Pacific Symposium on Biocomputing 4:214-225 (1999)

EFFECTIVE QUERY FILTERING FOR FAST HOMOLOGY SEARCHING

HUGH E. WILLIAMS
Department of Computer Science, RMIT University,
GPO Box 2476V, Melbourne 3001, Australia
hugh@cs.rmit.edu.au

To improve the accuracy of rapid homology searching it is common practice to filter all queries to mask low complexity regions prior to searching. We show in this paper, through a large-scale study of querying the PIR database, that applying popular filtering techniques unselectively to all queries may reduce retrieval effectiveness. We also show that masking queries with our new technique, cafefilter, which uses the overall distribution of motifs in a database, is at least as effective as current popular query filtering tools in large-scale tests.

1 Introduction

Most tools for fast searching of genomic databases use well-known string matching techniques to compare a query sequence to each sequence in a genomic database. An effective genomic sequence search may identify homologies between a query sequence and database sequences, lending support to a hypothesis as to the function or structure of the query sequence. To indicate the likelihood of the results of a search being indicative of homology, answers are typically ranked in decreasing order of local similarity to the query.

To quantify similarity between sequences, considerable research has been dedicated to the sensitivity and selectivity of algorithms through the application of statistical theory. Such statistical similarity theory is based on the model that the query and database sequences are drawn from identical distributions of amino-acid or nucleotide residues. However, this assumption is frequently invalid as genomic nucleotide sequences often contain microsatellites and other approximate tandem repeats, while amino-acid sequences are often rich in proline, glycine, or other such amino-acids. Such local regions of repetitive or periodic intervals are generally referred to as being regions of *low complexity*.

Queries containing low complexity regions are often difficult to evaluate using rapid homology search systems. Indeed, the analysis of search results with queries containing low complexity regions shows skewing in results where long, unrelated sequences are ranked higher than short, homologous sequences. Moreover, evaluating similarity measures for highly frequent, low complexity regions is computationally intensive, since most stored sequences contain such

regions and each matching region requires a computationally intensive heuristic local alignment. Case studies of individual queries, where regions of low compositional complexity are removed or masked, have shown benefits in reducing query evaluation costs and improving effectiveness in retrieving relevant answers.

In this paper we evaluate the effectiveness of general-purpose query filtering schemes for searching amino-acid databases and propose a new approach, CAFEFILTER. We show that, surprisingly, filtering a large query set using the popular query filtering tools SEG and XNU reduces the overall effectiveness of finding answers in a collection derived from the well-classified PIR protein database. Our novel approach, CAFEFILTER, uses overall collection frequencies to derive a set of common motifs that can be used to mask or filter queries prior to general-purpose searching. The advantage of using overall collection frequencies is that frequent motifs that are common to both homologous and unrelated sequences are filtered. CAFEFILTER again works well in individual case studies but, similarly to SEG and XNU, reduces the effectiveness of searching with our large query set.

Our results suggest that the common practice of filtering all queries prior to searching often reduces the effectiveness of searching. Indeed, we have found that the default filtering using SEG in the new release of the BLAST search system frequently reduces accuracy. Our conclusion is that query filtering should be used selectively on a query-by-query basis after evaluating and assessing the results of a search with an unfiltered query sequence.

2 Techniques for Filtering Queries

Rapid homology searching is used to select and rank database sequences by the similarity of *high complexity regions* to a given query. By identifying similar regions between a query and database sequence, it is often possible to identify homology and lend evidence to a hypothesis as to the chemical structure, biochemical role, or evolutionary relationship of a query sequence. At present, over 20,000 such homology searches each day are processed by the BLAST [1,2] servers at the US National Centre for Biotechnology Information (NCBI) [3].

As the result of a homology search with a query on a genomic database, a set of responses is returned ranked in decreasing order of statistical similarity to the query sequence. The significance of alignments between high complexity regions is typically estimated using statistical theory based on the assumption that the query and answer sequences are drawn from the same distribution of residues [4,5,6]. However, the significance of amino-acid sequence alignments of *low complexity regions* which are, for example, glutamine-rich, glycine-rich,

or proline-arginine-rich are poorly estimated by such statistics. Indeed, it is frequently suggested that the alignment of these repetitive low complexity regions skews query results and that alignment by position of low complexity regions does not aid in identifying homologues [7,8,9,10].

Several tools have been proposed to improve the retrieval effectiveness of rapid homology searches by masking or filtering low complexity regions in query or database sequences. Such approaches to filtering include SEG [10,11], XNU [12], SIMPLE34 [13], CENSOR [14], and SAPS [15]. All methods except SAPS filter nucleotide queries, while XNU, SEG, and SAPS also filter amino-acid sequences. In general, masking low complexity regions prior to searching replaces the regions with IUPAC-ISBMB wildcard characters [16] such as N in nucleotide sequences or X is amino-acid sequences; N represents a valid substitution of any nucleotide and X of any amino-acid residue.

We discuss in the next section the existing approaches to detecting redundancy as a property of a single sequence. In Section 3 we discuss an existing approach to redundancy detection as the property of a collection and our new approach, CAFEFILTER.

2.1 Detecting Low Complexity in Individual Sequences

To detect low complexity regions within a single sequence a *dot-plot* is often made of a sequence and the plot visually inspected. A dot-plot is a matrix where the sequence is listed in the first row and the first column of the matrix and, if two residues are identical for a particular cell, a dot is plotted. Figure 1 shows a dot-plot of the around 500 residue protein sequence from GenBank "U12707—Human Wiskott-Aldritch Syndrome Protein (WASP)". Two repetitive, low complexity regions can be seen in the figure as significant initial match regions, or *blotches*, at offsets $x, y \approx 172 \ldots 197$ and $x, y \approx 340 \ldots 435$. Blotches typically consist of short-period repeats, that is, areas that may be inferred to be of low complexity.

To translate the process of visually inspecting a dot-plot to an automated process of detecting redundancy, Claverie and States [12] propose a tool, XNU, to mask short-period repeats in amino-acid and nucleotide sequences. XNU removes low complexity regions through scoring the self-alignment of a sequence with a computationally intensive local alignment technique commonly used for homology searching [17]. XNU identifies regions of high-scoring similarity and outputs the query with such regions in a masked format for subsequent use in searching. By replacing each residue with a masking wildcard character, masked sequence regions are neglected in subsequent rapid homology searching.

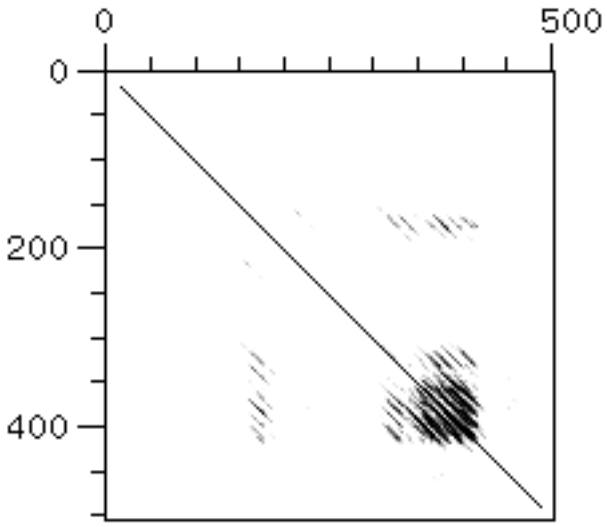

Figure 1: Self-alignment of GenBank accession U12707, "Human Wiskott-Aldritch Syndrome Protein (WASP)" using DOTTER. Redundant regions typically appear as "blotches", since repetitive regions self-align well with small offsets. In particular, two redundant regions can be seen at offsets 172-197 and 340-435.

A complementary technique for removing redundancy through content-based filtering of a sequence is SEG [10,11], which proposes masking selected fixed-length overlapping subsequences or *intervals*. SEG locates vectors of low complexity, where each vector is a decreasing count of residue occurrences for a reasonable interval length; a low complexity vector has a small number of distinct residues. As an example, the nucleotide interval, for $n = 8$, ATATATGG has a complexity vector $(3, 3, 2, 0)$, since A and T occur 3 times, G twice, and C does not occur. The interval GCGCAAGC has the same complexity vector, $(3, 3, 2, 0)$, since the vector is ordered by decreasing frequency of occurrence and not by a fixed ordering of the nucleotides; the lowest complexity vector for $n = 8$ is $(8, 0, 0, 0)$. Two passes are used in SEG to assess vectors and identify candidate regions for masking.

An interesting question is "when should a query be filtered prior to searching?". The sensitive FASTA [18,19,20] search system assists in answering this question by evaluating whether search results are skewed by the presence of low

complexity regions in the query. It has been shown that results of searches where a query and answers are drawn from the same distribution of residues have an extreme value distribution when the frequency of each similarity score is plotted against the score [6]. FASTA produces a plot of the expected ideal extreme value distribution and a plot of the actual distribution of scores observed. By inspecting the FASTA plot, FASTA can be used to answer the question of whether a query should be filtered and re-evaluated.

FASTA, therefore, can be used to assess the reliability of answers to a given query. A query can be first evaluated and a visual assessment of the actual and expected distributions made. If the expected and actual distributions suggest the presence of low complexity regions, the query can be filtered and the query re-evaluated. In contrast, the current version 2.0 of BLAST [1] filters all queries by default using SEG.

In the next section we present details of a collection-frequency based approach to query filtering, which was used in a previous version of BLAST, and we propose a new approach, CAFEFILTER.

3 Stopping based on collection interval frequencies

Earlier versions ($1.x$) of BLAST [2], in addition to the Karlin-Altschul statistical techniques [6], optionally use a collection-frequency based approach to address the problem of skewed results from redundant queries. Collection-frequency techniques filter queries based on the distribution of fixed-length subsequences, or *intervals*, over the database, rather than filtering regions based on the distribution of residues in the query sequence. The advantage of a collection-frequency approach is that filtered intervals are common to most database sequences and therefore such intervals do not effectively discriminate between answers.

In BLAST $1.x$ nucleotide database searching, a table of intervals of length $n = 8$ occurring over 200 times in the database can be stored for nucleotide database searching [2]. An option then, when searching with BLAST $1.x$, is to mask likely low complexity query intervals by removing any that appear in the table. This can be used to filter terms from queries that are likely to return a significant number of false matches, a process referred to as *cleaning* the database. Additionally, both SEG and XNU are supported in BLAST $1.x$ as non-default command-line query filters.

With growth in genomic databases, the cleaning approach will reduce retrieval effectiveness. As, for example, GenBank increases in size, a fixed threshold of 200 occurrences implies that the number of intervals cleaned will increase at a proportional rate. Therefore, with an increase in database size is a propor-

tional increase in clean-list size, resulting in less sensitive searching, a potential decrease in matches, and a likely fall in retrieval effectiveness.

We propose a variation on the cleaning approach used in BLAST that addresses the problem of varying list size. Our approach, which we call CAFEFILTER, forms a *stop-list* of intervals to be masked in a query. We filter any interval occurring in more than x% of the sequences, where smaller values of x lead to larger stop-lists and fast query evaluation. This approach effectively removes from queries intervals that are unlikely candidate terms to discriminate between sequences relevant to a given query. Clearly, however, there is a point at which retrieval effectiveness is affected by not evaluating sufficient query intervals. CAFEFILTER forms part of the CAFE indexed genomic search system, which we have described in part elsewhere [21,22,23,24]. We detail experiments with CAFEFILTER in the next section.

4 Test Collection and Results

4.1 Test Collection

To explore comparative retrieval effectiveness between existing filtering approaches and CAFEFILTER in large-scale searching, we use a subset of the PIR database similar to that first used by Pearson [25] to evaluate the FASTA tool. A more recent variation used by Shpaer et al. [26] has been used to evaluate the effectiveness of a broad range of tools, in particular an implementation of Smith-Waterman local alignment [17] 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 [a] 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 [27].

Broadly using the guidelines of Shpaer et al. [26], 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

[a] Release 52.0, 31 March 1997.

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 extremely rare, 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 by a factor of around two 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 effect of a query filtering technique on ranking the 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 the fact 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" [27]. 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 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 a technique that is in fact more sensitive than another. However, we believe that, over the 1,834 queries, the PIRSF search results give a reasonable indication of the relative performance of different filtering techniques.

To quantify the relative performance or retrieval effectiveness of filtering 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 by Witten et al. [28].

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.

4.2 Results

Figure 2 shows the average precision at each of eleven recall levels for different query filtering techniques. We searched the PIRSF collection with our 1,834 query set and averaged the results for SEG, XNU, and CAFEFILTER. Each search uses the latest release 2.0 of the BLAST search system [1]. For comparison, we also show the recall-precision of version 1.4.8 of BLAST; we discuss these BLAST results later.

We show in Figure 2 CAFEFILTER with a stop-list containing all intervals of length $n = 3$ occurring in more than $x = 7\%$ of sequences; $n = 3$ is the default interval length in BLAST and FASTA for protein searching. The stop-list for $x = 7\%$ contains 1,004 intervals. We have tested other values of x, including $x = 3$ (3,946 intervals), $x = 5$ (2,095 intervals), $x = 9$ (428 intervals), $x = 10$ (270 intervals), and $x = 15$ (18 intervals). We do not present the results of other stop-list sizes here, however we have found in the individual assessment of results that $x = 7\%$ works well for both genomic nucleotide and amino-acid query and database filtering.

Surprisingly we have found that default query filtering reduces retrieval effectiveness. At low recall levels, that is, where the most statistically similar members of each SF are retrieved, precision with XNU and SEG is up to 2% worse than without query filtering. At higher recall levels, filtering with SEG reduces precision by more than 5%. Retrieval effectiveness with CAFEFILTER is better than that of XNU and SEG, but remains worse than effectiveness with no query filtering applied.

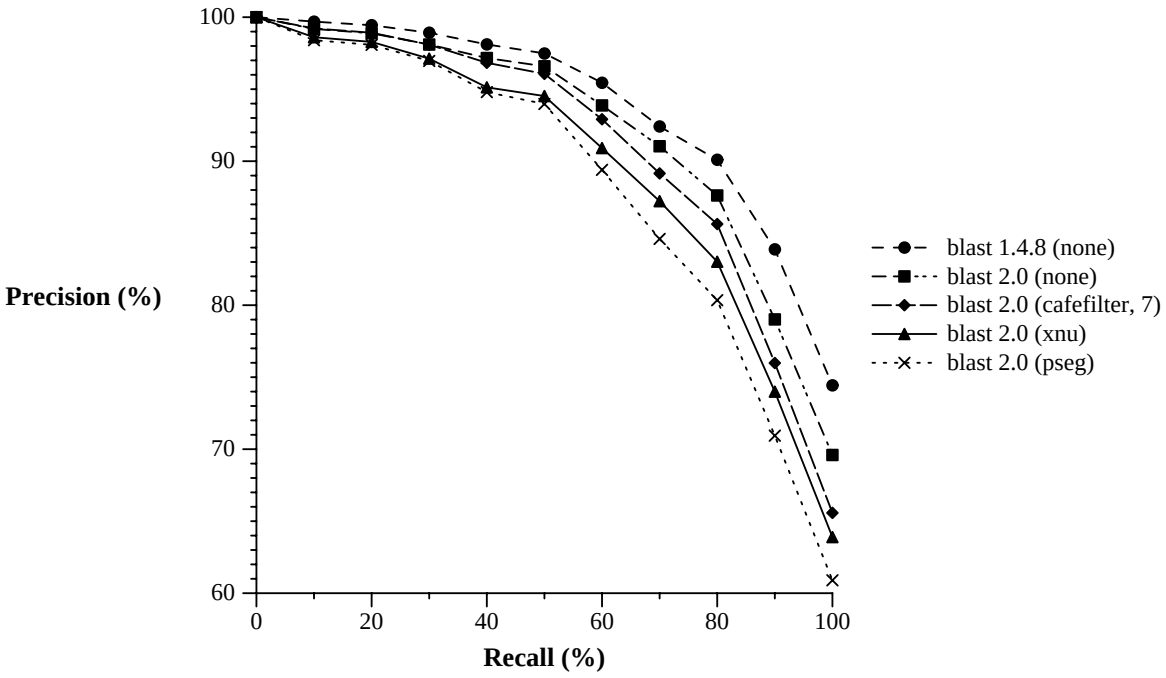

Figure 2: Recall-precision in searching PIRSF with the SEG, XNU, and CAFEFILTER query filtering schemes. The effect on recall-precision is shown in stopping all intervals of length $n = 3$ that occur in more than $x = 7\%$ of sequences. No stopping ($x = 100\%$) with BLAST 1.4.8 and BLAST 2.0 is shown for comparison. The collection, PIRSF, is derived from the PIR protein database and results are an average over 1,834 queries.

An advantage of query filtering is that it decreases disk seek and retrieval times, reduces ranking costs, and significantly reduces computationally intensive local alignment. As an example, with a CAFEFILTER stop-list size of $x = 7\%$ the average elapsed query evaluation time in searching PIRSF with our CAFE search engine is reduced by over 15% from 1.53 seconds to 1.29 seconds. In searching nucleotide sequences in GenBank with a nucleotide query set, this effect is similar with query times also reduced by more than 15%; because of space limitations we do not present detailed results here.

Retrieval Effectiveness of Blast 1.4.8 and 2.0

Figure 2 also shows a comparison of the retrieval effectiveness of BLAST 1.4.8 and BLAST 2.0 with no query filtering applied. In contrast to the findings of Altschul et al. [1], we have found that the new version of BLAST is less effective in searching PIRSF than the previous version. We believe that the reduced retrieval effectiveness can be attributed to new heuristics in BLAST 2.0 that are used to reduce the overall BLAST query costs. A summary of the differences in the two BLAST versions is as follows.

To reduce the query evaluation costs of exhaustively searching large genomic databases such as GenBank, the first version of BLAST [2] permits only the alignment of sequences with intervals common to a query of a minimum length of n. Moreover, this first version does not permit gap insertions or deletions in subsequent local alignments.

The second version of BLAST has removed the restriction of disallowing insertions and deletions by permitting *banded* local alignment; banded alignment allows limited insertions and deletions, permitting the detection of weaker evolutionary relationships with a small increase in computational cost. However, to permit the system to be in general faster than the first version, Altschul et al. have further restricted the frequency of local alignments by requiring two common intervals prior to each banded alignment.

Our results suggest that in version 2.0 of BLAST the improved sensitivity through allowing banded alignment is outweighed by the reduced sensitivity in introducing a more stringent condition for attempting alignments.

5 Conclusions

Previous studies of filtering techniques have focused on the retrieval effectiveness of individual queries [7,9,10]. In contrast, we have presented average retrieval effectiveness results of SEG, XNU, and a new scheme CAFEFILTER in searching a collection derived from the PIR [29] database. Surprisingly, we have found that default filtering of all queries with each technique reduces retrieval effectiveness. Our results suggest that a query should be assessed first before applying filtering, for example, through querying with the unfiltered query and assessing the resultant distribution of similarity scores.

Our new approach to query filtering, CAFEFILTER, filters queries using a stop-list of frequently occurring fixed-length subsequences in a collection. Other popular approaches use query sequence statistics to detect low complexity regions for masking. We have found that CAFEFILTER—despite its simplicity—performs better than existing approaches in large-scale search ex-

periments, but still reduces effectiveness compared to searching with the original, unfiltered queries.

There are several possible extensions to this study. First, better large-scale retrieval effectiveness assessments may be possible when future functional domain classification information is provided in the PIR database[27]. Second, it is likely that special-purpose CAFEFILTER stop-lists may be more effective in filtering particular queries; for example, it is possible to derive a stop-list from a database of sequences of a particular type, such as a database of repetitive or specific amino-acid sequences. Third, a study and characterisation of individual filtered search results may lead to further improvements in filtering techniques. Last, a more detailed assessment of the effectiveness of large-scale query filtering is required in searching genomic nucleotide databases.

Acknowledgments

I thank Justin Zobel for his ongoing involvement in the CAFE project. I also thank Gene Shpaer for providing additional details concerning test collections and the referees for valuable suggestions. This work was supported by the Australian Research Council and the Multimedia Database Systems group at RMIT University.

References

1. S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. *Nucleic Acids Research*, 25(17):3389–3402, 1997.
2. S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. *Journal of Molecular Biology*, 215:403–410, 1990.
3. S. McGinnis. Personal communication. (GenBank user services, National Centre for Biotechnology Information (NCBI), National Library of Medicine, US National Institute of Health), January 1998.
4. S.F. Altschul. *Journal of Molecular Evolution*, 36:290–300, 1993.
5. A. Dembo, S. Karlin, and O. Zeitouni. *Annals of Probability*, 22:2022–2039, 1994.
6. S. Karlin and S.F. Altschul. *Proc. National Academy of Sciences USA*, 87:2264–2268, 1990.
7. S. Altschul, M. Boguski, W. Gish, and J. Wootton. *Nature Genetics*, 6:119–129, 1994.
8. S.F. Altschul and W. Gish. *Methods in Enzymology*, 266:460–480, 1996.
9. W.R. Pearson. *Methods in Enzymology*, 266:227–258, 1996.

10. J.C. Wootton and S. Federhen. *Methods in Enzymology*, 266:554–574, 1996.
11. J.C. Wootton and S. Federhen. *Computers in Chemistry*, 17:149–163, 1993.
12. J.M. Claverie and D.J. States. *Computers in Chemistry*, 17:191–201, 1993.
13. J.M. Hancock and J.S. Armstrong. *Computer Applications in the Biosciences*, 10:67–70, 1994.
14. J. Jurka, P. Klonowski, V. Dagman, and P. Pelton. *Computers in Chemistry*, 20(1):119–122, 1996.
15. V. Brendel, P. Bucher, I. Nourbakhsh, B.E. Blaisdell, and S. Karlin. *Proc. National Academy of Sciences USA*, 89:2002–2006, 1992.
16. C. Liébecq, editor. Portland Press, London, 2nd edition, pages 122–126, 1992.
17. T.F. Smith and M.S. Waterman. *Journal of Molecular Biology*, 147:195–197, 1981.
18. D.J. Lipman and W.R. Pearson. *Science*, 227:1435–1441, 1985.
19. W.R. Pearson. *Methods in Enzymology*, 183:63–98, 1990.
20. W.R. Pearson and D.J. Lipman. *Proc. National Academy of Sciences USA*, 85:2444–2448, 1988.
21. H. Williams and J. Zobel. 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.
22. H. Williams. Compressed indexing for genomic retrieval. *Journal of Mathematical Modelling and Scientific Computing*, 1998 (to appear).
23. H. Williams and J. Zobel. *Computer Applications in the Biosciences*, 13(5):549–554, 1997.
24. H. Williams. *Indexing and Retrieval for Genomic Databases.* PhD thesis, RMIT University, 1998.
25. W.R. Pearson. *Protein Science*, pages 1145–1160, 1995.
26. E.G. Shpaer, M. Robinson, D. Yee, J.D. Candlin, R. Mines, and T. Hunkapiller. *Genomics*, 38:179–191, 1996.
27. W.C. Barker, F. Pfeiffer, and D.C. George. *Methods in Enzymology*, 266:59–71, 1996.
28. I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Van Nostrand Reinhold, New York, 1994.
29. D. George, W. Barker, H. Mewes, F. Pfeiffer, and A. Tsugita. *Nucleic Acids Research*, 24:17–20, 1996.