) and its score achieves statistical significance, the extended word is saved.

A limitation of these approaches is that for sequence similarity to be discovered, the two sequences should share a common word or k-tuple. For example, with default parameters BLASTN requires an 11-nucleotide match (11-tuple). Search sensitivity can be increased for distantly related sequences by not only storing exact words, but also storing the inexact words. The first level of inexact words for an 11-tuple would be all the 11-tuples that differ from the original 11-tuple by one base. At each base position, there are 3 other possible bases leading to a total of 33 neighboring 11-tuples. In BLAST, this is implemented by storing in the word table all the k-tuples that match some position in the query sequence or its reverse complement either exactly or nearly exactly. For a 40 kbase query, and a word size of 11 nucleotides, storing all the words that match the query sequence in at least 10 of the 11 nucleotide positions requires a word table of 2 (strands) x (3x11+1) x 40 kbase x 2 bytes/ location= 5.42 Mbytes. This table size increases to over 84 Mbytes if higher sensitivity is required by matching at least 9 of the 11 nucleotides. In SENSEI, the size of the word-table is reduced by indexing only exact words derived from the positive strand of the query sequence. Target sequences are searched in both the forward and reverse-complement directions, and inexact word matches for the target sequence are generated at run-time. This difference between BLAST and SENSEI is illustrated in figure 1. In SENSEI, the word table is the same size as the length of the query sequence; it does not depend upon the number of neighboring words searched. For a 40 kbase query sequence, the word table stores 40,000 addresses. This reduction in table size is at the expense of a small run-time penalty associated with setting up the reverse-complement sequence and generating the target neighborhood words. However, this is not the rate limiting step in the neighborhood search algorithm; HSP extension consumes most (~90%) of the CPU time.

The size of the word table is reduced further by masking regions of the query sequence that correspond either to simple sequence repeats such as $(CA)_n$ or to species-specific repeats such as *Alu's* in humans.

SENSEI offers a choice of two alternative encodings for the DNA alphabet. The first option encodes each base using 2 bits; BLASTN uses the same encoding. However, SENSEI uses an index word of 8 sequential bases while the default for BLASTN is 11 bases. The second encoding option uses a single bit (purine/pyrimidine bit) to represent each base; thus A and G are indistinguishable: both are coded by the same bit (0), and C and T are indistinguishable.

This is a default option in SENSEI, but it is not available in BLASTN. For nearly identical sequences, better search sensitivity is obtained with the 2 bit/base encoding. For distantly related sequences, search sensitivity is improved by using the 1 bit/base encoding. Typically, the search time with the 1 bit/base encoding is smaller because the word table is less sensitive to biases in the base and oligonucleotide content of the sequences. Figure 2 displays a significant repetitive sequence from HUMG6PD that is discovered by using a 1 bit/base encoding and 16-tuple match, but is missed if a 2 bit/base encoding and an 8-tuple match is utilized. This is because the alignment contains a run of 16 matching purines and pyrimidines, but it does not contain a run of 8 identical bases. This is just one example of the kind of similarities for which the 1 bit/base encoding performs better; there are other examples for which the 2 bit/base encoding performs better.

Extending word hits on both sides to check if they result in a significant HSP is the most computationally expensive phase of the search for BLASTN. When a word hit is found, it is first scrutinized to see if it falls within a previously identified HSP; if not, the word hit is extended to see if it results in a new significant HSP. Traditionally, BLAST extends the word hit by moving a single base at a time. It would be more efficient to move forward *k (k >1)* bases at a time. However, the problem with extending *k* bases at a time is that it requires knowing the score between every pair of k-tuples, resulting in a score table with 4^{2k} entries (table 1). The size of this score table can be reduced by hashing. A simple and effective hash function is the logical *exclusive-or* (XOR) of the two k-tuples. Computing the XOR of single bases, each represented by 2 bits, provides four different resultant values (table 2). These can be used to index a hash table that provides specific scores for a match, transition, and transversion. For k-tuples, the XOR operation reduces the domain of the hash function to the square root of the size of the original domain; therefore, a significantly smaller hash table is required. This is equivalent to the function for $score(k\text{-}tuple_1, k\text{-}tuple_2)$ being replaced by $score(XOR(k\text{-}tuple_1, k\text{-}tuple_2))$.

In summary, SENSEI uses a logical exclusive-or (XOR) to encode the score table and extends HSP scores 8 base-pairs at a time. Theoretically, this approach can increase the rate of HSP extension by a factor of 8. In practice, the speed-up depends on the cache size and memory hierarchy of the CPU. Scores in this phase of the calculation are calculated using limited precision discrete scores. If an HSP is found to be close to significance, it is rescored using a double-precision scoring matrix. The masking of the query sequence for the presence of repeated sequences is incorporated in this rescoring by reporting both the masked and unmasked score.

Scoring systems

In most species, the various substitution mutations have unequal rates. In particular, transitions (a purine substituted by a purine or a pyrimidine substituted by a pyrimidine) are more frequent than transversions (purine-pyrimidine ex-

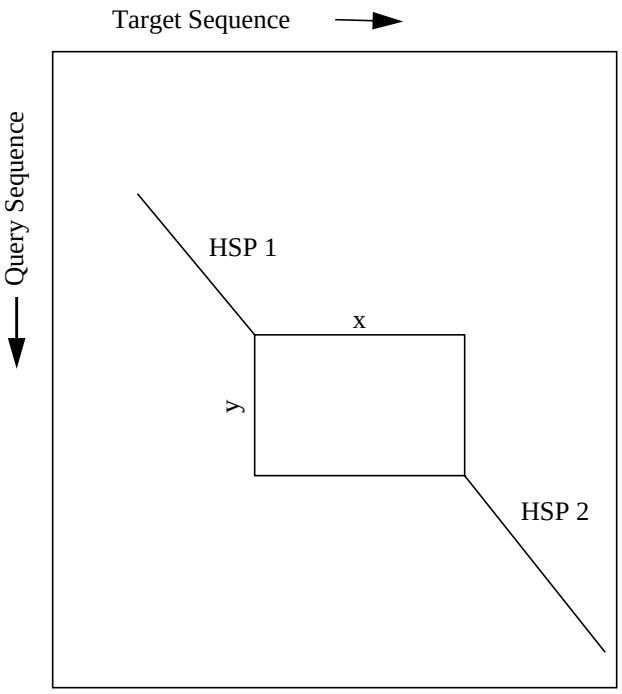
Table 1. The size of the score table for comparing k-tuples with direct score look-up and exclusive-or based hashing.

Oligo Length	Full Score Table Size	XOR Score Table Size
1	16	4
2	256	16
4	65,536	256
8	4,294,967,296	65,536
k	4^{2k}	4^k

Table 2. Representation of bases using 2 bits/base and the result of XOR operation.

	A (00)	G (01)	C (10)	T (11)
A (00)	0 (00)			
G (01)	1 (01)	0 (00)		
C (10)	2 (10)	3 (11)	0 (00)	
T (11)	3 (11)	2 (10)	1 (01)	0 (00)

Figure 3. The area searched at the end of HSP 1 to find HSP 2 is *xy*. Thus, a possible penalty for merging the two HSPs is $log_2(xy)$.



changes) by a factor of about 3 (Li, Wu, and Luo, 1985). Incorporating the correct model for evolution into the similarity scoring system increases the search sensitivity (States, Gish, and Altschul, 1992). The optimal scoring system also depends on the evolutionary distance (i.e., the number of substitutions that have occurred) between two homologous sequences. The transition/transversion rates and the evolutionary distance on which the scoring system is based can be specified for SENSEI searches. The evolutionary distance can be provided either as a fraction identity or as a PAM number (point accepted mutations, i.e., accepted substitutions per 100 bases) (Fitch and Margoliash, 1967; Dayhoff, Schwartz, and Orcutt, 1979). If the fraction identity is specified, the program automatically calculates the appropriate PAM number.

The scoring matrix is calculated and applied using double-precision floating-point arithmetic. This makes it impossible to use the dynamic programming algorithm to compute the K term in Karlin and Altschul statistics (Karlin and Altschul, 1990). Instead, a Monte Carlo sampling is performed. This calculation is also used to estimate the expected length per unit score (in bits) for a randomly occurring positive score alignment, so that the expected fraction of sequence masked at random can be estimated. All the scores are calculated and reported in bits (Altschul, 1991).

Assembly of gapped alignments

Two or more ungapped similar segments (HSPs) in a sequence comparison may be either an unrelated occurrence, or they may indicate the presence of insertion/deletion mutations. A choice of two different gap penalty measures is provided. The first is the traditional affine gap cost with a fixed penalty for introducing the gap and an extension term that increases linearly with the gap length. The alternative is a statistical estimate that penalizes the gap by the size of the space searched to find the next HSP. The penalty assessed is $log_2(xy)$ bits, where xy is the size of the space searched to find the next HSP (consider figure 3). If the sum of the scores of the HSPs including the gap penalty is more than the score of the individual HSPs, then they are combined into a gapped alignment. A dynamic programming algorithm (Wilbur and Lipman, 1983) is used to choose the optimal sets of HSPs for assembly into gapped alignments given the constraint that no HSP can be used more than once.

Short period repeat masking

Genomic sequences often include arrays of short period tandem repeats that generate artifactual high-scoring alignments. The elimination of these simple sequence repeats enhances the specificity of sequence similarity searches (Claverie and States, 1993). The XNUN short-period tandem- repeat filter is built into SENSEI. Either the threshold score for masking a tandem repeat or the longest period of the tandem repeats to be masked can be specified. If no threshold is specified, the program automatically calculates a threshold that will mask a small fraction (default 0.1%) of the query, if the query was a random sequence.

Search Sensitivity

The probability of identifying an anchor point for an HSP of length n with fraction f identical nucleotides using a neighborhood table with word size k is

$$P_{\text{sensitivity}} = 1 - (1 - f^k)^{n-k+1} \qquad (1)$$

This assumes that the $n - k + 1$ words are independent. If transversion bit (1 bit/base) encoding is used, then the fraction identity corresponds to the fraction of purine-purine and pyrimidine-pyrimidine matches. Assuming that a transition is three times more likely than a transversion, two sequences that share 50% sequence identity have the purines and pyrimidines conserved at 75% of the positions. Thus, the effective fraction identity is higher with a 1 bit/base encoding. Therefore, for the same size k-tuples, the 1 bit/base encoding will be more sensitive; however, it will be less specific—we will need to evaluate many more word hits. To balance the specificity and sensitivity, longer k-tuples are used with the 1 bit/base encoding. Comparable specificity is obtained by using an equal number of bits for both representations; typically, 16 bits are used, giving a k of 8 for the 2 bit/base encoding and 16 for the 1 bit/base encoding.

Figure 4 contains sample plots with the sensitivity plotted as a function of the HSP length for the two encodings. These are theoretical predictions computed from equation 1. For very short HSPs, the 2 bit/base encoding does better, but for both longer HSPs and for low sequence identity, the 1 bit/base encoding performs better, though neither performs exceedingly well with evolutionary-distant sequences. All these plots assume that the nucleotide substitutions follow the 3:1 ratio of transitions to transversions. The sensitivity of the 1 bit/base encoding is proportional to the ratio of the transitions to transversions. Thus, this 1 bit/base encoding is more sensitive, if the ratio of transitions to transversions is high, and less sensitive, if it is low.

Results and Discussion

Comparison with BLASTN

In the BLAST suite of programs, the program designed for searching a nucleotide query against a nucleotide database is BLASTN. SENSEI and BLASTN are similar tools as they both use neighborhood word indices to search for nucleic acid sequence similarities. SENSEI has several features that are designed to make it especially convenient for use in the analysis of large segments of genomic sequence. These features include

- integration of short-period tandem-repeat sequence masking,
- species-specific repeat masking,
- summary reporting of repeats and tandem arrays,

Figure 4. The sensitivities of the two encodings plotted as a function of the length of the homologous sequences and the percentage sequence identity as computed from the equation (1).

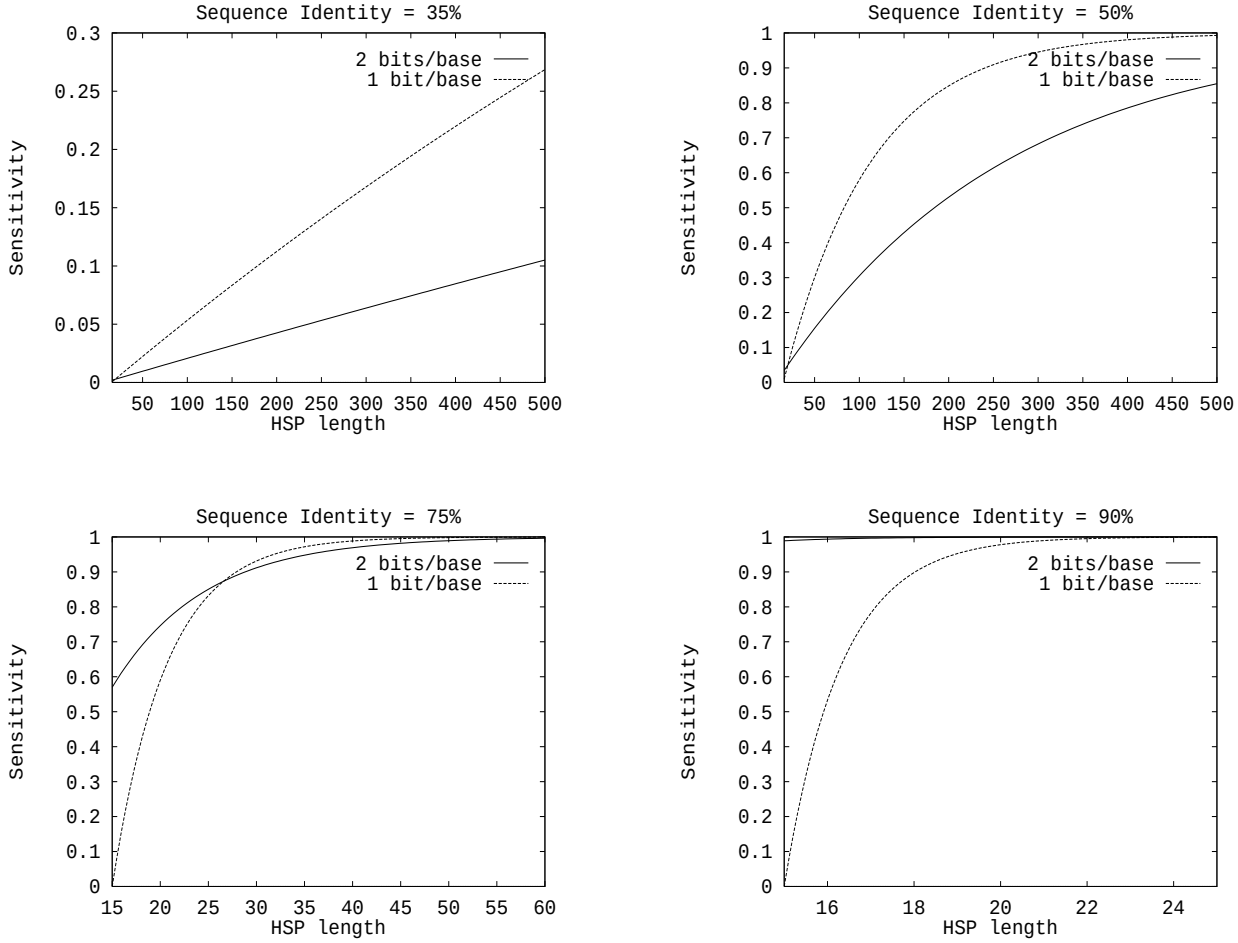


Table 3. Sample execution-times for SENSEI and BLAST to find all significant repeats in DNA genomic sequences. The "s" columns are speed-ups over BLASTN (W=8). A few of the important options for SENSEI are listed in columns 2 to 4. "E" specifies whether a 1 bit/base or a 2 bit/base encoding was used. "M" specifies if the query sequence was masked with known species-specific repeats. "X" specifies if XNUN was used on the sequence to mask simple sequence repeats.

Sequence				Human β globin (73 kbp)		Yeast Chr III (315 kbp)		Human T-cell receptor (685 kbp)	
Program	E	M	X	Time (in seconds)	s	Time (in seconds)	s	Time (in seconds)	s
BLASTN (W=8)				40	1.0	1,239	1	96,720	1
BLASTN (W=11)				18	2.2	203	6	24,896	4
SENSEI (mimic)	2	N	N	43	1.1	548	2	2,866	38
SENSEI (native)	1	Y	Y	18	2.2	51	24	345	280
SENSEI (no mask)	1	N	Y	7	5.7			348	278
SENSEI (2 bits/base)	2	N	Y	7	5.7	71	17	379	255
SENSEI (no XNUN)	1	N	N	10	4.0	60	21	420	230

- assembly of HSPs into gapped alignments,
- a flexible scoring system,
- enhanced search sensitivity,
- reduced memory requirements, particularly for long queries, and
- faster run-times for equivalent search sensitivity.

We hope that several of the features of SENSEI will be incorporated into future versions of BLASTN. BLASTN continues to be developed by Warren Gish and at NCBI. In particular, a major new release of BLAST is planned that will incorporate the assembly of HSPs into gapped alignments.

The data structure used to store the neighborhood table in BLAST requires more storage space per entry but is more efficient than SENSEI at storing and accessing very long index words. Expected score statistics in BLAST can incorporate knowledge of the database size and composition because BLAST uses a preprocessed copy of the database.

Empirical tests were performed to compare the execution speeds of SENSEI and BLASTN (version 1.4.8). These execution times are provided in table 3. Both BLASTN and SENSEI are reasonably complex programs with multiple parameters and optimizations. Thus, it is rather difficult to choose test data and parameter values that would treat both programs fairly. The relative execution times depend on the architecture, the amount of main memory, the size of the memory cache, the hand-optimization level of the program, and (most of all) the test data and the parameter values. For example, SENSEI provides the best performance in discovering evolutionary-distant repeats in large genomic human sequences, while BLASTN is optimized for locating homologs (with high-percentage identity) of smaller DNA sequences in large databases. BLASTN performs well with a word size of 11 bases, while SENSEI uses a word size of either 8 bases (2 bits/base) or 16 bases (1 bit/base). Thus, a "fair" comparison is rather difficult.

Table 3 contains some sample execution times in seconds for various modes of BLASTN and SENSEI. All these tests were run on a 55 MHz SUN Sparcstation 10 (Solaris 2.4). These test runs addressed the problem of finding all internal repeats (including local and distant, as well as direct and inverted) in three different large genomic DNA fragments: the human β-globin gene region on chromosome 11 (HUMHBB), yeast chromosome III (SCCHRIII), and the human germline T-cell receptor beta β chain (HUMTCRB). To discover all the repeats in a DNA sequence using BLASTN, the sequence is used both as the query and as the database; in addition, the *-span* option is used and *-hspmax* is set to a high value, so that all the repeats are found (Agarwal and States, 1994; 1996). For comparable sensitivity, the execution times for BLASTN (word size: W = 8) and SENSEI (mimic BLASTN with W=8) are utilized. The speedup obtained varies from almost nothing for the smaller sequence to a 38-fold improvement in speed for the long human sequence. The native configuration of SENSEI corresponds to 1 bit/base encoding, masking human-specific repeat sequences, such as *Alu*'s (Jurka, 1996), and low entropy sequence by XNUN. For the native configuration in comparison to BLASTN (W=8), there is a 2-fold improvement in speed for the β-globin locus, and a 280-fold improvement for the TCR-β region. We expect at most a 10- to 20-fold improvement in speed because of the improvements in the algorithm that have been discussed so far. The better-than-expected performance may possibly be attributed to other differences between the implementations. For example, BLASTN uses a finite automaton, and SENSEI uses a hash table. The automaton provides better speed at the expense of memory. For large query sequences, the automaton is not resident in memory cache resulting in poor performance. It is important to re-emphasize that BLASTN and SENSEI are general-purpose programs for DNA sequence similarity detection, and finding repetitive sequence in large genomic sequences is just one possible application.

The last three rows in the table show the comparable execution times for the various modes of SENSEI. Surprisingly, the execution times are rather similar for both the 1 bit and the 2 bits per base encodings. Masking by species-specific sequences also has a surprisingly minor effect; we predict that effect will be more appreciable as analyzed sequences get longer. The database of human-specific repeat sequences total about 82 kbp (Jurka, 1996); thus considerable search time is spent in computing the region of the query sequences that should be masked. Another benefit of masking the sequence is the easier interpretation of search results because the output is much smaller. For example, a single list of all the *Alu*'s is produced rather than a listing of the similarities between each pair of *Alu*'s in the query sequence. Filtering the sequence to remove short period tandem repeats (also called simple sequence repeats or microsatellites) using XNUN provides about a 30-50% improvement in speed. Versions of SENSEI that do not use XNUN take considerably longer to execute.

Availability

Executable versions of SENSEI for SUN Solaris 2.4, DEC Alpha, SGI mips 4, and SGI mips 5 are available from http://www.ibc.wustl.edu/sensei and ftp://ibc.wustl.edu/pub/sensei.

Acknowledgments

We wish to thank Michael Zuker and Warren Gish for informative discussions, the reviewers for their useful suggestions that helped improve this paper, and Laureen Treacy for proofreading the paper. This work was supported in part by Department of Energy grant DE-FG02-94ER61910.

References

Agarwal, P. and States, D.J. (1994). The Repeat Pattern Toolkit (RPT): Analyzing the structure and evolution of the *C. elegans* genome. In *Proceedings, Second International*

Conference on Intelligent Systems for Molecular Biology, 1-9, AAAI press, Menlo Park, CA.

Agarwal, P. and States, D.J. (1996). A Bayesian evolutionary distance from parametric sequence alignment. *J. Comp. Biol*. 3 (1), in press.

Altschul, S.F. (1991). Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.* 219: 555-565.

Altschul, S. F., Gish, W., Miller, W., Myers, E.W. and Lipman, D. J. (1990). Basic local alignment search tool. *J. Mol. Biol*. 215: 403-410.

Claverie, J.-M. and States, D.J. (1993). Information enhancement methods for large scale sequence analysis. *Comput. Chem.* 17: 191-201.

Dayhoff, M.O., Schwartz, R.M. and Orcutt, B.C. (1978). A Model of Evolutionary Change in Proteins. In M. O. Dayhoff (ed.), *Atlas of Protein Sequence and Structure*, National Biomedical Research Foundation, Washington, D.C. 5(3): 345-352.

Fitch, W. M., and Margoliash, E. (1967). Construction of phylogenetic trees. *Science* 155 (760): 279-284.

Jurka, J. (1996). Repbase. NCBI Repository. ftp://ncbi.nlm.nih.gov/repository/repbase/REF/humrep.ref

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. USA* 87: 2264-2268.

Li, W.-H., Wu, C.I. and Luo, C.C. (1985). A new method for estimating synonymous and nonsynonymous rates of nucleotide substitution considering the relative likelihood of nucleotide and codon changes. *Mol. Biol. Evol*. 2(2): 150-174.

Needleman, S. and Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequences of two proteins. *J. Mol. Biol*. 48: 444-453.

Pearson, W.R. and Lipman, D.J. (1988). Improved tools for biological sequence comparison. P*roc. Natl. Acad. Sci. USA* 85: 2444-2448.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J. Mol. Biol.* 147: 195-197

States, D. J., Gish, W. and Altschul, S. F. (1992). Improved sensitivity in nucleic acid database searches using application-specific scoring matrices. *Methods: A Companion to Methods in Enzymology* 3(1): 66-70.

Wilbur, W.J. and Lipman D.J. (1983). Rapid similarity searches of nucleic acid and protein data banks. *Proc. Natl. Acad. Sci. USA* 80: 726-730.