Indexing and Retrieval for Genomic Databases

Hugh E. Williams Justin Zobel
Department of Computer Science, RMIT University
GPO Box 2476V, Melbourne 3001, Australia
{hugh,jz}@cs.rmit.edu.au

Abstract

Genomic sequence databases are widely used by molecular biologists for homology searching. Amino-acid and nucleotide databases are increasing in size exponentially, and mean sequence lengths are also increasing. In searching such databases, it is desirable to use heuristics to perform computationally intensive local alignments on selected sequences only and to reduce the costs of the alignments that are attempted. We present an index-based approach for both selecting sequences that display broad similarity to a query and for fast local alignment. We show experimentally that the indexed approach results in significant savings in computationally intensive local alignments, and that index-based searching is as accurate as existing exhaustive search schemes.

Keywords Homology search, local alignment, indexing, genomic and scientific databases.

1 Introduction

Genomic databases assist molecular biologists in understanding the biochemical function, chemical structure, and evolutionary history of organisms. Popular systems for searching genomic databases match queries to answers by comparing a query to each of the sequences in the database. Efficiency in such exhaustive systems is crucial, since some servers process over 40,000 queries per day [26] and resolution of each query requires comparison to over one gigabyte of genomic sequence data. While exhaustive systems are practical at present, they are becoming prohibitively expensive—genomic databases are now doubling in size every 15 or 16 months, and user numbers and query rates are growing.

In this paper we investigate and propose new techniques for efficient, fast searching of genomic databases. In considering new approaches, we have developed several criteria for a successful implementation. First, a new system should support the same query types as existing exhaustive systems and be able to search the same databases. Second, the system must be fast and, importantly, scalable on general-purpose hardware in the presence of increasing user numbers, query rates, and database size. Third, the system should be comparable in accuracy to existing popular systems in identifying answers. Last, the system should have reasonable requirements for memory and disk space. We have previously described an initial implementation that addresses the second and last requirements [50, 47]. We propose in this paper new indexing and search techniques that successfully address all of the requirements.

```
GGGAATTCATGAACTCCGACTCCGAATGTCCATTGTCCCACGACGGTTACTGTTTGCAC
GACGGTGTTTGTATGTACATCGAAGCTTTGGACAAGTACGCTTGTAACTGTGTTGTTGG
TTACATCGGTGAAAGATGTCAATACAGAGACTTGAAGTGGTGGGAATTGAGATGATAAG
AATTCC
```

Figure 1: *Nucleotide structure of a synthesised human epidermal growth factor gene.*

Our indexing and retrieval techniques for querying genomic databases are embodied in a full-scale prototype retrieval system, CAFE [47, 48, 50, 51]. The disadvantages of indexing genomic databases are the need for time to build an index and for space to store the index on disk. The advantage of indexing is that searching is more scalable and much faster than exhaustive approaches. Cafe is based on techniques used in text retrieval and in approximate string matching for databases of names. The principal features of CAFE are the incorporation of new, efficient data structures for query resolution and the demonstration that, despite earlier negative results, indexing can be successfully applied to genomic databases.

We show experimentally that query evaluation using our new techniques has the requisite properties of speed, scalability, accuracy, and efficiency. With careful selection of parameters, CAFE can be used for the same search tasks as the popular FASTA [25, 30, 33] and BLAST [2, 6] search systems; FASTA has been shown to be the most sensitive rapid exhaustive search system, while BLAST is faster and more popular, but less accurate. In a direct comparison of CAFE to FASTA, we have found that CAFE is 50 to 100 times faster in searching the GenBank DNA database and has comparable accuracy. Moreover, the sensitive CAFE index is practical in size and the system remains efficient with increasing database size. Blast is much faster than FASTA but is still eight times slower than CAFE on the largest collection tested.

2 Background

Understanding the relationship of a query DNA or protein genomic sequence to well-understood sequences in a genomic database allows molecular biologists to assign function to poorly understood sequences. Indeed, one of the goals of sequence analysis is to determine sequence function, structure, and role from inspection and querying with a character string representation, or *linear sequence*, of a genomic sequence. In this section, we present a background of molecular biology, genomic databases, and techniques for practical linear sequence comparison.

2.1 Genomics

Genetic material, or DNA, stores complete instructions for all the cellular functions of an organism. The primary structure of DNA is represented as strings, or linear sequences, of a four-character alphabet, known as the *nucleotide bases*, represented by A, C, G, and T; a typical example is shown in Figure 1 [44]. In addition to the nucleotide bases, there are eleven standard wildcard characters used to represent different possible substitutions in a nucleotide sequence [24]. For example, B is used to represent the permitted substitution of either C, G, or T, but not A, into a sequence. The most common wildcard is N, which represents any base; some sequences contain thousands of occurrences of N, representing poorly understood regions of arbitrary length.

Many nucleotide sequences are precursors to the synthesis of proteins. Such *coding sequences* are used to transcribe RNA molecules, which are structurally similar to DNA molecules; the significant difference is that, in RNA, Uracil (U) replaces the DNA base T. One of the RNA molecules created with a coding sequence, mes-

```
MMNFFNFRCIHCRGNLHIAKNGLCSGCQKQIKSFPYCGHCGSELQYYAQHCGNCLKQEP
SWDKMVIIGHYIEPLSILIQRFKFQNQFWIDRTLARLLYLAVRDAKRTHQLKLPEAIIP
VPLYHFRQWRRGYNQADLLSQQLSRWLDIPNLNNIVKRVKHTYTQRGLSAKDRRQNLKN
AFSLAVSKNEFPYRRVALVFFVITTGSTLNEISKLLRKLGVEEIQVWGLARA
```

Figure 2: *Gene product from H. influenzae, a completely sequenced bacterial genome.*

senger RNA (mRNA), is based on the DNA template. Three-base combinations of the nucleotide bases from the mRNA, known as *codons*, transcribe amino-acids. Amino-acids in turn can be combined to create proteins. A *gene* is a region in a nucleotide sequence that codes a protein that performs a cellular function. Interestingly, in genomes such as the higher eukaryotes only a few percent of the DNA is coding, while the remainder is so-called "junk DNA" that is often repetitive in structure; other species, such as many bacteria, have much higher gene densities. Because coding regions are generally of more interest to molecular biologists, most genomic databases contain disproportionately large numbers of coding sequences.

Related nucleotide sequences from different species can have varied structure, where the distance in structure is proportional to the evolutionary distance of the two species. These regions are *homologous*, that is, derive from a common ancestor sequence, and identification of the existence of homology through sequence comparison of these regions can shed light on the evolutionary history, biochemical function, and chemical structure of these molecules.

There are twenty amino-acids, each of which is coded by between one and six codons. Proteins are polypeptide chains that typically contain tens or hundreds of amino-acids, while some consist of more than a thousand. A typical protein sequence, in this case a gene product from part of the *Haemophilus influenzae* bacteria [23], is shown in Figure 2.

Proteins have a complex structure dictated by the characteristics displayed by individual amino-acids, and amino-acids can be grouped according to characteristics including charge, hydrophobicity, and acidity. This classification of amino-acids allows prediction of relationships between sequences that are not easily seen by comparing nucleotide sequences. Moreover, because of the redundant mapping of codons to amino-acids, amino-acid sequences are much richer in information content. Because of this, in almost all cases, if an amino-acid sequence is available a molecular biologist uses it as a query to a genomic database in preference to the corresponding nucleotide sequence. However, occasionally nucleotide sequences are preferred as a query [7] and, often, an amino-acid sequence is not available and querying with a nucleotide sequence is the only possibility.

2.2 Genomic Databases

There are several public nucleotide sequence databases. Three of the larger repositories are GenBank [11], the DNA Databank of Japan, and the European Molecular Biology Laboratory database [36]. We use GenBank as the source of nucleotide test data for our experiments; the three databases are cross-updated daily and the three database structures are similar.

GenBank stores sequence data generated through the US human genome initiative, that not only focuses on the human genome, but also on model organisms such as the bacteria *E. coli*, the fruit-fly *D. melanogaster*, the nematode *C. elegans* and yeast *S. cerevisiae* [14]. The aim of the human genome initiative is determining the complete human sequence by 2003, with an intermediate goal of a "working draft" of the genome by 2001 [15]. The largest GenBank database (release 108.0, 15 August 1998) used in our experiments contains around 1,776 million nucleotide bases. Historically, the database has roughly doubled in size every 21 months since 1984, however GenBank is now doubling in size every 15 or 16 months: in August 1997 GenBank Release 102.0 contained 1,053 million bases, while the latest release 110.0 from December 1998 has over 2,162 million bases. The average sequence length is around 700 bases, with sequences ranging from a few bases to 300,000 bases in length; several sequences are longer than 300,000 bases, but have been stored as separate records according to GenBank guidelines. Data within GenBank is, in some cases, duplicated through the submission of identical and, rather more frequently, overlapping sequences. Additionally, there is a small but significant error rate, both as a result of sequence determination errors and of data entry errors [1].

GenBank contains amino-acid translations for many coding nucleotide sequences, however several solely protein databases also exist. Protein databases are typically well-managed and less redundant than nucleotide databases, commonly including classification of sequences into related families and, in some cases, superfamilies of families. Such databases include Swiss-PROT [8], which contains cross-references and data from around twenty smaller databases that investigate specific organisms and protein types. A typical specialist database is the Portable Mouse Genome Database [53].

In our experiments, we evaluate the accuracy of homology search systems using the Protein Identification Resource—International Protein Sequence Database, or PIR [20], a database of well-classified amino-acid sequences. Sequences in PIR are first classified by *homeomorphic domains*, that is, into families where the member sequences exceed a high threshold of sequence similarity, thereby inferring homology. Homologous families, with the same domains, in the same order, are then classified further into superfamilies, where the similarity threshold for grouping families into superfamilies is less stringent than that for originally grouping sequences [9]. Similarity scores, thresholds, and algorithms for determining similarity are discussed later.

2.3 Practical Sequence Comparison

With the widespread availability of practical sequence comparison techniques, molecular biologists have changed their approach to characterising sequences. Fundamental to understanding the function of genomic sequences is finding homology between two sequences. By comparing sequences and finding homology between two sequences, one of which has known function, structure, origin, or product, the biochemical role, evolutionary history, and chemical structure of the second unknown sequence may be inferred. Homologous sequences always share common elements of three-dimensional folding and secondary structure. Sometimes, however, homologous sequences do not share common function.

The most common method for analysing an unknown sequence is large-scale sequence comparison to characterised and annotated sequences in a large genomic database. Sequence comparison techniques have aided in the discovery of many useful homologous relationships between sequences. For example, recently, a gene that suppresses tumour growth in humans was found to be related to enzymes in the bacterium *E. coli* and in *C. cerevisiae*, a well-studied yeast genome.

To illustrate the use of sequence comparison, consider a simplistic example of a *mutation* in the bacteria *E. coli*. This mutation, or change in the nucleotide sequence, causes an individual to be unable to reproduce. The mutated sequence can be compared to other sequences to try and identify homology. If homologous sequences are found, and there has been additional work on a sequence from a different species, the product, for example a hormone, may be identified. This would allow the research on reproduction in *E. coli* to focus on that hormone.

Sequence comparison techniques measure statistical similarity of regions common to two sequences and, where statistical similarity exceeds a confidence value, homology is inferred. A common benchmark is that if more than 30% of two amino-acid sequences are identical, then the sequences are most likely homologous [17]. Lack of statistical similarity does not infer non-homology; for example, two sequences that do not share significant statistical similarity are homologous if both are related to a third sequence.

Generally, homology between sequences is measured locally, as similarity of regions, rather than measured globally as overall similarity of complete sequences. For example, by comparing two nucleotide sequences whose overall primary structure is dissimilar, local similarity measures may find homology between exons (coding regions) that are separated by differently composed and varying length introns (non-coding regions). Typically, only closely related amino-acid sequences are of the same length and overall composition. Nucleotide sequences, which are sections of a much longer strand and, therefore, have no notion of "ends", generally do not have overall similarity.

Estimation of evolutionary distance requires a measurement of the number of point mutations, or elementary changes, to transform one sequence into another. Generally, evolutionary distance estimations use string comparison algorithms to find the least number of mutations, that is, an *optimal alignment*, between two sequences. There may be many such optimal alignments that are equally evolutionally plausible and, indeed, equally interesting. This model of using specialised string comparison algorithms for genomic sequences has been shown to be an effective model of the evolutionary process [34].

To find an optimal alignment at a given evolutionary distance, scoring functions are used to measure and score each possible evolutionary pathway between two sequences, with the goal of finding the alignment, or set of alignments, with the highest similarity score. Similarity scores are calculated through pairwise alignment between two identical nucleotide bases or amino-acid *residues* in each of the sequences, or by scoring a mutation event. The measurement of local similarity using Smith-Waterman local alignment [42] is exhaustive and guarantees finding the optimal alignment, requiring $l \times m$ calculations of similarity for two sequences of lengths l and m. For a comparison of a sequence of length l to a database of N nucleotides, a total of $l \times N$ comparisons are required.

There are three general classes of mutation: substitution, insertion, and deletion. Substitution is the mutation, in the pairwise alignment of two sequences, of one residue into another residue, which may or may not be similar. Deletion is the non-alignment of a residue in the first sequence with any residue in a second sequence; deletion signifies that a particular residue is to be removed in the scoring of a particular evolutionary pathway between the two sequences. Insertion is the same as deletion, but from the perspective of the second sequence; if a residue is deleted from the first sequence, an alignment is achieved by inserting a null residue in the second sequence. Insertion and deletion are generally referred to collectively as an *indel*, where more than one consecutive indel is a *gap*.

An optimal local alignment of two globin sequences, human β-chain and α-chain hemoglobin, is shown in Figure 3. This optimal local alignment extending over 145 amino-acids is shown in the format typically returned to the user. Parameterisation of local alignment, that is, the choice of mutation data matrix for substitution and identity scores, and gap model costs are discussed elsewhere [3, 4, 10, 16, 40, 45]. An extract of the results of a database search of the PIR database with human α-chain hemoglobin are shown in Figure 4. In this extract, we show 3 of 695 ranked answers returned by our CAFE system, in a typical format for answers from a homology search system. The first answer shown is identical to human α-chain hemoglobin (and was ranked as the first response) but is an α-chain hemoglobin from a chimpanzee, the

```
                   10        20        30        40              50
HAHU        VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLS-----HGS
             | |- | -|-|-||||    -|-| |||-|   - |-| - |--|-|||      |
HBHU      MVHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGN
                  10          20        30        40        50

                60        70        80        90       100       110
HAHU      AQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAH
          - || |||||--|     || |  -     || ||--|| |||-|| ||  -| - ||-|
HBHU      PKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHH
          60        70        80        90       100       110

               120       130       140
HAHU      LPAEFTPAVHASLDKFLASVSTVLTSKYR
           --||||-| | - |- | |   | -||
HBHU      FGKEFTPPVQAAYQKVVAGVANALAHKYH
         120       130       140
```

Figure 3: *Local alignment of human β-chain hemoglobin (PIR code HBHU, 147 amino-acids) and α-chain hemoglobin (PIR code HAHU, 141 amino-acids). A BLOSUM-50 mutation matrix is used, with gap opening penalty of −12 and extension penalty of −2. The local alignment score is 381, with 43.4% identity in a 145 amino-acid overlap. A ' ' (space) indicates a conservative substitution, a '−' indicates both an indel and alignment of two dissimilar amino-acids, and a '|' indicates an identity (match).*

```
Ranking: 1   Scored : 558
Query    0 :LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKV   61
            |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subj     0 :LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKV   61

Query   61 :ADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHA  122
            |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subj    61 :ADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHA  122

Query  122 :SLDKFLASVSTVLTSK     138
            ||||||||||||||||
Subj   122 :SLDKFLASVSTVLTSK     138

Ranking: 282 Scored : 273
Query    4 :DKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DL---SH--GSAQVKGHGK   65
            |||-|-| ||||  |||| |||||-|||| || || |||-| ||   |   ||||||||||
Subj     6 :EKATVSGLWGKV--NADNVGAEALGRLLVVYPWTQRYFSKFGDLSSASAIMGNPQVKAHGK   67

Query   65 :KVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAV  126
            || |||-||| ||||| -||| ||||| ||||||| ||||||| |||-|| ||- |||| -
Subj    67 :KVINAFNDGLKHLDNLKGTFAHLSELHCDKLHVDPENFRLLGNMIVIVLGHHLGKEFTPCA  128

Query  126 :HASLDKFLASVSTVLTSK     144
            |||||| ||||||-|| |
Subj   128 :QAAFQKVVAGVASALAHK     146

Ranking: 695 Scored : 53
Query    5 :KTNVKAAWGKVG-AHAGEYGAEALERMFLSFPTTKTYFPHFD-LSHGSAQVKGHGKKVADA   66
            |||  - --| - |-|-| ||-  |  | | ||| -|||- | |||-|- -| -- ||-||
Subj   156 :KTNKPVIFTKSNLAKSPELDAKMYDICY-STAAAPIYFPPHHFVTHTSNGAR-YEFNLVDG 217

Query   66 :LTNAVAHVDDMPNALSA-LSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAE 118
             --|||| - || |||| ||| ||   -|-||||| |   |||||||  -||
Subj   217 :AVATVGDPALLSLSVATRLAQEDPAFSSIKSLDYKQM---LLLSLGTGTNSE 269
```

Figure 4: *Extract of the 695 ranked results of a* CAFE *search with human α-chain hemoglobin (PIR code HAHU) on the PIR database. The three results shown are ranked 1, 282, and 695 are, respectively, an identical, homologous α-chain hemoglobin from the chimpanzee, an homologous β-2-chain hemoglobin from the common rat, and an unrelated protein from a potato.*

second answer shown is ranked 282 and is a β-2 chain hemoglobin from a rat, and the third answer is ranked last and is a probably non-homologous precursor to a potato storage protein.

Fast exhaustive search techniques retrieve and process each sequence in a genomic database in response to a query. However, such systems typically use heuristics to reduce the number of sequences that require a local alignment to a small fraction of the database size. Given a set of database sequences, the well-known Wilbur-Lipman approach is to first pre-process, through hashing, each *interval* in the query sequence [46]. An interval in this context is a fixed-length overlapping subsequence from a sequence, such that there are $l - n + 1$ intervals, for a sequence of length l and interval length n. By first preprocessing the query into a hash table structure, each database sequence can be processed by hashing, and a single-step lookup used to locate each interval in the query hash table. If an interval is present in the search structure, the one or more matching offsets in the query can be retrieved, yielding considerable computational saving over scanning the query sequence for each database interval. With the offsets, scores for promising alignments without gaps can be calculated by taking the difference between the offset of the interval in the query and offset of the interval in the database sequence. An *accumulator* can then be used for each alignment without gaps to record a score for the matches.

There are three popular exhaustive search systems, all of which use variations of the Wilbur-Lipman interval approach for practical exhaustive searching. All systems have the limitation that an alignment is only possible where a common interval exists between a query and database sequence; in the case of BLAST, neighbourhood intervals—those that differ by one or two characters—may also form the basis of matches. We describe these systems, FASTA, BLAST 1, and BLAST 2 briefly.

The FASTA search system [25, 30, 33] uses a four-stage approach to alignment where the first stage is an application of the Wilbur-Lipman technique. After scoring matching intervals, the second stage re-scores the top ten accumulators for each sequence allowing gaps and the third stage attempts to join high-scoring regions represented by the accumulators. The final stage locally aligns the sequences using a memory-efficient implementation of local alignment [13] based on the regions joined in stage three. Three scores are reported for each sequence, along with a graph of the distribution of mean similarity scores and standard deviation of the scores for each sequence in the database. Our CAFE system, which we describe in detail in Section 3, uses similar heuristics to FASTA in the final stage of alignment.

Altschul et al. [2] propose additional heuristics in BLAST 1 to improve the database search times of the FASTA tool, with the goal of maintaining a comparable level of retrieval effectiveness. BLAST 1 deals with the high computational overhead associated with both an exhaustive database search and local alignment by using heuristics that find "ungapped, locally optimised ranked sequence alignments" [2]. This approach limits BLAST 1 to not allow for the insertion or deletion of residues, but only for the substitution of one residue for another. However, by building on the well-understood theory of ungapped local alignment [22, 5], BLAST 1 is able to effectively filter irrelevant answers.

BLAST 1 has an underlying assumption that indels are a less significant class of evolutionary event. This assumption has some merit in nucleotide comparison, since both single deletion and insertion events cause the meaning of codons to be completely lost. As such, it may be a reasonable compromise to ignore gapped alignments in coding regions of nucleotide sequences. However, FASTA has been shown to be significantly more sensitive than BLAST 1 at detecting distant homologous relationships, which typically contain more indel events [32, 41]. Shpaer et al. [41] have shown that there is on average one indel per fourteen aligned amino-acid residues, suggesting that while the well-developed ungapped alignment statistics may favourably filter false matches, BLAST 1 may also fail to detect many homologous sequences.

A new version of BLAST, known as BLAST 2 [6], aims to improve both the accuracy and speed of BLAST 1. BLAST 2 permits the limited use of indels in forming alignments, thus requiring more computation to evaluate each local alignment. To ensure that BLAST 2 is on average faster than BLAST 1 the criteria for attempting a local alignment are more stringent, with local alignment only permitted when two intervals (or neighbourhood intervals) match between a query and a database sequence. Altschul et al. have stated that BLAST 2 has improved accuracy over the previous versions [6]; however, the results presented later in this paper do not support this conclusion.

A significant problem with all exhaustive systems is that they may become prohibitively expensive because of the volume of data to be processed for each query. Conventional databases use indexing to provide fast access to relevant data. In particular, indexing has been shown to work well for information retrieval [39, 38], which has marked similarities to genomic retrieval: in both domains a typical query returns a large set of responses, many entries in the database exhibit some degree of similarity to the query, and matches are approximate rather than exact. Previous genomic indexing efforts have, however, been largely unsuccessful and indexed systems are not in widespread use. We discuss existing indexed approaches in the next section.

2.3.1 Indexed Genomic Searching

A general method for reducing searching costs is to store an abstraction or index that can be used to assess broad similarity to a query. The cost is the need to store the index, the potential saving is that fetching a limited volume of information should enable identification of a small number of sequences as likely answers, thus reducing both disk traffic and the computation required to resolve a query.

Interval-based indexing of genomic databases was first proposed by Orcutt and Barker [29]. Specifically, Orcutt and Barker proposed their algorithm as a method of identifying amino-acid sequences in the PIR database. No detail is given of an implementation of the approach, but Barton [10] notes that an implementation, SCAN, was available with the PIR database for use in exact matching.

The SCAN approach was not highly successful and has not been developed for several reasons. First, simple measurements of matching intervals are used as a ranking technique, limiting the sensitivity and selectivity of the indexed approach; we have found that without novel ranking techniques ranking detects similarity in composition but not necessarily homology. Second, non-overlapping intervals are extracted from the database and query sequences; all current exhaustive and indexed approaches, including BLAST and FASTA, require overlapping intervals to achieve acceptable retrieval effectiveness. Third, the algorithms of Orcutt and Barker do not use current approaches to managing large document collections that make indexing practical; for example, queries are limited to 25 residues in length and search structures are unlikely to have been compressed, resulting in a large index and increased disk transfer times.

Altschul et al. [2] have implemented a similar approach to SCAN that uses a table of all database intervals. It was found, possibly because of limitations similar to those in SCAN and because GenBank was around 45 times smaller in 1990 than it is now, that this approach was somewhat slower than exhaustively searching the database.

The most recent indexed scheme is the Rapid Access Motif database (RAMDB) system for finding short patterns in genomic databases [19]; such patterns, or *motifs*, are typically around ten bases in length. In the approach of Fondrat and Dessen, each genomic sequence is indexed by its constituent overlapping intervals in a hash table structure. For each interval in the collection, an associated list of sequence numbers and offsets is stored, allowing rapid location of any motif matching a query motif.

The primary application of RAMDB is the location of motifs either equal or slightly longer in length than the indexed interval length. The indexed approach of RAMDB is shown to result in a 0 to 800-fold speedup in search times over comparable exhaustive approximate pattern matching approaches.

The FLASH search tool redundantly indexes genomic data based on a probabilistic scheme [12]. For each interval of length n, the FLASH search structure stores, in a hash-table, all possible similarly-ordered contiguous and non-contiguous subsequences of length m that begin with the first base in the interval, where $m < n$. As an example, for a nucleotide sequence ACCTGATT the index terms for the first $n = 5$ bases, where $m = 3$, would be ACC, ACT, ACG, ACT, ACG, and ATG; each of the permuted strings begins with the base A, the first base in the interval of length $n = 5$. The hash-table then stores each permuted m-length subsequence, the sequences that contain the permuted subsequences, and the offsets within each sequence of the permuted subsequence. The permuted scheme gives an accurate model that approximates a reasonable number of insertions, deletions, and substitutions in genomic sequences.

Califano and Rigoutsos found that FLASH was of the order of ten times faster for a small test collection than BLAST and was clearly superior in accurately and sensitively determining homologies in database searching. In addition, given adequate system resources, scaling-up of the system to larger databases suggested that the computational time saving would remain at a similar order for the complete GenBank collection.

However, the redundant index, which is stored in a hash-table and is uncompressed, is impractically large. Rigoutsos and Califano report that, for a nucleotide collection of around 100 Mb, the index requires 18 Gb on disk, around 180 times the collection size. Barton [10] reports that the index for Swiss-PROT Release 25, a collection of around 10 Mb, requires almost 2.8 Gb of disk space, around 280 times the size of the Swiss-PROT collection.

3 Indexed Genomic Retrieval with Cafe

Inverted files have been shown to be a successful tool for large text database retrieval [27, 39, 54]. In such environments, indexes are used to selectively retrieve relevant records without exhaustive scanning of the database. Indexing trades space against time; for the cost of storing the index, retrieval is typically many orders of magnitude faster than an exhaustive search.

To address the problems with indexing encountered in other attempts, we propose a two-component partitioned search process embodied in a research prototype system, CAFE. The first component of our approach, a *coarse search*, uses an inverted index to select a subset of sequences that display broad similarity to the query sequence. The second component, a computationally more expensive *fine search* mechanism, ranks the resultant sequences from the coarse search in order of relevance to the query, presenting the ranked results to the user. The partitioning of searching into coarse and fine mechanisms has, for example, been successfully used for pattern matching in databases of names [57, 58]. To ensure efficient query evaluation, we use a query evaluation technique adapted from such methods.

A significant feature of CAFE is our method of addressing the problem of index size, where we have adapted compression techniques developed for text indexing [27]. In text indexing, index size is reduced with careful compression by a factor of three to six, while in genomic databases we have found that more than three-fold reductions are possible. We have previously discussed the compression of CAFE indexes in a preliminary description of our approach [50]; in addition, we have developed a method for compressing genomic nucleotide sequences that reduces the query evaluation costs in our CAFE system by over 20% [51].

In this section we explore coarse searching, that is, using an index to assess broad sequence similarity prior to retrieving and fine searching the sequence data. Our novel fine searching approach is not described in detail in this paper, but is a gapped scheme that is similar in sensitivity and technique to the FASTA approach. We present details of our fine searching local alignment algorithm elsewhere [48]. To evaluate the CAFE approach, we present in Section 4 a framework to enable comparison of aspects of rapid homology search tools, such as retrieval effectiveness, speed, and space. We use this evaluation framework to compare CAFE to BLAST and FASTA.

3.1 Indexing with Cafe

To achieve efficient and effective retrieval from genomic databases, we propose several modifications to the methods used for general strings. Improvements are required for the following reasons. First, to reduce the computational overhead of fine searching, it is preferable for the coarse search phase to provide a framework for subsequent local alignment; fine searching using a preliminary version of CAFE [50] consumes between 40%–90% of the total query evaluation cost. Second, because of the length of the stored sequences, simply identifying which sequences are likely matches requires the often inefficient retrieval of complete genomic sequences. Typical nucleotide queries are hundreds of base-pairs in length, while some sequences in GenBank are around 300,000 base-pairs. By incorporating extra information in the index, it is possible to identify where in the sequence a similar region can be found and allow partial sequence retrieval of only the matching region.

For indexing genomic data we suggest that an appropriate choice of index term is the intervals occurring in each sequence, where the intervals are overlapping substrings of some fixed-length n; choice of n is discussed later. For example, if $n = 3$ then the intervals of ACCTGTC are ACC, CCT, CTG, TGT, and GTC. Fixed-length overlapping intervals have been shown to be practical in other indexed [19] and exhaustive search systems [2, 25, 30, 33]. In addition, fixed-length overlapping intervals work well in other areas of genomics, including query filtering [49, 56], fragment assembly [28], consensus alignment [55], sequence classification [37], and pattern detection [35]. Indexing on fixed-length substrings has also been successfully used in pattern-matching for large lexicons [57, 58], a domain that is however rather different: strings in lexicons are typically around ten characters, not hundreds of characters, and similarity is global rather than local.

An inverted index has two components: a search structure and postings lists. The search structure consists of the set of unique searchable terms, in this case the set of intervals, while associated with each term in the search structure is a postings list. The postings list contains the ordinal numbers of the documents containing the associated search term. The CAFE inverted file indexing scheme is extended so that within each postings list is stored not only the ordinal sequence number that contains the interval, but also offset information. For example, consider the following postings list

ACCC 12,(3:144,154,962),38,(2:47,1045)$\cdots$

in which the indexed sequences, the 12th and 38th, contain the interval ACCC. The interval occurs 3 times in the 12th sequence, at offsets 144, 154, and 962, and twice in the 38th sequence at offsets 47 and 1,045.

With postings lists typically on disk, an inverted index is intensive in disk usage—frequently a large number of postings lists are retrieved and the average length of postings lists is high. To reduce the overhead of using an index for retrieval, compression techniques used for text database indexes [27] and string indexing [57, 58] are used to reduce index size. The benefits of compression are two-fold: there is a saving in space used by the index and often a saving in query evaluation time—if retrieval of compressed lists and subsequent decompression is faster than retrieving uncompressed lists [52, 54].

3.2 Coarse Searching with Cafe

To achieve efficient and effective retrieval from genomic databases, we propose several ranking techniques that use our index structure. We introduce in this section a novel ranking structure, FRAMES, that addresses the deficiencies of simple ranking schemes. In particular, FRAMES reduces the number of sequences that need to be fine searched, by supporting accurate, inexpensive metrics for coarse ranking with a fine-grain index. Frame-based metrics incorporate the relative positioning of matching intervals, as well as other calculated metrics, to give a model of likely homologous alignments. We have previously described a preliminary implementation of FRAMES for nucleotide searching [47].

By using the offsets stored in the fine-grain index, a ranking structure can be constructed to selectively and accurately detect homologous sequences. We refer to this structure as FRAMES. A frame is a set of one or more matching intervals between a database sequence and a query sequence that are at the same relative offset. There can be many frames created and each frame is treated independently, regardless of whether the frame represents intervals from different matching offsets from the same database sequence, or is from a different database sequence.

To illustrate FRAMES, consider Figure 5, which shows three matching frames between two sequences. Each of Figures 5(A), 5(B), and 5(C) represents a single frame. In this example, with an interval length of $n = 3$, we have identified four distinct regions of at least the interval length that match between the sequences; two regions are shown in Figure 5(A) and one each in Figures 5(B) and 5(C). There are other intervals and regions that match between the two sequences that have been omitted for clarity in the figure.

To show how the four regions form three frames, consider the first region shown in Figure 5(A), which is a match between GTTT at offset 9 in the first sequence and offset 7 in the second. The relative offset of the match is $9-7 = 2$, creating the frame F_2, which contains the offsets of each of the two interval matches $\{(9,7),(10,8)\}$; the offsets within a frame can be read as an interval-length match between two sequences beginning at the respective offsets in each sequence. The second region in Figure 5(A) contains six intervals matches and adds to frame F_2, which then contains eight tuples:

$$\{(9,7),(10,8),(27,25),(28,26),(29,27),(30,28),(31,29),(32,30)\}$$

Figure 5(B) shows a matching region containing two intervals TGG and GGG, beginning at offsets 17 and 16 of the two sequences respectively. This creates a second frame F_1, since $17-16 = 1$, containing the two interval matches $\{(17,16),(18,17)\}$. Figure 5(C) shows two matching intervals that create a third new frame F_{26} containing $\{(53,27),(54,28)\}$.

A simple ranking scheme would rank sequence similarity based on the interval matches between the two sequences. For example, an obvious simple ranking scheme would be to score matches based on the count of interval matches between the two sequences over all regions. In our example, the total score would be twelve, as there are twelve intervals in the four regions identified. This coarse-grain ranking

```
A            10        20        30        40        50
    ACCCTGAGGTTTTTTTTGGGAGAGCTTTCTTCTTAGAGAGGAGGCTAGCTAGCTTCG
            ::::              ::::::::
      GTGTGTGTTTGTGTGTGGGGTAAGTTCTTCTTCTT
               10        20        30

B            10        20        30        40        50
    ACCCTGAGGTTTTTTTTGGGAGAGCTTTCTTCTTAGAGAGGAGGCTAGCTAGCTTCG
                    ::::
     GTGTGTGTTTGTGTGTGGGGTAAGTTCTTCTTCTT
              10        20        30

C            10        20        30        40        50
    ACCCTGAGGTTTTTTTTGGGAGAGCTTTCTTCTTAGAGAGGAGGCTAGCTAGCTTCG
                                                        ::::
                              GTGTGTGTTTGTGTGTGGGGTAAGTTCTTCTTCTT
                                       10        20        30
```

Figure 5: *Alignment of two sequences showing three frames, where each frame consists of matching intervals of length $n = 3$ at the same relative offset. (A) The first frame, frame F_2, consists of two regions: the first begins at offset 9 in the first sequence and offset 7 in the second sequence; the second begins at offsets 27 and 25. (B) The second frame, frame F_1, contains one matching region, beginning at offset 17 in the first sequence and offset 16 in the second. (C) The third frame, frame F_{26}, contains one region.*

overestimates the similarity between the two sequences, since it is not possible to present a local alignment based on all of the identified interval matches.

By using FRAMES, coarse ranking can provide a notion of order and arrangement of common regions. The frames F_1 and F_2 cannot be combined with F_{26} to form a plausible alignment and should be treated independently in coarse ranking for subsequent fine searching. This independence of regions is reflected by using the FRAMES structure, where a coarse search of the two sequences identifies three separate frames, of which only one is likely to result in an interesting alignment. The two frames with shorter matching regions will be ranked equally with each other and, assuming ranking uses a simple count of frame offset tuples, equal with any other frames from other database sequences that contain only two interval tuples.

Often, however, related frames may be able to be combined to form a single alignment with gaps. For example, frames F_1 and F_2 may be able to be combined with a single indel to form a higher-scoring alignment than the individual ungapped alignments. We discuss heuristics for forming frames into weighted coarse ranked *neighbourhoods* later.

The FRAMES approach is similar to the heuristic method for local alignment of two sequences proposed by Wilbur and Lipman [46] and applied in the FASTA exhaustive search system. However, the significant advantage of FRAMES is that it can be applied to inverted lists, making it possible to rapidly rank both regions within sequences and amongst sequences. Moreover, accurate scoring metrics that measure overall similarity of ungapped alignments [22] or gapped alignment statistics can be calculated. There are several such possible scoring metrics.

A simple scoring metric that can be calculated using frames is to rank frames based on the number of intervals in each frame for two sequences s and t, so that

$$\text{framecount}(s,t) = max(|F(I(s) \cap I(t))|)$$

where $I(s)$ is the set of intervals in sequence s, $I(t)$ the intervals in sequence t, and $F()$ the frame function that returns one or more sets of intervals that are at the same relative offset. For Figure 5, the framecount measure would identify frame F_2

```
A            10        20        30        40        50
    ACCCTGAGGATTTTTTTGGGAGAGCTTTCTTCTTCGAGAGGAGGCTAGCTAGCTTCG
                              :::::::::
    ACGTGTGTGTTTGTGTGTGGGGTAAGTTCTTCTTCTTCTCTTTCTCTTTCTTTCCTC
             10        20        30        40        50
B            10        20        30        40        50
    ACCCTGAGGATTTTTTAAAGAGAGCTCCCTTAGGAGAGAGGAGGCTAGCTAGCTTCG
    :::   :::       :::         :::          :::  :::    :::
    ACCTGTAGGTTTGTGCAAAAGGTAAGTTCTTCTTCTTCTCTAGGATAGTTCTCTTAT
             10        20        30        40        50
```

Figure 6: *An illustration of* COVERAGE *in a single frame, with an interval length of $n = 3$. Other frames exist between the sequences shown, but these are omitted in this example. (A) Two sequences of the same length are shown, with a single frame containing 7 overlapping interval matches. In this case, the* COVERAGE *is 9, since there are 9 base identities. (B) Two similar sequences of the same length are shown, with a single frame containing 7 non-overlapping interval matches. In this example, the* COVERAGE *is 21, as there are 21 base identities in the frame.*

as the best coarse match, since the cardinality of frame F_2 is 8, and frames F_1 and F_{26} have a cardinality of 2.

This simple FRAMECOUNT measure can be refined to incorporate more information than the cardinality of frames. For example, by considering the relative positioning of intervals in a single frame, we can calculate two computationally inexpensive metrics for ranking: COVERAGE and LENGTH.

Frame COVERAGE is a count of the actual number of residues or bases that match between two sequences. Given that a frame may consist of multiple overlapping intervals, a new interval of length n may contribute between 0 and n new base or residue identities to the frame. We propose scoring a frame using the COVERAGE, where frames with higher numbers of distinct non-overlapping intervals are ranked higher.

Figure 6 shows an example that contrasts the FRAMECOUNT scheme with COVERAGE. In the case of FRAMECOUNT, both Figures 6(A) and 6(B) score 7, since in both cases there are 7 matching intervals of length $n = 3$ in the frames shown; we omit other frames that exist between the sequences in this example. Although FRAMECOUNT and COVERAGE scores cannot be directly compared, in the case of COVERAGE, the frame shown in Figure 6(A) scores 9, since 9 base identities are contributed by the alignment of the 7 matching intervals. In contrast, the COVERAGE contribution of the 7 non-overlapping intervals in Figure 6(B) is $7 \times 3 = 21$. Although COVERAGE is reasonably cheap to evaluate, it is simplistic and relies on the assumption that a frame with higher COVERAGE is statistically more likely to result in a high-scoring local alignment. However, we have found that COVERAGE works well in identifying homologous sequences [48].

The LENGTH of a frame match is the total number of bases that lie between the two intervals that have the smallest and largest offsets. In Figure 7, extending Figure 6, we again illustrate one frame for each of three sequence pairs containing seven interval matches of length $n = 3$. The LENGTH is the difference between the minimum and maximum offsets covered by the frame intervals, in the case of Figure 7(A) a LENGTH of 21 and for Figure 7(B) a LENGTH of 55. This scheme weights a frame highly if it covers a larger region of the query sequence.

As in COVERAGE, the LENGTH scheme does not consider all aspects of the interval matches in a particular frame. For example, Figure 7(C) shows a frame that has the same LENGTH of 55 as the frame in Figure 7(B). Figure 7(C) has two interval matches and no obvious homology between the matches, while in Figure 7(B) there

```
A             10        20        30        40        50
    ACCATGATGATTTTGTACAGAAAGCTCCTTTAGGAGAGAGGCGGCTCGCTAGCATCG
    ::: ::: :::: ::: ::::
    ACCCTGAGGATTGTGCCCAGAGTAAGTTCTTCTTCTTCTCTAGGATAGTTCTCTTAT
              10        20        30        40        50
B             10        20        30        40        50
    ACCCTGAGGATTTTTTAAAGAGAGCTCCCTTAGGATACACGAGGCTAGCTAGCTTCG
    :::   :::       :::         :::          ::: :::    :::
    ACCTGTAGGTTTGTGCAAAAGGTAAGTTCTTCTTCTTCTCTAGGATAGTTCTCTTAT
              10        20        30        40        50
C             10        20        30        40        50
    ACCCTGCTTATTTTTTTTTGAGAGCTCCTCCAGGATACACGGAGCTACCTAGCTTCG
    :::                                                 :::
    ACCTGTAGGTTTGTGCAAAAGGTAAGTTCTTCTTCTTCTCTAGGATAGTTCTCTTAT
              10        20        30        40        50
```

Figure 7: *An illustration of* LENGTH *in a single frame, with an interval length of* $n = 3$. *Other frames exist between the sequences shown, but these are omitted in this example. (A) Two sequences of the same length are shown, with a single frame containing 7 interval matches. In this case, the* LENGTH *is 21, since the total length of the region in the frame is 21 bases. (B) Two similar sequences of the same length are shown, with a single frame again containing 7 interval matches. In this example, the* LENGTH *is 55, since the total region length for the frame is 55 bases. (C) Two sequences are shown with a frame that has the same* LENGTH *of 55 as the sequences shown in (B), but has no obvious homology between the matching intervals.*

are seven interval matches and a much higher overall similarity between the two sequences.

Without a combined consideration of COVERAGE, the LENGTH scheme does not factor in the probability of a homologous local alignment in the absence or presence of matching intervals between the extremities. In particular, the probability of a successful local alignment of distant intervals decreases with increasing distance. Despite this, the LENGTH scheme is particularly attractive since it ranks highly regions that are longer and, therefore, will rank long homologous alignments ahead of shorter alignments. We have found that LENGTH is reasonable in identifying homologous sequences, but not as accurate as the COVERAGE approach [48].

We propose also a combined coverage and length scheme that addresses some of the problems in COVERAGE and LENGTH. This new approach, which we call COMBINED, factors in both COVERAGE and LENGTH in a frame, so that

$$\text{COMBINED} = \text{COVERAGE} - k \times (\text{LENGTH} - \text{COVERAGE})$$

The calculation of (LENGTH − COVERAGE) is the count of residues in the initial match region that are not part of interval matches identified in the coarse search phase. Typically, we choose a constant k where $k \ll 1$, since interval matches that contribute to the COVERAGE score are indicators of possibly homology, while regions containing substitutions or less than interval-length identities are not the opposite, that is, regions not containing interval matches may still have high similarity through conservative substitutions; we have tested values of k through empirical observation of ranked results and we use a value of k optimised for amino-acid searching in Section 4.

By using this approach, frames containing interval identities spanning a reasonable portion of sequences being compared are ranked highly. If the result of the calculation is negative, where the matches are sparse over a long region, then the matching regions are recursively divided and treated independently until each portion is positive. We show in Section 4 that the COMBINED scheme is excellent for coarse ranking with FRAMES and we have found it has better accuracy than either LENGTH or COVERAGE.

3.3 Applications of Frames

In our discussion of FRAMES so far, we have neglected variation in the scores of each matching interval. In most nucleotide sequence searching, this assumption has no impact; an identity between two nucleotide bases is usually scored as $+5$, so a matching interval of length n will always score $5n$. However, in matrix-based amino-acid or more complex nucleotide comparison [43], this scoring scheme is only partly valid, since each matching base or residue may contribute a different score depending on frequency and the evolutionary likelihood of mutation. FASTA partially addresses this problem by initially scoring interval matches using constant match weights and then later re-scoring the matches using a matrix.

In amino-acid searching, it is likely that ranking using the COVERAGE or LENGTH metrics can be improved by considering the score of a partial alignment. In the case of COVERAGE, incorporating an accurate model of scoring is possible, since the count of residues contributed by an interval can be replaced by the score contributed by the alignment of the residues in the interval, where scores are usually calculated using a mutation matrix. Moreover, we can ensure by using FRAMES that each residue identity is only incorporated in the overall score once in each frame. Further, at an in-memory space cost of one byte per interval in the search structure, pre-calculated scores can be stored in the search structure to reduce the computational time-cost of scoring matching intervals. We incorporate this pre-calculation of scores during initialisation of each CAFE query and augment the COVERAGE scheme to use scores calculated from the appropriate matrix, rather than counts of matching bases or residues.

For LENGTH, the relationship between the length of a matching region and the potential score is not linear or, indeed, easily calculated. The composition of a sequence between any two matching intervals affects the possible score. For example, a query may contain a short region of amino-acids that can achieve a higher identity score than a much longer region of lower-scoring identities. When using the Dayhoff PAM-250 scoring matrix [16] an identity match between two tryptophan (W) amino-acid residues scores 17, while an identity between two alanine residues (A) scores 2, indicating that a single W identity alignment is roughly equivalent to eight matching A residues as an indicator of likely homology. In contrast, an alignment of tryptophan with any other amino-acid results in an average score of -4.1, while an alignment of alanine with another amino-acid has an average score of -0.8.

Composition and alignment between matching intervals in the LENGTH metric is difficult to model without fine-searching the region. We therefore elect not to modify the LENGTH scheme for complex scoring models, rather we apply a statistical normalisation to incorporate variations in composition in the resultant score.

Karlin and Altschul [22] propose addressing the problem of amino-acid sequence composition through normalising similarity scores based on query composition, scoring matrix used, and sequence length. However, Shpaer et al. [41] have shown in large-scale empirical retrieval effectiveness tests—but not theoretically—that a measure that is equally effective as the Karlin and Altschul scheme is

$$S_{norm} = \frac{S \times 21.21}{\ln l_1 \times \ln l_2}$$

where S_{norm} is the resultant normalised score, S is the original score, and l_1 and l_2 are the lengths of the two aligned regions. In the case where both regions are 100 residues in length, the normalisation has no effect, since $\ln(100) \times \ln(100) \approx 21.21$.

This approach to normalisation has significant benefit, as it does not require the calculation of the query or comparison matrix composition parameters.

Given the improved retrieval effectiveness demonstrated by Shpaer et al. in normalising similarity scores, and that the scheme proves more effective than that of Karlin and Altschul, we employ the Shpaer normalisation scheme in the LENGTH and COMBINED schemes. In addition, we can also use the scheme to normalise the COVERAGE scores, since query composition is not factored into normalisation. We describe experiments with the score-based and normalised COMBINED metric in Section 4.

3.4 Optimising Frames

As we show in Section 4, FRAMES is a successful method of accurate coarse ranking in the CAFE system. However, since each interval in a query requires the retrieval of an inverted list, and each entry in each inverted list either creates a new frame or adds weight to an existing frame, the in-memory and CPU overhead of FRAMES is often significant, particularly for longer queries. The overheads and problems of using *accumulators*, that is, structures similar to FRAMES that are used for ranking in English text retrieval are well-known [27, 54]. Such techniques used to minimise problems in English text retrieval have direct applicability to genomic database searching.

A simple heuristic to reduce the overhead of FRAMES is to apply a fixed ceiling on the number of frames that can be created for any given query. Assuming that query intervals are sorted, the threshold can be chosen so that most intervals that discriminate well between sequences will be processed prior to the threshold being reached; a sorting order for query intervals requires careful consideration. Further, assuming a reasonable threshold, it may be possible to process all intervals for short queries or where some intervals are filtered prior to querying [49].

Once the threshold of FRAMES is reached, there are two possible options. First, we can cease processing query intervals and present coarse rankings immediately; this has the benefit of reducing coarse search time. Second, the approach favoured in English text retrieval, and the one we have chosen, is to continue to adjust the scores of existing frames by processing all query intervals and adding new offset tuples for matching intervals to the existing frames. This second method has the disadvantage of not reducing evaluation time significantly, however we have found that the retrieval effectiveness is considerably better than with the first approach. This effect is similar to that found in English text retrieval [54].

To detect likely indels and relationships between FRAMES, we propose a *neighbourhood* algorithm that passes through the frame table at the conclusion of coarse searching adding to the frame score $s^{1/d}$, where s is the score of a frame from the same sequence and d is the difference in offset of the related frame; we have also tried other neighbourhood weighting schemes but do not present detail here. By using a sliding window of a maximum size, this scheme can be bounded and requires only a single pass of the frame table. We refer to this scheme as NEIGHBOURHOOD and present results in the next section.

Our fine search scheme, which is described in detail elsewhere [48], uses the FRAMES structure and the related NEIGHBOURHOOD frames as a basis for efficient, optimised gapped local alignment. FRAMES are used in two ways in fine searching: first, the regions identified as matches in coarse searching are used as the basis of subsequent fine searching and not recomputed, saving alignment costs and allowing fine searching to begin by evaluating alignments that are likely to be high-scoring; second, FRAMES permit partial sequence retrieval, where only the region of a database sequence that can be aligned to give a positive score is retrieved from the database (this is beneficial when, for example, a query of a few hundred bases has a coarse search match with a database sequence of several hundred thousand bases).

4 Results

Test Data

The derivation of suitable test collections, as has been noted by Barton [10], is difficult. There are, however, several possible approaches to forming test databases. In our experiments, because of limitations in forming suitable collections, we use the PIR database to assess the accuracy of search systems and the much larger GenBank database to assess speed and CAFE index space requirements.

To explore comparative retrieval effectiveness between search systems in large-scale searching, we use a subset of the PIR database similar to that first used by Pearson [31] to evaluate the FASTA tool. A more recent variation used by Shpaer et al. [41] has been used to evaluate the effectiveness of a broad range of tools, in particular an implementation of local alignment [42] in hardware.

The PIR database consists of four smaller databases, PIR1 through PIR4. Each amino-acid sequence is stored in one of the smaller databases, depending on the state of classification and annotation. Of interest in forming a characterised collection for assessing retrieval effectiveness are sequences stored in PIR1 and PIR2.

Sequences from the PIR1 and PIR2 subcollections of the PIR database used in our experiments[1] have been fully classified and annotated and are generally assigned a sequential *super family* (SF) number. Of the 85,618 sequences (13,583 sequences in PIR1 and 72,035 sequences in PIR2), 38,224 have been assigned one of the 3,781 SF numbers; most sequences in PIR2 are not classified by SF number but by SF name, by domain name, or are annotated but not classified. The classification process is described in detail elsewhere [9].

Broadly using the guidelines of Shpaer et al. [41], we used PIR1 and PIR2 to compile a PIR test collection. We extracted all sequences that had an assigned SF number, that is, all of PIR1 and around one-third of PIR2, to form a database of 38,224 sequences. The total size of the database was 12.7×10^6 residues, with a mean sequence length of 328. We refer to this collection as PIRSF.

In compiling a query set for retrieval effectiveness assessments using PIRSF, we used the first-occurring member sequence in the database from each SF. However, we did not use as queries the 1,175 SFs represented that have only one member; searching for a query sequence in the database does not illustrate the detection of homology. Feedback from users of homology search tools suggests that long amino-acid queries are highly infrequently posed, and we have removed any query longer than 500 residues. With the removal of single-member SF queries and long queries, our query set was reduced to 1,834 sequences.

The resultant query set and test collection has several advantages and disadvantages. The primary advantage is that retrieval effectiveness for each query can be measured by assessing the ranking of other sequences that are members of the SF and the success of the technique in discriminating between SF and non-SF sequences. A disadvantage of using the PIR databases is the rigid classification of sequences into SFs. Although a minor concern is the erroneous classification of a few sequences into incorrect SFs, a more major consideration is that SFs are sometimes closely related. To illustrate, the complex protein domain “HisI bifunctional enzyme” contains two simple protein domains, “HisI protein” and “Histidinol dehydrogenase”. This complex domain, along with a third simple domain, is present in the sequence “HisI-hisD trifunctional enzyme”, which is a member of only SF

[1] Release 52.0, 31 March 1997.

635.0. The same complex domain, without the third simple domain, is also present in SF 636.0. Many sequences, classified in different SFs, may contain subsets of the simple protein domains, combined with other simple protein domains typical of other super families.

Detection of false positives through a homology search on the PIRSF collection for a given query may sometimes be an example of sensitive detection of weak inter-family similarities. This may result in penalising one search technique that is in fact more sensitive than another. While some inaccuracy in measurement is likely, we believe that over the 1,834 queries, the PIRSF search results give a good approximation of the relative performance of different search methods.

To quantify the relative performance or retrieval effectiveness of search techniques, we use the measures of recall and precision. Recall and precision are frequently used to demonstrate the retrieval effectiveness of systems, particularly those used for English text retrieval; the application of these measures to text retrieval is described elsewhere [39].

Precision is a measure of the fraction of relevant answers retrieved at a particular point, that is

$$P = \frac{\text{SF sequences retrieved}}{\text{SF and non-SF answers retrieved}}$$

Recall, in contrast, measures the fraction of the relevant answers that have been retrieved at a particular point, or

$$R = \frac{\text{SF sequences retrieved}}{\text{Total size of SF}}$$

In most practical applications, the assessment of recall is impractical, since it is not feasible to assess each record in the database for relevance to each query. However, by using our PIRSF approach it is possible to approximate recall by considering only family members as relevant answers and non-family members as irrelevant answers. This somewhat restrictive assumption allows the practical calculation of recall values.

We expect that an exhaustive approach would have comparable search times to CAFE for smaller databases such as PIRSF. However, for larger collections, we would expect that CAFE would be relatively much faster. While both exhaustive approaches and CAFE should experience linear increases in search times with increasing database size, CAFE search time growth will have smaller constant factors that will be less significant in the context of searching large databases.

To compare the speed of systems, we used nucleotide collections of different sizes derived from the GenBank database.[2] The GenBank 97.0 collection has 652 million nucleotide bases, in 1,021,211 sequences, with an average length of around 638. This complete past version of GenBank database is referred to as GENBANK97 throughout. The GenBank 108.0 collection has 1,797 million nucleotide bases, in 2,532,359 sequences, with an average length of around 707; we refer to this collection throughout as GENBANK108.

The third collection used, referred to throughout as VERTE, was around one third of the GENBANK97 database, and contained the mammalian, primate, rodent, and miscellaneous vertebrate and invertebrate databases. It contained approximately 177 million nucleotide bases in 121,623 sequences, with an average sequence length of 1,456 bases. We chose this collection as it contains known homologues to some of the query sequences we used, as well as a range of organisms and sequence types; we use this collection both for speed comparison and for simple comparison of the results sets obtained with different search tools.

[2] Flat-file, releases 97.0 (15 October 1996) and 108.0 (15 August 1998). The most recent release 110.0 (15 December 1998) has 2,162 million bases.

Collection	Inverted lists (Mb)	Other structures (Mb)
GBMAM	25.4	3.3
VERTE	427.7	3.3
GENBANK97	1420.4	3.3
GENBANK108	3682.6	3.3
PIRSF	23.6	0.1

Table 1: CAFE *index size for the nucleotide collections,* GENBANK108*,* GENBANK97*,* VERTE*, and* GBMAM *and the amino-acid collection* PIRSF*. For nucleotide searching,* CAFE *has an interval length of* $n = 9$ *and, for amino-acid searching, an interval length of* $n = 3$*.*

A third small collection was used for comparison of system speed only and is a subsection of the VERTE collection. The collection was derived from the "other mammalian sequence entry" database and is referred to throughout as GBMAM. The database contains just over 10 million nucleotide bases in 9,938 sequences with an average length of 1,016 bases. Again, we chose this database for known similarity to our query set and for the wide variation of mammals represented, with the exception of primates.

We do not use nucleotide databases extensively for the assessment of retrieval effectiveness, rather, we use them for the comparison of retrieval speed. However, we have collected a query set from a variety of sources to at least evaluate responses from the CAFE approach and from exhaustive search tools. Some were sequences obtained from users who work regularly with GenBank. Other queries were sequences in GenBank that are known to have both close and distant matching sequences. In total, there were 41 queries. The average query length was around 488 bases, the shortest was 134, and the longest was 1,382. All experiments are carried out on a recent Intel Pentium II dual processor machine running the Linux operating system under light load, where the machine is otherwise largely idle.

4.1 Space

We show in Table 1 the CAFE index sizes for the nucleotide GenBank collections, and for the amino-acid PIRSF collection. An interval length of $n = 9$ is used in all nucleotide collections and $n = 3$ used in PIRSF. The sequence identifiers and interval offsets in the CAFE inverted lists are compressed using parameterised Golomb codes [21], with a local Bernoulli model used for selecting the parameter k [54]. Elias gamma codes [18] are used to store in inverted lists the count of offsets in each sequence. The search structure files largely consist of the distinct indexed intervals. We have tried other interval lengths but have found that $n = 9$ in nucleotide collections and $n = 3$ in amino-acid collections are reasonable in size, support fast searching, and suitable for accurately identifying answers [48].

Compared to the GENBANK108 and GENBANK97 collections, the CAFE indexes are 2.2 times the size of the collections, while the corresponding VERTE index is 2.5 times the size of VERTE, and the index for PIRSF is 1.9 times the collection size.

Without incorporating our special-purpose CAFE compression scheme, the uncompressed fine-grain inverted list size of PIRSF is 133.3 Mb, or around 5.6 times the size of the compressed inverted lists and over 10 times the size of the collection. Similarly, the uncompressed fine-grain inverted list size for GENBANK97 is 6105 Mb, around 3.9 times the compressed size and 9.7 times the collection size. While we have not carried out speed evaluations of uncompressed indexes, it is likely that

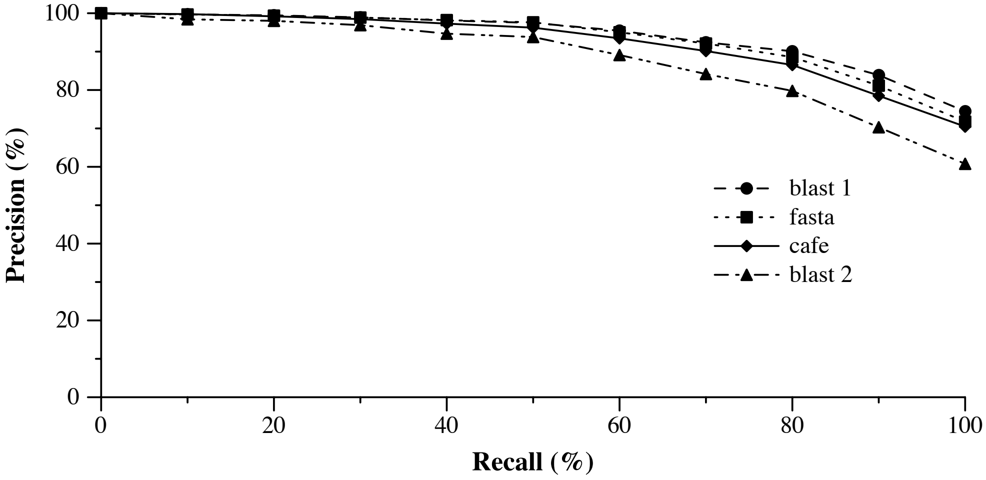

Figure 8: *Mean recall-precision for both versions of* BLAST*,* FASTA*, and* CAFE*. The* PIRSF *collection is used, with 1,834 amino-acid queries. We parameterise* CAFE *to use a banded local alignment fine search (similar to that used in* FASTA*) and the* NEIGHBOURHOOD *frames ranking metric. All systems use a PAM-250 matrix scoring matrix.*

CAFE query evaluation costs would be significantly slower through the increase in disk retrieval associated with retrieving large, uncompressed lists; we have shown elsewhere that, for example, the retrieval of uncompressed nucleotide data reduces the speed of our CAFE system [51].

Note that the GenBank database contains a great deal of information, such as bibliographic data and other annotations, in addition to the sequences themselves. The latest release of GenBank is almost exactly 10 Gb in size, and would require a 4 Gb CAFE index. The size of the CAFE indexes should be viewed in the context of the whole database, not just the indexed sequences.

4.2 Retrieval Effectiveness

Figure 8 shows a mean recall-precision graph for both versions of BLAST, FASTA, and CAFE. Results are shown for searching the PIRSF collection with our 1,834 sequence query test set. For parameters, CAFE uses a gapped local alignment scheme for fine searching (which is similar to that used in FASTA), an interval length of $n = 3$, a fixed maximum number of frames of $f = 110{,}000$, a maximum number of fine searched frames of $m = 5{,}000$, a query filtering technique where any interval occuring in more than 10% of database sequences is ignored in coarse searching, and the score-optimised COMBINED ranking scheme with NEIGHBOURHOOD.

Our results show that CAFE searching is only marginally less accurate than BLAST 1 and FASTA, and that BLAST 2 is 1%–2% lower in precision at low recall levels and has up to 10% lower precision at higher recall levels. Importantly, CAFE searching has both similar underlying heuristics and performance to FASTA, which others have shown is the most accurate rapid homology search system [31, 41]. Indeed, analysis of results suggests that CAFE and FASTA both accurately identify close, moderate, and distant evolutionary relationships, including distant superfamily (SF) similarities in the PIRSF test data. As discussed earlier, because of the limitations of PIRSF, we believe that more accurate retrieval effectiveness measurement is at lower recall levels; while BLAST 1 appears to have better performance at higher recall levels, FASTA performs as well at lower recall levels and it is likely

Average	BLAST 1	BLAST 2	CAFE	FASTA
11-Point	93%	88%	92%	92%
3-Point	95%	91%	94%	95%

Table 2: *Mean three-point and eleven-point recall-precision comparison of both versions of* BLAST*,* FASTA*, and* CAFE*.*

	BLAST 1	BLAST 2	CAFE	FASTA
GBMAM	0.6	0.4	1.2	8.0
VERTE	6.8	7.4	3.2	108.4
GENBANK97	67.1	19.2	9.2	823.0
GENBANK108	192.5	182.5	20.2	—

Table 3: *Mean elapsed time for 41 nucleotide queries (average seconds per query) on the* GENBANK108*,* GENBANK97*,* VERTE*, and* GBMAM *collections, using both versions of* BLAST*,* FASTA*, and* CAFE*. It was not possible to experiment with* FASTA *on* GENBANK108 *as the* GENBANK108 *uncompressed file size exceeds operating system file system limits.*

that given better sequence classification that FASTA would be better at higher recall levels also.

These results suggest that CAFE is almost as effective as FASTA in finding homologous gapped alignments at lower recall levels, being particularly sensitive in ranking highly both closely and moderately related sequences. In ranking distantly related sequences, CAFE may be slightly less effective, however it is unlikely that a difference of 1%–2% would be significant to users.

Often, a three-point and eleven-point average precision are calculated as standardised measures of retrieval effectiveness. The three-point precision averages precision at recall levels of 20%, 50%, and 80%; this gives a useful insight in the case of the PIRSF collection, since most queries result in precision values of 100% at levels of 10%–20% recall and often do not result in 100% recall. Mean three-point and eleven-point precision results, as shown in Table 2, also show the closeness in performance of CAFE to FASTA and BLAST 1: results are either identical or within 1%. In contrast, the three-point and eleven-point of BLAST 2 are 4%–5% lower than other systems, suggesting that the speed improvements in BLAST 2 compared to BLAST 1 (which we report in the next section) are at the detriment of search accuracy.

4.3 Speed

We show in Table 3 the relative speeds of both versions of BLAST, FASTA, and CAFE in searching the genomic nucleotide collections with our small 41-query test set. These results are plotted in Figure 9. The parameters used in all searches are the same as in Section 4.2.

Searching with CAFE can be over eighty times faster than with FASTA and eight times faster than with BLAST 2. CAFE has high constant costs, due to the necessity to initialise and search the FRAMES structures. These costs are reflected in times for GBMAM and, to a lesser extent, VERTE. As the data set size increases, CAFE times rise reasonably slowly, with a doubling in data size leading to around a 90% increase in time.

In contrast, FASTA and BLAST have smaller initial costs, but search times are linear in database size. (For BLAST 2 the times reported in Table 3 are distorted by a single difficult query that, while not especially long, contains large numbers

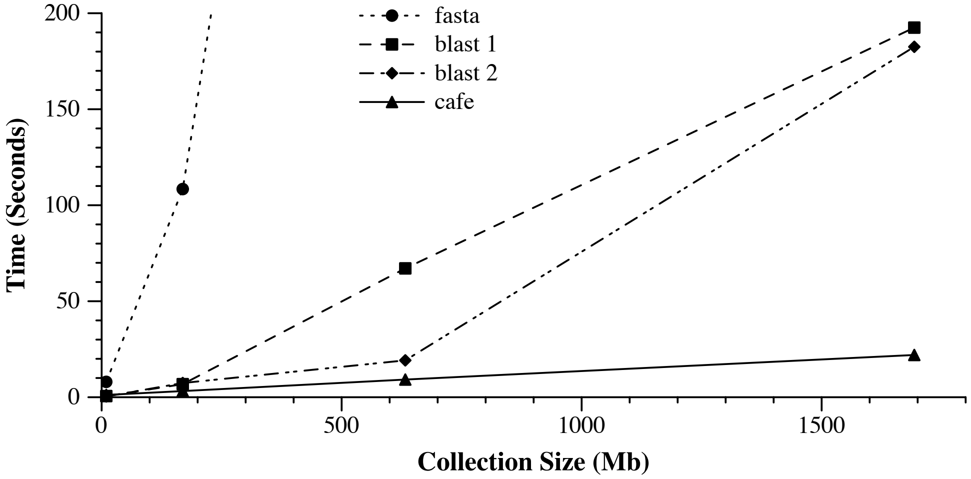

Figure 9: *Plot of the time for* BLAST*,* CAFE*, and* FASTA *to search the nucleotide collections* GBMAM*,* VERTE*,* GENBANK97*, and* GENBANK108*, averaged over 41 queries.*

of repetitive intervals that are in most database sequences. Without this query, for the smaller collections the correlation between size and time is extremely close.)

However, these algorithms are highly reliant on having sufficient memory to store the complete database. When database size exceeds available memory, times increase steeply; indeed, the BLAST documentation states that BLAST should only be run on machines with sufficient main memory to hold the complete stored data. For BLAST 2, in which data is stored compressed, this transition occurs between GENBANK97 and GENBANK108. For BLAST 1 this transition occurs between VERTE and GENBANK97. For the most recent release of GenBank, BLAST 2 would require over 600 Mb of main memory. In contrast, CAFE would only require around 100 Mb, while for the smaller GENBANK97 about 70 Mb is required. That is, while the memory requirements of BLAST scale linearly with database size, those of CAFE grow very slowly.

In CAFE evaluation times depend on query length and on the statistics of the intervals in the query; intervals with longer inverted lists require more processing. On GENBANK108 the shortest query (of 58 bases) requires only 7.4 seconds, compared to the average of 20.2 seconds. The longest query requires twice the average time. As for RAMDB extremely short queries can be run very quickly indeed. For BLAST query length is less important with the shortest query requiring 174.9 seconds, little different from the average.

Although we do not present detailed results here for searching the small PIRSF collection, CAFE times are similar to those of BLAST 2, but around 1.5 times faster than BLAST 1, and over 2.2 times faster than FASTA; as we have shown for nucleotide searching, CAFE is likely to become relatively much faster with increasing collection size.

5 Conclusions

There is a growing need for fast, efficient searching of genomic databases. Current databases contain billions of nucleotide bases and are queried tens of thousands of times per day. These databases are growing rapidly, and, as they become increasingly complete and cover increasing numbers of species, will reach trillions of nucleotide bases within the next few years.

In this paper, we have shown that genomic query evaluation using the CAFE indexed scheme and appropriate data structures affords much faster query evaluation than exhaustive searching, with better accuracy than the most commonly used search tool, BLAST 2. This successful application of indexing to genomic data shows that, as for other search applications, indexing becomes preferable to exhaustive search once data set size is sufficiently large. The innovations that allow CAFE to be faster than previous techniques are the incorporation of compression for data and indexes; the use of FRAMES, a novel data structure for identifying and managing likely homologous sequences; partial sequence retrieval of longer sequences to minimise local alignment costs; and new local alignment techniques based on FRAMES. In the process of developing CAFE we found that each of these innovations was essential to good performance.

In contrast to previous search techniques, CAFE is fast and has low memory requirements but depends on large indexes. We believe that this trade-off is necessary for searching of large genomic databases. Moreover, these indexes are smaller than the annotated source databases, and are much smaller than the indexes of previous indexed systems such as FLASH. Overall, we are confident that indexed systems such as CAFE will be the only practical option for searching the vast quantities of genomic data that will soon be available.

Acknowledgments

We thank Craig Primmer from the University of Helsinki, Finland for his ongoing involvement in the CAFE project. This work was supported by the Australian Research Council and the Multimedia Database Systems group at RMIT University.

Availability

The index-based gapped search system (CAFE) is available free of charge by request to the authors.

References

[1] S. Altschul, M. Boguski, W. Gish, and J. Wootton. Issues in searching molecular sequence databases. *Nature Genetics*, 6:119–129, 1994.

[2] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[3] S.F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *Journal of Molecular Biology*, 219:555–565, 1991.

[4] S.F. Altschul. A protein alignment scoring system sensitive at all evolutionary distances. *Journal of Molecular Evolution*, 36:290–300, 1993.

[5] S.F. Altschul and W. Gish. Local alignment statistics. *Methods in Enzymology*, 266:460–480, 1996.

[6] S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[7] I. Anderson and A. Brass. Searching DNA databases for similarities to DNA sequences: when is a match significant? *Bioinformatics*, 14(4):349–356, 1998.

[8] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence data bank and its new supplement TrEMBL. *Nucleic Acids Research*, 24:21–25, 1996.

[9] W.C. Barker, F. Pfeiffer, and D.C. George. Superfamily classification in PIR-International protein sequence database. *Methods in Enzymology*, 266:59–71, 1996.

[10] G.J. Barton. Protein sequence alignment and database scanning. In M. J. E. Sternberg, editor, *Protein Structure Prediction: A Practical Approach.* IRL Press at Oxford University Press, 1996.

[11] D.A. Benson, M.S. Boguski, D.J. Lipman, J. Ostell, and B.F. Ouellette. GenBank. *Nucleic Acids Research*, 26(1):1–7, 1998.

[12] A. Califano and I. Rigoutsos. FLASH: A fast look-up algorithm for string homology. In *International Conference on Intelligent Systems for Molecular Biology*, pages 56–64, Bethesda, MD, 1993.

[13] K-M. Chao, W.R. Pearson, and W. Miller. Aligning two sequences within a specified diagonal band. *Computer Applications in the Biosciences*, 8(5):481–487, 1992.

[14] F. Collins and D. Galas. A new five-year plan for the US human genome project. *Science*, 262:43–46, 1993.

[15] F.S. Collins, A. Patrinos, E. Jordan, A. Chakravarti, R. Gesteland, and L. Walters. New goals for the U.S. human genome project: 1998-2003. *Science*, 282(5389):682–689, 1998.

[16] M.O. Dayhoff. *Atlas of Protein Sequence and Structure*. National Biomedical Research Foundation, Washington, D.C., 1978.

[17] R.F. Doolittle. *Of URFs and ORFs*. University Science Books, Mill Valley, CA, 1986.

[18] P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, IT-21(2):194–203, March 1975.

[19] C. Fondrat and P. Dessen. A rapid access motif database (RAMdb) with a search algorithm for the retrieval patterns in nucleic acids or protein databanks. *Computer Applications in the Biosciences*, 11(3):273–279, 1995.

[20] D. George, W. Barker, H. Mewes, F. Pfeiffer, and A. Tsugita. The PIR-international protein sequence database. *Nucleic Acids Research*, 24:17–20, 1996.

[21] S.W. Golomb. Run-length encodings. *IEEE Transactions on Information Theory*, IT–12(3):399–401, July 1966.

[22] S. Karlin and S.F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. National Academy of Sciences USA*, 87:2264–2268, 1990.

[23] T.G. Larson and S.H. Goodgal. Sequence and transcriptional regulation of com101a, a locus required for genetic transformation in haemophilus influenzae. *Journal of Bacteriology*, 173:4683–4691, 1991.

[24] C. Liébecq, editor. *Biochemical Nomenclature and Related Documents*, pages 122–126. Portland Press, London, 2nd edition, 1992.

[25] D.J. Lipman and W.R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227:1435–1441, 1985.

[26] S. McGinnis. Personal communication. (GenBank user services, National Centre for Biotechnology Information (NCBI), National Library of Medicine, US National Institute of Health), January 1998.

[27] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *ACM Transactions on Information Systems*, 14(4):349–379, October 1996.

[28] E.W. Myers. Advances in sequence assembly. In C. Venter, editor, *Automated DNA Sequencing and Analysis Techniques*, pages 231–238. Academic Press, 1994.

[29] B.C. Orcutt and W.C. Barker. Searching the protein database. *Bulletin of Mathematical Biology*, 46:545–552, 1984.

[30] W.R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods in Enzymology*, 183:63–98, 1990.

[31] W.R. Pearson. Comparison of methods for searching protein-sequence databases. *Protein Science*, pages 1145–1160, 1995.

[32] W.R. Pearson. Protein sequence comparison and protein evolution. In *International Conference on Intelligent Systems for Molecular Biology*, Cambridge, UK, 1995. (tutorial).

[33] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proc. National Academy of Sciences USA*, 85:2444–2448, 1988.

[34] W.R. Pearson and W. Miller. Dynamic programming algorithms for biological sequence comparison. *Methods in Enzymology*, 210:575–601, 1992.

[35] G. Pesole, N. Prunella, S. Liuni, M. Attimonelli, and C. Saccone. WORDUP: an efficient algorithm for discovering statistically significant patterns in DNA sequences. *Nucleic Acids Research*, 20(11):2871–2875, 1992.

[36] C.M. Rice, R. Fuchs, D.G. Higgins, P.J. Stoehr, and G.N. Cameron. The EMBL data library. *Nucleic Acids Research*, 21:2967–2971, 1993.

[37] E. Rivals, O. Delgrange, J.-P. Delahaye, M. Dauchet, M.-O. Delorme, A. Hénaut, and E. Ollivier. Detection of significant patterns by compression algorithms: the case of approximate tandem repeats in DNA sequences. *Computer Applications in the Biosciences*, 13(2):131–136, 1997.

[38] G. Salton. *Automatic Text Processing*. Addison Wesley, Massachusetts, 1989.

[39] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[40] D. Sankoff and J.B. Kruskal, editors. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, Massachusetts, 1983.

[41] E.G. Shpaer, M. Robinson, D. Yee, J.D. Candlin, R. Mines, and T. Hunkapiller. Sensitivity and selectivity in protein similarity searches: A comparison of Smith-Waterman in hardware to BLAST and FASTA. *Genomics*, 38:179–191, 1996.

[42] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[43] D.J. States, W. Gish, and S.F. Altschul. Improved sensitivity in nucleic acid database searches using application-specific scoring matrices. *Methods: A Companion to Methods in Enzymology*, 3(1):66–70, 1991.

[44] M.S. Urdea, J.P. Merryweather, G.T. Mullenbach, D. Coit, U. Heberlein, P. Valenzuela, and P.J. Barr. Chemical synthesis of a gene for human epidermal growth factor urogastrone and its expression in yeast. *Proceedings of the National Academy of Science*, 80(24):7461–7465, 1983.

[45] M.S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman and Hall, London, 1995.

[46] W.J. Wilbur and D.J. Lipman. Rapid similarity searches of nucleic acid and protein data banks. *Proceedings of the National Academy of Science*, 80:726–730, 1983.

[47] H.E. Williams. Compressed indexing for genomic retrieval. *Journal of Mathematical Modelling and Scientific Computing*, 9(2):144–154, 1998.

[48] H.E. Williams. *Indexing and Retrieval for Genomic Databases*. PhD thesis, RMIT University, 1998.

[49] H.E. Williams. Effective query filtering for fast homology searching. In *Pacific Symposium on Biocomputing*, volume 4, pages 214–225, Hawaii, 1999.

[50] H.E. Williams and J. Zobel. Indexing nucleotide databases for fast query evaluation. In *Proc. International Conference on Advances in Database Technology (EDBT)*, pages 275–288, Avignon, France, March 1996. Springer-Verlag. Lecture Notes in Computer Science 1057.

[51] H.E. Williams and J. Zobel. Compression of nucleotide databases for fast searching. *Computer Applications in the Biosciences*, 13(5):549–554, 1997.

[52] H.E. Williams and J. Zobel. Compressing integers for fast file access. *Computer Journal*, 42(3):193–201, 1999.

[53] R.W. Williams. The portable dictionary of the mouse genome: a personal database for gene mapping and molecular biology. *Mammalian Genome*, 5:372–375, 1994.

[54] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, New York, 1994.

[55] F. Wolfertstetter, K. Frech, G. Herrmann, and T. Werner. Identification of functional elements in unaligned nucleic acid sequences by a novel tuple search algorithm. *Computer Applications in the Biosciences*, 12(1):71–80, 1996.

[56] J.C. Wootton and S. Federhen. Statistics of local complexity in amino acid sequences and sequence databases. *Computers in Chemistry*, 17:149–163, 1993.

[57] J. Zobel and P. Dart. Finding approximate matches in large lexicons. *Software Practice and Experience*, 25(3):331–345, March 1995.

[58] J. Zobel, A. Moffat, and R. Sacks-Davis. Searching large lexicons for partially specified terms using compressed inverted files. In *Proc. International Conference on Very Large Databases*, pages 290–301, Dublin, Ireland, 1993.