Variable-length Intervals in Homology Search

Abhijit Chattaraj **Hugh E. Williams**

School of Computer Science and Information Technology
RMIT University, GPO Box 2476V
Melbourne, Australia

{abhijit,hugh}@cs.rmit.edu.au

Abstract

Fast, accurate, and scalable search techniques for homology searching of large genomic collections are becoming an increasingly important requirement as genomic sequence collections continue to double in size almost yearly. Almost all homology search techniques rely on extracting fixed-length overlapping sequences from queries and database sequences, and comparing these as the first step in query evaluation; this is a feature of well-known tools such as FASTA, BLAST, and our own CAFE technique. In this paper we discuss a novel, variable-length approach to extracting subsequences that is based on homology scoring matrices. Our motivation is to achieve a balance between the speed and accuracy of fixed-length choices, that is, to encapsulate the speed of longer subsequence lengths and the accuracy of shorter ones. We show that incorporating this approach into our CAFE technique leads to a good compromise between accuracy and retrieval efficiency when searching with BLOSUM matrices sensitive to distant evolutionary relationships. We expect the same results would be achieved with other homology search techniques.

Keywords Homology search, Scoring matrices, Efficiency, Effectiveness

1 Introduction

Homology search techniques are used by biologists to investigate evolutionary relationships. Such searches are often made on the data contributed through world-wide collaborations of biologists that have helped determine complete genomes, including the human genome. The data is stored in large repositories of nucleotide and protein sequence data (Benson et al. 2002, Wu et al. 2002).

The primary structure of nucleotide sequences is represented by strings drawn from a four-letter alphabet, while protein sequences are strings over a twenty-letter alphabet. The linear string representation of genomic sequences allows the use of text retrieval and string matching techniques to be applied in searching genomic data (Setubal & Meidanis 1997).

Conventional genomic search techniques evaluate queries using a several step process. In the first step of this process — which is common to almost all techniques — fixed-length overlapping subsequences or *intervals* are extracted from a query and compared to intervals from database sequences. In the popular BLAST and FASTA techniques, this comparison is *exhaustive*, that is, the query intervals are compared to the intervals extracted from all database sequences (Altschul et al. 1990, Altschul et al. 1997, Lipman & Pearson 1985, Pearson & Lipman 1988).

To speed up this process, exhaustive search techniques are often adapted to run on specialised high-end hardware[1] or parallelised to permit greater throughput[2]. Hardware and parallelisation provides one solution. However, an entirely different approach is to use text indexing structures (Witten et al. 1999) that are commonly used in web search engines to determine the subset of the sequences in the collection that are similar to the query. One such successful approach to indexing genomic collections is our CAFE technique (Williams & Zobel 2002). However, regardless of whether the approach is exhaustive or index-based, intervals are the key to determining coarse similarity in the first step of the retrieval process.

The choice of interval length in the first step of the search process is crucial to efficiency. Short interval lengths are more sensitive to matches between sequences, but are less selective and, therefore, require more processing and result in slower searches. Long interval lengths are less sensitive, but more selective and permit faster processing. A related issue is the use of main-memory for the matching process: there are less unique short intervals, which allows a more compact main-memory model than for longer interval lengths. In practice, an interval length of $n = 2$ to $n = 4$ is used for protein databank searches, and $n = 9$ to $n = 12$ for nucleotide searches.

In this paper, we explore a novel variable-length interval extraction technique. Our aim in proposing this approach is to develop techniques that combine both the fast query evaluation property of longer fixed-length intervals and the accuracy characteristics of shorter fixed-length intervals. Our variable-length technique uses alignment scoring matrices to extract intervals that score equally: under this approach a

[1] For example, TeraBlast at http://www.timelogic.com/

[2] For example, TurboBlast at http://www.turbogenomics.com/

rare amino-acid that is likely to be strongly indicative of homology contributes to a short interval, while a common amino-acid contibutes to a longer interval. In protein databank searching, we show that variable-length intervals can reduce query evaluation costs compared to a short fixed-length choice by around 30% with only a small accuracy penalty. We conclude that variable-length intervals are a useful tool for fast and accurate genomic homology search.

2 Background

The basic requirement for understanding the function of both nucleotide and amino-acid sequences is establishing homology between two sequences. Homology means "possessing a common evolutionary origin" (Reeck et al. 1987), and it is often inferred by sequence comparison when it is found that two sequences share significant statistical similarity (Setubal & Meidanis 1997).

By establishing that homology exists, researchers can often infer the structure, function, role, and evolutionary history of an unknown sequence. Indeed, sequence comparison techniques for homology searching have been crucial in the discovery of many useful homologous relationships between sequences. Such discoveries have then led to advancements in critical areas such as cancer research.

The homology search process typically requires that a *query sequence* be compared to the *candidate sequences* in a genomic collection. The conventional approach to comparison is *sequence alignment* using a variant of Smith-Waterman local alignment (1981) that identifies the region of highest similarity between the query and each of the collection sequences. In this process, sequences are compared as a sequence of characters, and each possible alignment of characters is assigned a score. For example, when two identical characters are aligned — an *identity* — the score is usually a positive integer weight. In contrast, when two characters are different — or a insertion or deletion of a character is chosen — the score may be negative. We describe these scoring schemes in the next section, and return to homology search techniques in the following section.

2.1 Alignment Matrices

Precomputed *alignment* or *scoring matrices* that tabulate integer scores associated with character pairings are used for scoring in the amino-acid alignment process (Dayhoff 1978, Henikoff & Henikoff 1992), along with a function to compute the cost of inserting or deleting a character (Altschul & Erickson 1986). Alignment matrices are based on observations of the frequency of conservation and mutation of amino-acids in protein sequences. For nucleotide alignment, simple scores of $+5$ for an identity and -4 for a mismatch are most often used (Altschul et al. 1990); schemes to derive matrices based on observations of mutations have been proposed (States et al. 1991) but are not in widespread use.

The well-known PAM (Dayhoff 1978) matrices are derived from observed alignments of closely related sequences and tabulate the probability of each of the twenty amino-acids changing to another during an evolutionary interval. A PAM distance of one corresponds to a 1% change in an amino-acid sequence, that is, to the mutation on average of one amino-acid in a hundred. For example, a PAM0 matrix — which tabulates an evolutionary interval of zero — has all ones on the diagonal and zeros elsewhere. As PAM values increase — for example, in a PAM10 matrix — the values on the diagonal are close to one, while values elsewhere in the matrix are close to zero.

From the tabulation of probabilities in the PAM matrices, a log-odds matrix is derived so that scores can be summed and are integer values. In this derivation, the elements of the probability matrix are each divided by the frequency of the replacement amino-acid, so that each element is the probability of replacement of amino-acid a with amino-acid b per occurrence of amino-acid b. In general-purpose exploratory searching, the PAM250 matrix — which is sensitive to an evolutionary change of 2.5 amino-acids in each 100 — is often used.

The BLOSUM matrices are in most widespread use and are derived from ungapped local alignments of distantly related sequences (Henikoff 1993, Henikoff & Henikoff 1992). All matrices are calculated directly and, unlike the PAM matrices where approximations were used for rare events, no extrapolations are used. The frequently-used log-odds BLOSUM62 matrix is, for example, derived from sequences within families that display a 62% similarity.

The choice of alignment matrix is crucial to the outcome of an investigation. The PAM matrices were originally developed for global alignments, but have also been found to be useful in local alignment (Altschul 1991). However, the BLOSUM62 matrix has been shown to work well for general-purpose exploration and the BLOSUM matrices have been shown to be more useful for finding homologous sequences than the PAM matrices (Henikoff 1993). Recently, the PHAT and SLIM matrices (Muller et al. 2001, Ng et al. 2000) have been shown to outperform the BLOSUM matrices for specific tasks. However, overall, the BLOSUM matrices are the popular choice for most homology searching tasks and we use these in our experiments.

2.2 Homology Search

There are three popular techniques for general-purpose homology search with large genomic collections: exhaustive local alignment (Smith & Waterman 1981), FASTA (Lipman & Pearson 1985, Pearson & Lipman 1988), and BLAST (Altschul et al. 1990, Altschul et al. 1997). Of these techniques, BLAST is the most popular, with the US NCBI web search service processing around 120,000 queries per day on the large GenBank nucleotide collection (Madden 2003). Moreover, the GenBank collection is doubling in size almost yearly, query lengths are steadily increasing, and user numbers continue to grow (Williams 2003). However, despite this, users expect fast accurate answers to queries.

Heuristics are essential to fast and scalable homology search. Smith-Waterman local alignment is impractical on general-purpose hardware; the algorithm is $O(n^2)$ in both time and space, and would take several days to evaluate a query of moderate

length on the GenBank collection on modern general-purpose hardware. Therefore, both FASTA and BLAST use a several step process to identify a small set of candidate sequences that display broad similarity to a query before continuing with an heuristic, computationally expensive local alignment. However, even with these heuristics, both approaches take minutes to evaluate a single query using general-purpose hardware.

The first step in FASTA and BLAST is to pre-process the query sequence using a variant of the Wilbur & Lipman (1983) approach. Through this technique, fixed-length overlapping subsequences — we refer to these as *intervals*, and they are also known as n-mers, n-grams, or words — are stored in a fast lookup structure, along with the offset of the interval within the sequence. Consider the sequence ATTAATT and an interval length of $n = 3$. For this sequence, the intervals and their offsets with the sequence are ATT (1,5), TTA (2), TAA (3), and AAT (4).

After pre-preprocessing the query, each database sequence from the genomic collection is sequentially retrieved and parsed into its consituent intervals. The intervals are looked-up in the query search structure and, if the interval is present, then an initial match region is recorded by computing the difference in the query and database sequence offsets, and storing the identity of the matching interval. After the first step, the steps used in FASTA and BLAST diverge but the goal is the same: initial match regions that are likely to be indicative of homology are locally aligned, and results returned to the user.

We have previously proposed an alternative to the exhaustive approaches used in the popular homology search tools. The CAFE homology search technique uses an *inverted index* to support fast and scalable querying of large genomic collections (Williams & Zobel 2002). Inverted indexes (Witten et al. 1999) are used to support almost all information retrieval applications and, most prominently, are the search structures used by web search engines such as Google. Building an inverted index requires terms to be extracted from the collection to be searched — in the case of English text, this is usually words — and the creation of a list for each term that records, for example, a list of documents that the term appears in. Query evaluation proceeds by looking-up each term in the search structure, retrieving the lists associated with each term, and then retrieving the documents that contain the terms.

Similar to BLAST and FASTA, our CAFE approach is an interval based scheme. The search terms stored in the inverted index are the intervals extracted from the database sequences and, at query time, intervals are extracted from the query and matched against the index in the first step. Again, the motivation is to find a subset of sequences that have broad similarity to the query sequence, and then more computationally-intensive local alignment is used. Williams and Zobel showed that in 1997 their implementation of CAFE was more than eight times faster the the popular BLAST search system, and more scalable with increasing collection size.

In general, for all homology search techniques, short interval lengths permit slow, sensitive searching and longer intervals allow fast, selective searching. For example, for an interval of length $n = 1$, almost all sequences in a collection share similarity with the query; the search process is therefore sensitive and slow — almost all sequences are locally-aligned — but not selective in determining a subset of sequences. In contrast, for an interval length of $n = 10$, only a small fraction of the sequences in the collection contain a specific interval, and therefore only that fraction of sequences are locally-aligned; the process is therefore fast because it is selective, but less sensitive to distant similarity. For protein databank searching, interval lengths of $n = 2$, $n = 3$, or $n = 4$ are preferred.

3 Variable-Length Intervals

In this section, we propose a novel technique for extracting intervals from amino-acid sequences. Our aim in proposing this approach is to permit a compromise between the fast query evaluation of longer fixed-length intervals and the accurate query evaluation of short fixed-length intervals. Our approach uses alignment matrices to guide the interval extraction process, leading to *variable-length intervals*.

Codons are three-base nucleotide sequences that code for an amino-acid. Since there are 64 possible codons — there are 4 bases and therefore 64 possible three-base combinations — and only 20 amino-acids, there is redundancy in the coding process. For example, the amino-acid arginine is coded for by six codons; in contrast, tryptophan is coded by only one.

By using fixed-length intervals as index terms, the initial step in the heuristic homology search process described in the previous section neglects that some amino-acids may be better indicators of possible homology than others. To continue our example, a match between two tryptophan residues using the BLOSUM62 alignment matrix scores 11 in the alignment process, while an arginine scores only 5. We therefore might conclude that a single tryptophan identity is more than twice as strong an indicator of homology as an arginine identity.

Based on this observation, we propose a variable-length interval scheme where the interval length is dependent on the sum score of its composite amino-acids. In this approach, we set a threshold k as the minimum identity score of each interval. To extract intervals, we then set a score $s = 0$ prior to parsing the interval from the sequence. As an amino-acid residue is added to the interval, we use an alignment matrix to determine the identity score for that amino-acid and add the score to s. When s is greater than or equal to k, the interval is complete, and the process continues with the next overlapping interval.

To illustrate our approach, consider this technique applied to the sequence fragment AGVEWAEPT. Table 1 lists the identity scores from a BLOSUM62 matrix for each of the amino-acids in the fragment. The highest score awarded by this matrix is 11 for a tryptophan identity and, by default, we use this as the value of k; by setting $k = 11$, the minimum interval length is 1, that is, when the interval consists of a single tryptophan residue. For AGVEWAEPT, the first interval is AGV, since the scores of an A and G identity total $s = 10$, and the addition of V is required to

Amino Acid	Single Letter Code	Three Letter Code	BLOSUM62 Score	BLOSUM40 Score
Alanine	A	Ala	4	5
Glutamic Acid	E	Glu	5	7
Glycine	G	Gly	6	8
Proline	P	Pro	7	11
Threonine	T	Thr	5	6
Valine	V	Val	4	5
Tryptophan	W	Trp	11	19

Table 1: Selected amino-acid identity scores from BLOSUM matrices.

exceed $k = 11$. The second interval is GVE and scores $s = 15$, the third is VEW with $s = 20$, and the fourth EW with $s = 16$. At the completion of processing the fragment, the following intervals — with scores shown in brackets — are index terms: AGV (14), GVE (15), VEW (20), EW (16), W (11), AEP (16), EP (12), and PT (12).

The choice of scoring matrix and threshold k are important parameters in our variable-length approach. The diversity of identity scores correlates with the range of lengths of the variable-length intervals: for matrices that are sensitive to close evolutionary events, the range of scores varies, for example, from 4 to 20 in a BLOSUM30 matrix, but only from 6 to 16 for BLOSUM80. Similarly, the choice of k affects the minimum and maximum lengths: if k exceeds the maximum score (usually for a tryptophan identity), then the minimum interval length is two; similarly, k divided by the minimum score (usually for arginine) defines the maximum interval length We expect, therefore, that k should be carefully chosen with consideration to the matrix, and the desired range of interval lengths; as we show later, choosing values of k so that the minimum interval length is $n = 2$ works well in practice.

In addition to the twenty amino-acids, three wildcards Z, X, and B are used to represent the possible substitutions of more than one amino-acid. For example, the wildcard B represents D or N. The wildcard X represents any amino-acid and, in almost all matrices, scores negatively when aligned with itself. We tried several methods for handling a negative contribution to the value of s in creating variable-length intervals. However, since the collection we used contained only 1,701 wildcard occurrences across only 0.2% of the sequences, we found that the choice of approach had no significant impact on our results. We therefore chose a simple approach of replacing negative identity scores with a constant score of +1.

4 Experiments

In this section, we describe the test collection and query sets used in our experiments, and the measures we used for evaluating the accuracy of our approach.

4.1 Test Collection

To compare the retrieval effectiveness of different interval extraction approaches, we use a subset of the Protein Identification Resource–International Protein Sequence Database (PIR), a collection of well-classified amino-acid sequences (Barker et al. 2000). The PIR collection is divided into a set of four smaller databases, PIR1 to PIR4. Entries in PIR1 are annotated and fully classified, PIR2 entries are well-classified, sequences in the PIR3 collection are largely unclassified, and those in PIR4 are unclassified.

We used an approach similar to that used by Williams and Zobel (2002) in creating a collection for accuracy assessment, where we formed a collection based on the PIR1 and PIR2 databases. Sequences from PIR1 and PIR2 — in addition to being classified and annotated — are usually also assigned a *super-family* (SF) number. A super-family is a group of sequences that have the same domains in the same order, that is, they can reasonably be inferred as being homologous. We extracted from PIR1 and PIR2 those sequences with an SF number, and this resulted in a database of 67,543 sequences. The collection contains 22,675,479 residues, with a mean sequence length of 335.

4.2 Queries

The query set was compiled by selecting the first-occurring member sequence in the database from each super-family. Super-families with single members were excluded from the query set; there were 1,175 single member sequences. Sequences of lengths exceeding 500 bases were also removed, based on the assumption that amino-acid queries rarely tend to exceed this limit. After the above filtering process, the query set consisted of 4,021 queries.

We also compiled a smaller query set of 403 queries by selecting every tenth query from the larger query set. We used this smaller set in selected initial experiments that we describe later.

4.3 Measuring Accuracy

A common method of measuring relative retrieval effectiveness of information retrieval techniques are the measures of precision and recall (Bollmann 1983, Raghavan et al. 1989, Salton 1989, Witten et al. 1999). Recall is the proportion of the known relevant answers that have been retrieved, while precision is the proportion of relevant answers retrieved in the set of answers.

Recall is often said to be an impractical measure, because it is not always possible to judge all sequences in a collection as being relevant or irrelevant to each query. However, using the PIR collection and queries, it is possible to approximate recall if super-family sequences are deemed as relevant to a query drawn from

a super-family, and non super-family sequences are deemed as irrelevant. For example, using this approach the members of SF 12 are deemed as the only relevant answers to the query that was extracted from SF 12; this approach has limitations, which are discussed in detail by Williams & Zobel (2002), but the approach has been shown to offer a reasonable guide to the relative effectiveness of search techniques.

In our work, precision is therefore the fraction of relevant sequences retrieved:

$$\text{Precision} = \frac{\text{Relevant sequences retrieved}}{\text{Total sequences retrieved}}$$

Recall measures the proportion of total relevant sequences retrieved:

$$\text{Recall} = \frac{\text{Relevant sequences retrieved}}{\text{Total relevant sequences}}$$

We report accuracy as *11-point average interpolated precision*. Interpolated precision is the highest precision achieved at a particular recall level, and all subsequent levels of recall. We calculate average interpolated precision for all queries at eleven standard recall levels — from 0% to 100% in increments of 10% — and these precision values are then averaged into a single value.

Each recall level reflects the fraction of relevant answers reported, while the corresponding precision value is the fraction of reported answers that are relevant. The 0% recall level is treated as a special case and is assigned the highest precision achieved at any recall level (Witten et al. 1999).

5 Results

In this section, we describe the results of our experiments with fixed and variable-length intervals. We use our CAFE retrieval technique to evaluate accuracy and speed; however, we believe that the relative results are indicative of the tradeoffs that would occur in all interval-based homology search tools. All experiments were conducted on an Intel Pentium IV system under light-load, that is, where no other significant tasks were running.

5.1 Overall Results

Table 2 shows our overall results using a BLOSUM62 matrix and 4,021 queries on our PIR collection. The first three rows show the effect of choosing fixed interval lengths of $n = 3$ through to $n = 5$. As expected, with increasing fixed interval length, average 11-point precision declines and, as expected, accuracy is highest when the interval length is most sensitive at $n = 3$. In addition — and again as expected — average query time falls and index size increases from $n = 3$ to $n = 5$.

As more intervals are stored in the index structures of CAFE, the number of locations of the intervals falls, and therefore the index becomes less compressible. However, as the amount of information per interval decreases, less information is retrieved from disk per query, and query evaluation speed improves. Overall, the benefit of $n = 3$ is the 2.6% absolute improvement in accuracy over $n = 4$ but with a three-fold increase in average query time; these are similar results to those we have observed in BLAST.

The fourth and fifth lines of Table 2 show the effect of using variable-length intervals with two score thresholds of $k = 13$ and $k = 14$. For a threshold of $k = 13$, accuracy is around 0.7% less than $n = 3$ but speed improves by 0.09 seconds per query. For $k = 14$, the speed improvement is a substantial 0.65 seconds per query, with a penalty of an 1.05% reduction in absolute precision. Both variable-length results offer accuracy at least 1.5% better than $n = 4$, but are slower by a factor of two. Overall, our results show that variable-length intervals permit a compromise between the speed of fixed-length intervals of length $n = 4$ and the accuracy of $n = 3$.

5.2 Choosing the Threshold k

The choice of the interval score threshold k is crucial to the performance of variable-length intervals. Table 3 shows the effect of varying k for a BLOSUM62 matrix on accuracy, the number of intervals found in the collection, and interval length statistics; the trends are typical of the BLOSUM matrices we tested in the range BLOSUM30 to BLOSUM100.

Overall, our results show that accuracy peaks when k is the maximum identity score — in this case, 11 for tryptophan — or slightly higher. We therefore conclude that the value of k should be chosen so that the minimum interval length is 2, and the mean is close to 3.5; this is again not an unexpected result, since variable-length intervals have been proposed to offer a compromise between fixed-length intervals of $n = 3$ and $n = 4$.

5.3 Choosing a Matrix

Figure 1 shows how query evaluation speed and accuracy are affected by the choice of BLOSUM matrix. Our results show that for matrices in the range BLOSUM30 to BLOSUM50, variable-length intervals permit both accurate and fast query evaluation; for example, with a BLOSUM40 matrix and threshold of $k = 21$, accuracy is only around 0.7% less than fixed-length intervals of length $n = 3$, while being almost 30% faster. In contrast, for matrices above BLOSUM70, searching using variable-length intervals is unacceptably slow and accuracy falls.

Figure 2 shows on the x-axis the number of sequences that contain an interval plotted against the frequency of such intervals for two matrices. For example, there are just under 100 different intervals extracted using a BLOSUM40 matrix that occur in only 10 sequences. The distribution of intervals explains the improved speed for lower-numbered BLOSUM matrices: there are more intervals that occur fewer times for the BLOSUM40 than the BLOSUM62 and, therefore, the average number of initial match regions created in the query evaluation process is likely to be lower. This is perhaps unsurprising, given our observations previously that the diversity of scores affects the diversity of intervals, and that matrices sensitive to close evolutionary events are those that have score diversity. Variable-length intervals are therefore both accurate and fast for BLOSUM matrices sensitive to distant evolutionary events.

Scheme	Interval Parameter	Average Precision	Average Query Time (sec)	Total Index Size (Mb)
Fixed	$n = 3$	91.86	2.43	68.65
Fixed	$n = 4$	89.21	0.81	83.21
Fixed	$n = 5$	86.59	0.59	140.63
Variable	$k = 13$	91.10	2.34	70.15
Variable	$k = 14$	90.81	1.78	72.99

Table 2: Speed, accuracy, and index size for the fixed and variable-length schemes. Results are averaged for 4,021 queries using the BLOSUM62 matrix.

Interval	11-pt Average	Distinct	Interval Length			
Threshold k	Precision	Intervals	Maximum	Minimum	Mean	Median
4	75.43	257	5	1	2.40	2
6	81.57	1,113	7	1	2.93	3
8	86.93	2,600	9	1	3.23	3
9	88.76	4,707	10	1	3.35	3
10	90.25	7,589	11	1	3.45	3
11	91.69	10,360	12	2	3.52	3
12	91.50	15,303	13	2	3.67	4
13	91.06	26,317	14	2	3.83	4
14	91.09	44,747	15	2	3.93	4
15	90.19	67,930	16	2	3.98	4
20	89.12	932,651	21	2	4.92	5
25	85.37	6,062,082	26	3	5.77	6
30	79.48	11,898,501	31	3	6.50	7

Table 3: 11-point average precision, and interval statistics from searches using variable-length intervals. These experiments use the small 403 query set and the BLOSUM62 matrix.

6 Conclusion

With genomic sequence collections rapidly growing in size, there is a need for fast, accurate, and scalable homology search techniques. In this paper, we have investigated a novel technique for extracting *intervals* that are used to discover promising matches between queries and database sequences. In our approach, intervals have a variable length that is determined by the alignment matrix used in the search process. Our aim in proposing this approach is to offer a compromise between the accuracy of short fixed-length intervals and the speed of longer fixed-length intervals.

We have shown that our variable-length scheme works well for matrices sensitive to distant evolutionary events. For example, when searching with a BLOSUM40 matrix, our variable-length schemes are up to 30% faster than the most accurate fixed-length scheme with less than a 1% reduction in accuracy. Therefore, variable-length indexes offer a good compromise between accuracy and speed for index-based homology searching.

In future work, we plan to investigate the use of variable-length intervals in the initial phases of exhaustive techniques such as BLAST and FASTA. We also aim to further develop our techniques in order to apply variable-length intervals on nucleotide data. In addition, we plan to explore techniques that are accurate for closely related sequences.

References

Altschul, S. F. (1991), "Amino acid substitution matrices from an information theoretic perspective", *Journal of Molecular Biology* **219**, 555–565.

Altschul, S. F. & Erickson, B. W. (1986), "Optimal sequence alignment using affine gap costs", *Bulletin of Mathematical Biology* **48**(5-6), 603–616.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. (1990), "Basic local alignment search tool", *Journal of Molecular Biology* **215**, 403–410.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W. & Lipman, D. J. (1997), "Gapped Blast and Psi-Blast: a new generation of protein database search programs.", *Nucleic Acids Research* **25**, 3389–3402.

Barker, W., Garravelli, J., Huang, H., McGarvery, P., Orcutt, B., Srinivasrao, G., Xiao, C., Yeh, L., Ledley, R., Janda, J., Pfeiffer, F., Mewes, H., Tsugita, A. & Wu, C. (2000), "The protein information resource (PIR)", *Nucleic Acids Research* **25(1)**, 41–44.

Benson, D. A., Karsh-Mizrachi, I., Lipman, D. J., Ostell, J., Rapp, B. A. & Wheeler, D. L. (2002), "Genbank", *Nucleic Acids Research* **30**, 17–20.

Bollmann, P. (1983), The normalized recall and related measures, *in* M. McGill & M. Koll, eds, "Proceedings of the 6th annual international ACM SIGIR conference on Research and development in information retrieval", Bethesda, Maryland, pp. 122–128.

Dayhoff, M. O. (1978), *Atlas of Protein Sequence and Structure, Vol 5, supplement 3*, National Biomedical Research Foundation.

Henikoff, S. (1993), "Performance evaluation of amino acid substitution matrices", *Proteins* **17**(1), 49–61.

Henikoff, S. & Henikoff, J. (1992), "Amino acid substitution matrices from protein blocks", *Proc. National Academy of Sciences USA* **89**, 10915–10919.

Lipman, D. & Pearson, W. (1985), "Rapid and sensitive protein similarity searches", *Science* **227**, 1435–1441.

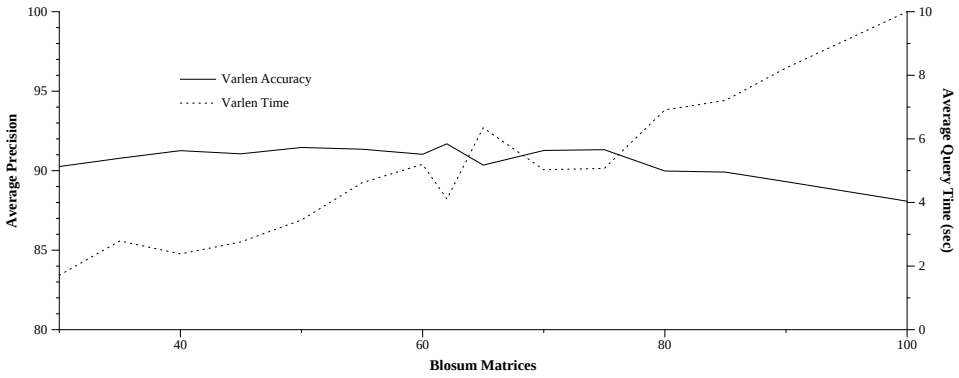

Figure 1: *Impact of* BLOSUM *matrix choice on the accuracy and query evaluation speed using variable-length intervals. The values shown are for the threshold k that achieved the maximum accuracy.*

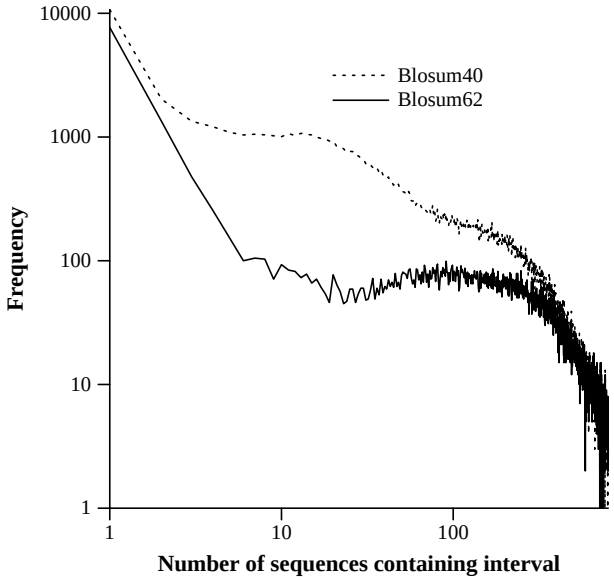

Figure 2: *Comparison of the distribution of intervals of different lengths for* BLOSUM40 *and* BLOSUM62 *matrices with variable-length intervals.*

Madden, T. (2003). National Center for Biotechnology Information. Personal communication.

Muller, T., Rahmann, S. & Rehmsmeier, M. (2001), "Non-symmetric score matrices and the detection of homologous transmembrance proteins", *Bioinformatics* **17**(Supplement 1), 760–766.

Ng, P. C., Henikoff, J. G. & Henikoff, S. (2000), "Phat: a transmembrane-specific substitution matrix", *Bioinformatics* **16**(9), 760–766.

Pearson, W. & Lipman, D. (1988), "Improved tools for biological sequence comparison", *Proc. National Academy of Sciences USA* **85**, 2444–2448.

Raghavan, V., Jung, G. & Bollmann, P. (1989), "A critical investigation of recall and precision as measures of retrieval system performance", *ACM Transactions on Information Systems* **7**(3), 205–229.

Reeck, G., de Haen, C., Teller, D., Doolittle, R., Fitch, W., Dickerson, R., Chambon, P., McLachlan, A., Margoliash, E., Jukes, T. & Zuckerandl, E. (1987), "Homology in proteins and nucleic acids: A terminology muddle and a way out of it", *Cell* **500**, 667.

Salton, G. (1989), *Automatic Text Processing*, Addison-Wesley.

Setubal, J. & Meidanis, J. (1997), *Introduction to Computational Molecular Biology.*, Brooks-Cole, Boston, USA.

Smith, T. F. & Waterman, M. S. (1981), "Identification of common molecular sequences", *Journal of Molecular Biology* **147**, 195–197.

States, D., Gish, W. & Altschul, S. (1991), "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. & Lipman, D. J. (1983), "Rapid similarity searches of nucleic acid and protein data banks", *Proceedings National Academy of Sciences USA* **80**, 726–730.

Williams, H. (2003), Genomic information retrieval, *in* K.-D. Scheme & X. Zhou, eds, "Australasian Database Conference", Australian Computer Society, Adelaide, Australia, pp. 27–35.

Williams, H. & Zobel, J. (2002), "Indexing and retrieval for genomic databases", *IEEE Transactions on Knowledge and Data Engineering* **14**(1), 63–78.

Witten, I. H., Moffat, A. & Bell, T. C. (1999), *Managing Gigabytes: Compressing and Indexing Documents and Images*, second edn, Morgan Kaufmann Publishing.

Wu, C., Huang, H., Arminski, L., Castro-Alvear, J., Chen, Y., Hu, Z., Ledley, R., Lewis, K., Mewes, H., Orcutt, B., Suzek, B., Tsugita, A., Vinayaka, C., Yeh, L., Zhang, J. & Barker, W. (2002), "The protein information resource: an integrated public resource of functional annotation of proteins", *Nucleic Acids Research* **30**, 35–37.