MOTIF EXPLORER - A TOOL FOR INTERACTIVE EXPLORATION OF AMINOACID SEQUENCE MOTIFS

PAUL BIEGANSKI, JOHN RIEDL AND JOHN V. CARLIS
Computer Science Department, University of Minnesota
{biegansk, riedl, carlis}@cs.umn.edu

ERNEST F. RETZEL
Medical School, University of Minnesota
ernest@lenti.med.umn.edu

Short amino acid sequence patterns, called *motifs,* play an important role in molecular biology research. While a number of tools for locating motifs in sequence databases have been developed, no existing tool performs fast enough to allow interactive searching of entire databases. Interactive searching enables biologists to explore the effects of changes to a motif on the set of sequences matched by the motif. We refer to this process of repetitive searching and motif description modification as *motif exploration.* In this paper, we discuss the application of generalized suffix tree-based indexing structures to high performance motif exploration. We present *Motif Explorer,* a tool accessible via the World Wide Web capable of locating patterns in genetic databases fast enough to support interactive exploration of sequence motifs.

1.0 Introduction

Short amino acid sequence patterns, called *motifs,* play an important role in molecular biology research. They have been shown to be associated with biological functions, such as recognition and targeting. The discovery, exploration and description of sequence motifs is a significant research activity [24].

In order to determine the sequences containing a motif, its description is matched against a database of known sequences. Using presently available software and hardware, the matching process can take from about a minute to many hours. To determine the effects of modifying the motif description on the set of sequences it matches, the search process must be repeated. Performing the search process repeatedly for many variations of the motif description, or to interactively *explore* a motif, the search process has to be fast enough to provide answers within seconds, not minutes or hours [25].

In this paper we discuss using indexing structures based on generalized suffix trees for high-performance motif searches and present an implementation of Motif Explorer, an interactive protein sequence motif exploration tool. We evaluate the performance of Motif Explorer using all patterns in the latest release of the ProSite database and compare it to the performance of other regular expression search programs. We also discuss motif search performance prediction and evaluate a hybrid approach to motif searching combining generalized suffix tree indexing-based approach with other methods. In section 2.0 we present related work, in section 3.0 we discuss the estimation of performance of GST-based regular expression searches and in section 4.0 we discuss the implementation of GST-based indexing structures for large sequence databases and present the Motif Explorer tool. In section 5.0 we present a series of experiments evaluating the performance of the Motif Explorer tool and in section 6.0 we summarize.

2.0 Related work

2.1 Sequence motifs

A sequence *motif* is a short, conserved sequence fragment or set of sequence fragments with biological significance [24]. Examples of biological significance of a motif include recognition, targeting and tertiary structure of the molecule. While a motif consisting of a set of similar fragments can be represented by enumerating the fragments, such a motif can be more efficiently described by a higher level representation. Many different high-level representations have been studied. They include *profiles* based on scoring of sequence alignments [15], *patterns* using constructs based on regular expressions [3], and approaches based on formal grammars [16]. The most widely used repository of sequence motifs, the ProSite database [3], contains mostly pattern descriptions based on a restricted version of regular expressions.

2.2 Regular expressions

Regular expressions [17] are a powerful and efficient method for describing patterns in strings, both in text [17] and in biological sequences [16, 20]. Algorithms for searching for patterns described by regular expressions have been extensively studied in the literature [13, 17] and implementations of such algorithms are com-

monly used in practice [22]. Algorithms for regular expression-based pattern searching fall into two broad categories: direct searching and indirect searching. These two categories are sometimes referred to as searching without preprocessing and searching with preprocessing [13], respectively.

Direct searching uses an algorithm scanning along the sequences being searched to detect the presence of a pattern. The detection task is usually accomplished by a variant of a state machine generated based on the pattern being searched and driven by the sequences being scanned. The performance of such algorithms has a lower bound determined by the size of the sequences being searched (O(N), where N is the size of the sequences) and an upper bound dependent on the complexity of the pattern being sought [13].

Indirect searching preprocesses the sequences from the database to be searched and stores the resulting information in an indexing structure which is used to speed up subsequent searches. The indexing structure allows the detection of the presence and location of certain features of the pattern, such as small fragments, or character composition characteristics within the sequences being searched without having to access all of them. As a result, an indirect search is performed in the feature domain encoded within the indexing structure as opposed to the sequence domain, as is the case during direct search. The performance of indirect search algorithms is dependent on the type of indexing structure used. One of the indexing structures that can be used for indirect pattern searching in sequences is the generalized suffix tree.

2.3 Generalized suffix trees

A *generalized suffix tree*, or *GST*, is based on several simpler structures. First, a *trie* is an indexing structure used for indexing sets of key values of varying sizes [13, 18]. A trie is a tree in which the branching at any level is determined by a partial key value. Second, a Patricia tree [4] is a a trie in which information in each node is used to indicate which portion of the key is used to determine branching. Third, a suffix tree is a Patricia tree [4] built over the set of all suffixes of a given sequence. The applications of GSTs to a number of genetic sequence processing and analysis problems have been discussed in [5] and [6]. GSTs can also be applied to regular expression searching. Given a regular expression (RE) a deterministic finite automaton (DFA) recognizing it can be constructed [13, 17] and consequently used for

locating all of its occurrences in any string. A DFA recognizing all occurrences of a regular expression in a string can be modified to operate by searching a GST corresponding to the string instead of the string itself [13] (Figure 1). The RE search

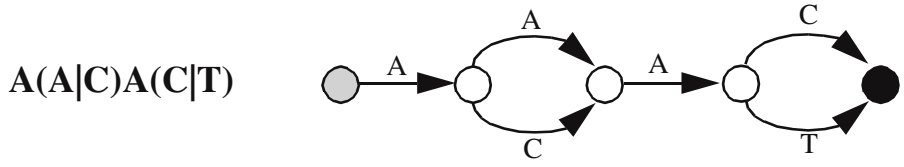

Figure 1. A DFA recognizing strings expressed by the regular expression A(A|C)A(C|T) in a GST. The '|' (vertical bar) operator is equivalent to the '+' operator used by some authors. The grey DFA state is the initial state, the black DFA state is the final (accepting) state.

algorithm operates by traversing the GST depth-first from the root along edges matching state transition labels in the DFA (Figure 2). A substring matching the RE is located when a transition to an accepting state in the DFA occurs. In the following section we analyze the performance of GST-based RE searching and present a method of predicting the search time for a given GST and RE.

3.0 GST-based regular expression search performance prediction

A regular expression recognizes a set of strings. By constructing a trie [13] of all strings recognized by a given RE a trie representation of an RE can be obtained. We will refer to such a trie as RE-trie (Figure 1). For RE's containing the Kleen closure ('*') operator the RE-trie may contain infinite length strings. RE's using a closure operator with a restricted number of repetitions, such as the form used for describ-

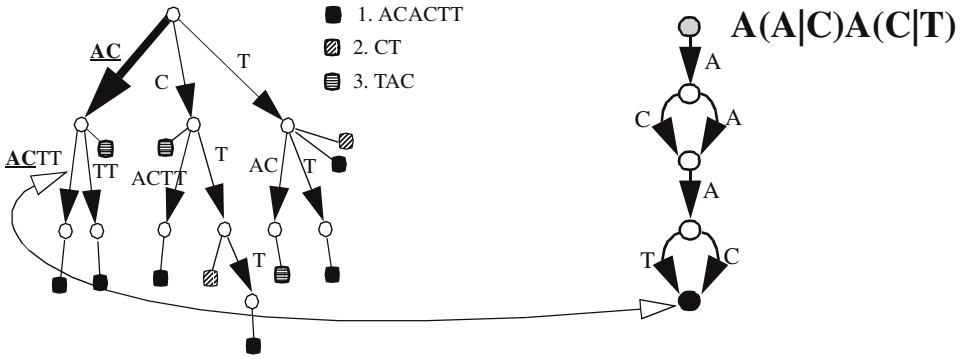

Figure 2. Matching of a GST corresponding to the sequences ACACTT, CT and TAC against a DFA corresponding to the regular expression A(A|C)A(C|T). During the matching process the GST is traversed in a depth-first manner along paths corresponding to strings matching transitions in the DFA. A sequence fragment matching the RE is located when the DFA enters a final state (marked black).

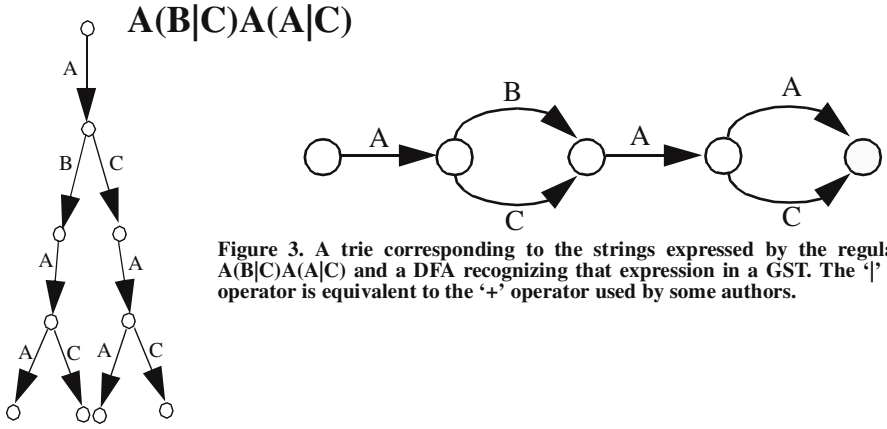

Figure 3. A trie corresponding to the strings expressed by the regular expression A(B|C)A(A|C) and a DFA recognizing that expression in a GST. The '|' (vertical bar) operator is equivalent to the '+' operator used by some authors.

ing biological sequence patterns [16, 20], result in finite RE-tries. Expressing REs in terms of RE-tries makes the RE search algorithm more intuitive and allows for its performance to be analyzed in terms of GST matching (Figure 4). The time necessary to perform the search is proportional to the size of the portion of the RE-trie that matches the GST being searched. The size of the matching portion of the RE-trie is bounded by the size of the entire RE-trie. The only way to exactly determine the portion of the RE-trie that is going to match a given GST is to perform the actual match. An estimate of the matching portion can, however, be obtained by analyzing the structure of the RE-trie and the structure of the GST of the database.

Given the probability P_i of a trie node i being visited during a search in a K-node RE-trie the expected number of nodes visited during the search is:

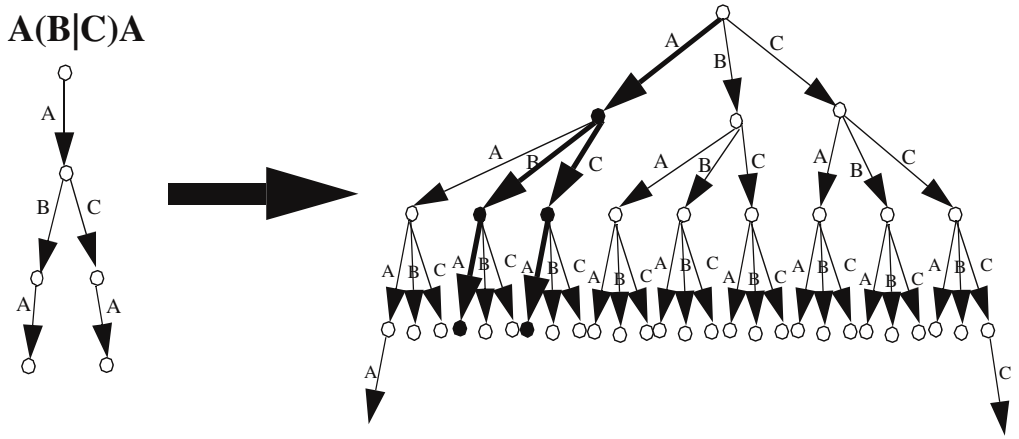

Figure 4. Matching the trie corresponding to a regular expression A(B|C)A against a GST containing all possible suffixes of strings from a 3-letter alphabet [A, B, C]. Only the top three levels of the GST are shown. The search time is proportional to the size of the portion of the trie matching the GST.

$$N = \sum_{i=0}^{K-1} P_i \tag{1}$$

P_i can be expressed recursively in terms of the probability of its parent being visited and the probability M_i of that node having a matching 'twin' node in the GST:

$$P_i = P_{parent(i)} M_i \tag{2}$$

The probability of the root node being visited is 1:

$$P_0 = 1 \tag{3}$$

The probability M_i of an RE-trie node having a match in the GST can be expressed as the ratio of the actual number of nodes at level l(i) of the GST to the maximum possible number of nodes at that level, expressed as a function of the alphabet size A:

$$P_i = \frac{N_{GST}(l(i))}{A^{l(i)}} \tag{4}$$

$N_{GST}(l(i))$ can be taken directly from the GST. It can also be estimated based on the known size D of the database stored in the GST and the distribution of elements in the database [6]. The following equation (5) estimates the number of unique words of length L in a random sequence of length D from a uniformly distributed alphabet of size A and hence the number of paths of length L present in a GST containing the same sequence [6]:

$$N_{GST}(L) = A^L \left(1 - \left(\frac{A^L - 1}{A^L} \right)^D \right) \tag{5}$$

Table 1 presents a comparison of the values predicted by formula (5) for the PIR database [21] and the numbers from an actual GST of the same database. It reveals that formula (5) does provide a good estimate of the actual word count. In order to

Table 1: Estimated vs. actual node counts in a GST corresponding to the PIR database. The apparent error maximum at tree level 5 corresponds to subsequences of length 5, which there are fewer of than predicted under a uniform distribution assumption.

Tree level	**Value estimated according to equation (5)**	**Actual value**	**Estimate error [%]**
0	1	1	0%
1	20	20	0%
2	400	400	0%
3	8000	8000	0%
4	16000	16000	0%
5	2937328	2192742	29%
6	7520198	6524376	15%
7	7975052	7780074	3%
8	7998750	8041095	-1%
9	7999937	8290782	-4%
10	7999996	8382265	-5%

predict the expected number of RE-trie nodes visited, and thus the GST-RE match time, the computation corresponding to equation (4) would have to be performed for the entire RE-trie. This computation, however, can also be performed directly on the DFA by replacing RE-trie branches with DFA state transitions. The computation does not have to be completed to obtain an accurate estimate. In fact, for REs corresponding to infinite RE-tries the computation cannot be completed. It is sufficient, however, to traverse the RE-trie down to a depth at which P_i becomes negligibly small, and therefore does not significantly contribute to the estimate. At depth 8 for the PIR database equation (4) already yields a very small value of 8,041,095 / 20^8 = 0.0003. The estimate computation can also be aborted as soon as the expected number of nodes to be visited becomes larger than some limit, for example a significant fraction of the size of the database, which might indicate that a search using a tool performing sequential scanning of the database such as ANREP [20] instead of a GST could yield better results. The search time estimate can be used by a search program to ask that the user restrict the search or to modify the search strategy to

lower the search time. Examples of possible optimizations include reversing the regular expression and performing a search on a GST of reverse sequences (if available). A GST-based RE search time estimator could also be integrated into a database system's query optimizer.

In Figure 5, we present three sample regular expressions, their corresponding RE-tries and the expected number of nodes matching the GST. They clearly illustrate the value of regular expression search time estimates: seemingly similar queries can result in quite different run times. In Table 2, we present the results of regular

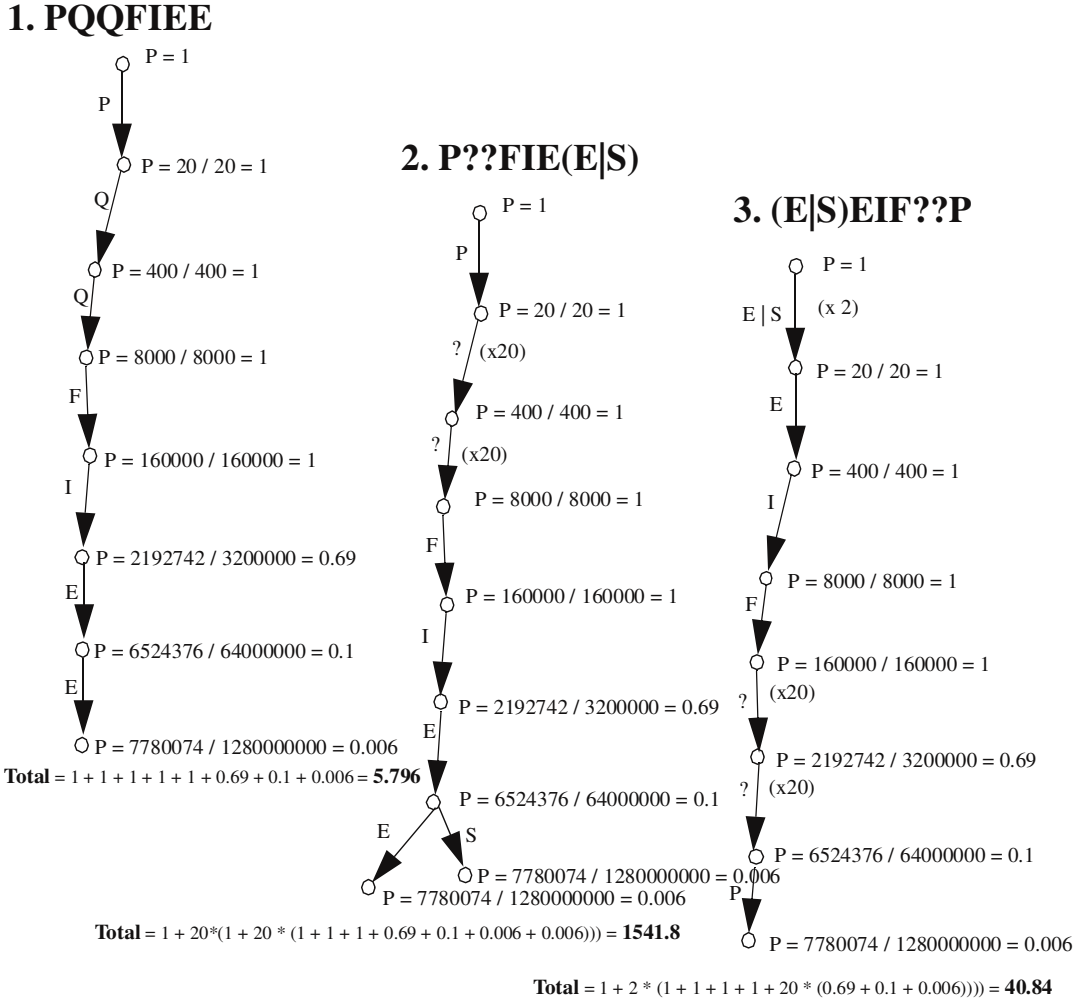

Figure 5. Sample regular expressions and their corresponding RE-tries. The expected match size computation against the PIR database for each RE has been performed. Some multiple RE-trie branches are represented by a single branch with a multiplicity indication (e.g. x 20 indicating 20 identical sub-trees). The match probability P at each level has been computed according to formula (4). Notice the difference between the expected search times of RE #2 and RE #3, which are reverses of each other. By executing a search of the reverse of query #2 (i.e. query #3) on a GST of reverses of sequences in the database the expected search time could be reduced 37 times.

#	Regular expression	Search time [steps]	
		Estimated	Actual
	REs corresponding to known sequence motifs		
1	[ILM][DS][FL][F[ACS]G?[GM][AG][FIL]??[AGS]???G	65	221
2	[ILV]??[INS][DE]?[DFN]??[AI]??[STV][FYI]??[IN]	28120	55011
3	D[IV][RST]	10	9
4	[DN]?[ILV]?[AGS]G[FPS]PC[PQ]?[FW]S??G?????[EDS]	8323	11772
5	[EDP]?[QR][GN]?[LMV][FY]	5262	6263
	Random REs		
6	PQQFIEE	6	5
7	P??FIE[ES]	1542	1299
8	[ES]EIF??P	41	185
9	[QK]QIW??P	41	46
10	K?A?P?Y?Q?A	1590	1977

Table 2: Estimated and actual regular GST expression search times. Expressions 1-5 correspond to motifs for the structure of cytosine methyltransferases [20]. Expressions 6-10 have been randomly chosen. Expressions 6, 7, and 8 are taken from Figure 5. Correlation of the estimated and actual search times is 0.995.

expression search experiments performed on a GST of the PIR database. The experiments compare estimated and actual search times for REs describing known sequence features [20] (rows 1 through 5), and for random REs (rows 6 through 10). Data in Table 2 shows that the search time estimate performs remarkably well, especially when the assumption of uniform distribution is considered. The correlation of the estimated and actual search times is 0.995.

4.0 Implementation

4.1 GST

Implementing and maintaining GST-based indexing structures for large sequence databases requires intricate implementation and significant computational resources. The key performance obstacle is the need to update large data structures residing on disks, which are many orders of magnitude slower than main memory.

We have developed a technique allowing GST to be created and processed as linear streams of data, using limited main memory and without random access to the entire structure. For a detailed discussion of issues involved in implementing large scale GST structures we refer the reader to [6].

Our GST implementation allows for a GST corresponding to the entire PIR database to be constructed in a few hours on a 100MHz Silicon Graphics workstation. The 140Mb release 44.0 of the PIR database [21], results in a 310Mb GST structure.

4.2 Motif Explorer

The Motif Explorer is an interactive sequence motif exploration tool using a GST structure to provide performance levels acceptable for interactive searching. It combines our implementation of large-scale GST structures (see section 4.1) with the GST regular expression search algorithm (see section 2.3). The Motif Explorer uses the GST regular expression search performance predictor to improve the performance of the search. Motif Explorer is equipped with a World Wide Web (WWW)-based user interface (Figure 6). It is accessible at http://lenti.med.umn.edu/gst/MotifExplorer.html.

Figure 7 presents an overview of the main functional blocks of motif Explorer. All user interface functions are performed using a WWW front end. Using WWW makes the Motif Explorer easily accessible to the research community without requiring large computational resources at each site. Motif Explorer accepts queries in UNIX-style regular expression format [22] as well as in ProSite database format [3]. Following parsing, the query is passed on to a search time prediction module implementing the algorithm described in section 3.0. Based on the results of the prediction the forward or reverse GST structure is chosen for searching. If only one structure is available, which may happen in case of limited disk resources, it is used. Optionally, if the prediction algorithm indicates a very long search, the user may be warned that an answer cannot be quickly produced or an alternative, fall-back search algorithm may be activated. Following the search, results are formatted and presented to the user as a hyper-text document. The search results include the matching sequences as well as additional information obtained from the GST structure, such as unique matching sequence fragments and the frequency of their occur-

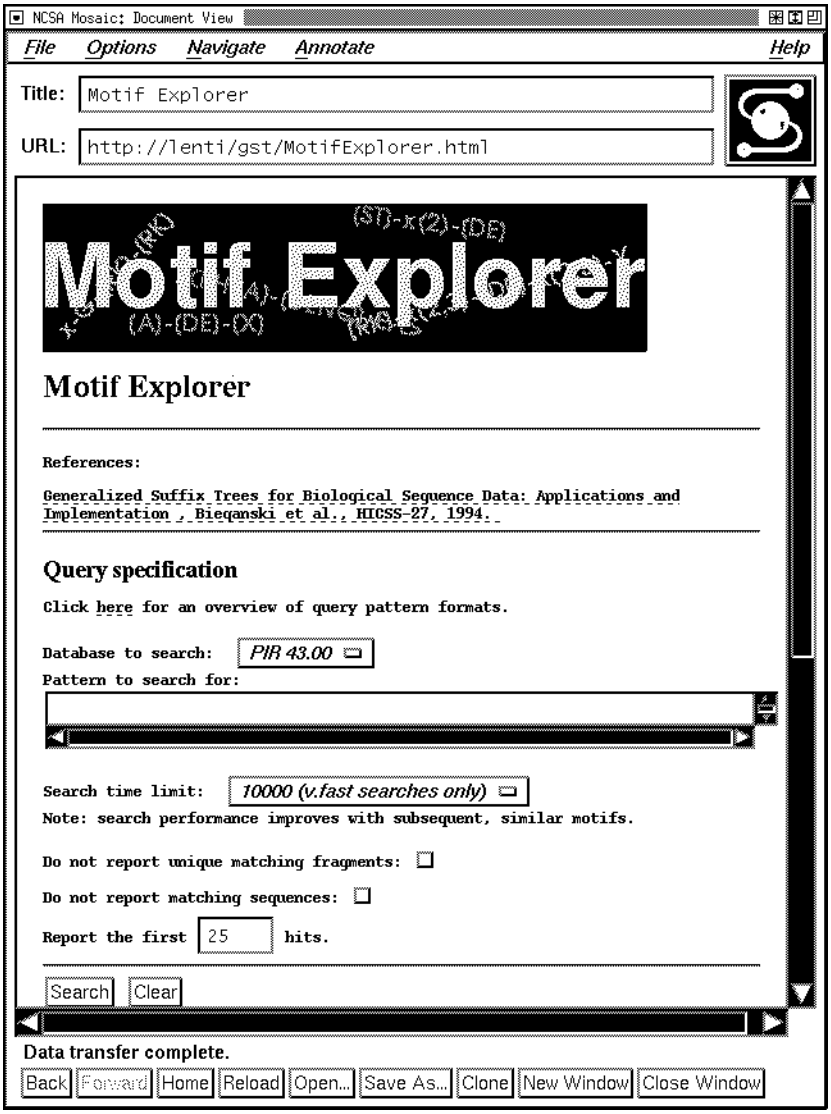

Figure 6. WWW-based Motif Explorer user interface. Using WWW as the user interface makes the tool accessible to any user equipped with a WWW browser and does not require any local high-performance computing.

rence within the database. Whenever possible, the Motif Explorer includes WWW links to appropriate databases containing information on the matching sequences.

5.0 Experiments

In this section, we present a series of experiments aimed at determining the suitability of GST-based pattern searching for interactive exploration of sequence motifs. In the experiments presented below we have studied the performance of the GST-

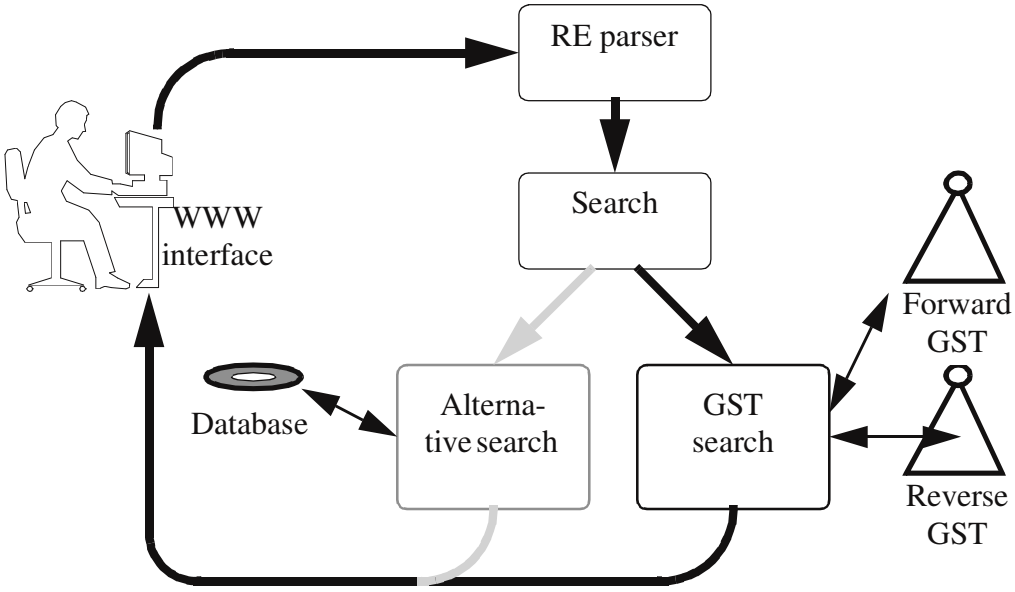

Figure 7. Main functional blocks of the Motif Explorer tool. Motif Explorer is accessed via a WWW-based form interface. The user query is processed by the query parser and passed to the search time predictor for evaluation. Based on the prediction results a GST-based search strategy is chosen or (optionally) a different search engine is invoked. The query results are formatted according to user specified criteria and presented via the WWW interface.

based regular expression search time predictor described in section 3.0 and compared the performance of a GST-based regular expression search tool to the performance of a tool using a non-indexed, direct search algorithm.

5.1 Experimental design

All experiments have been performed on a 4-processor Sun 690/MPA computer with 256M RAM and a locally attached disk. Experiments were performed in a production environment, with the machine also acting as a file server for approximately 30 workstations. The system load during the experiments varied from 2 to 8. Search times were obtained using *time* -- an operating system-provided resource usage monitoring function [22]. Both time spent executing actual search code (*user time*) and time spent on reading data from disk (*system time*) were included.

The goal of our experiments has been the comparative evaluation of an indirect, GST index-based motif search algorithm implementation with a direct, non-indexed motif search algorithm implementation. Preliminary testing and examination of the source code and documentation of one non-indexed genetic sequence motif search application, ANREP [20], has revealed that it has not been optimized

for performance and therefore could not provide an experimental verification of the upper limit of performance of direct search-based programs. On the same hardware platform that was used for all other experiments ANREP has been found to scan approximately 63,500 characters per second (15 seconds per megabyte) when looking for short, exact matching words. This resulted in search times of over 5 minutes for the PIR database, making it unacceptable for interactive exploration. We have therefore chosen GNU grep 2.0, a program performing near the theoretical upper limits for regular expression text scanning [12]. GNU grep 2.0 has been found to scan approximately 5 million characters per second when looking for short, exact matching words.

In all of our experiments, we have used release 43.00 of the PIR database as the data set being searched. We have used the 1025 pattern entries of release 12.00 of the ProSite motif database as sample queries. In order to evaluate the performance of the grep program, we have converted the PIR database into a one-line-per-entry format. The ProSite database patterns have been converted to a format compatible with the grep program. The Motif Explorer tool used a pair of GST structures constructed for the PIR database and unmodified ProSite pattern descriptions. In order to automate the large number of test runs, we have used a command-line version of the Motif Explorer tool. In order to eliminate highly redundant patterns resulting in searches lasting multiple hours and not suitable for interactive exploration a predicted search time cutoff of 150,000 steps was used. Both the grep program and the Motif Explorer tools were run without search report generation. Matches were located and counted, but no output was generated in order to simplify the timing procedure. In order to simulate multiple, repetitive searches that might be performed during interactive motif exploration each test consisted of a series of queries corresponding to 1025 pattern entries from the ProSite database.

5.2 Experimental results

5.2.1 grep

Figure 8 presents the average search time distribution for the grep program. Examination of Figure 8 reveals that the frequency distribution curve is bimodal. Its two peaks correspond to two major categories of patterns in the ProSite database. The first category is comprised of short, low-redundancy patterns, such as R-G-D,

resulting in a quick, 5 to 7 second scan over the database at a speed limited almost strictly by the I/O performance of the system. The second category is comprised of longer, higher redundancy complex patterns, such as [RK]-x(2,3)-[DE]-x(2,3)-Y, resulting in a large DFA or requiring backtracking during the search, leading to exponential performance [12]. The average and median search times using the grep program were 60.4 and 18.9 seconds, respectively.

5.2.2 Motif Explorer

Figure 9 presents the average search time distribution for the Motif Explorer tool. 75% of all searches represented in Figure 9 were performed in under 3.7 seconds. 95% of all patterns had predicted search times below the 150,000 step cutoff. The average and median search times for these patterns were 3.78 and 1.06 seconds, respectively.

5.2.3 GST search time predictor

We have tested the performance of the GST motif search time predictor described in section 3.0 and its applicability for choosing between a GST-based search algo-

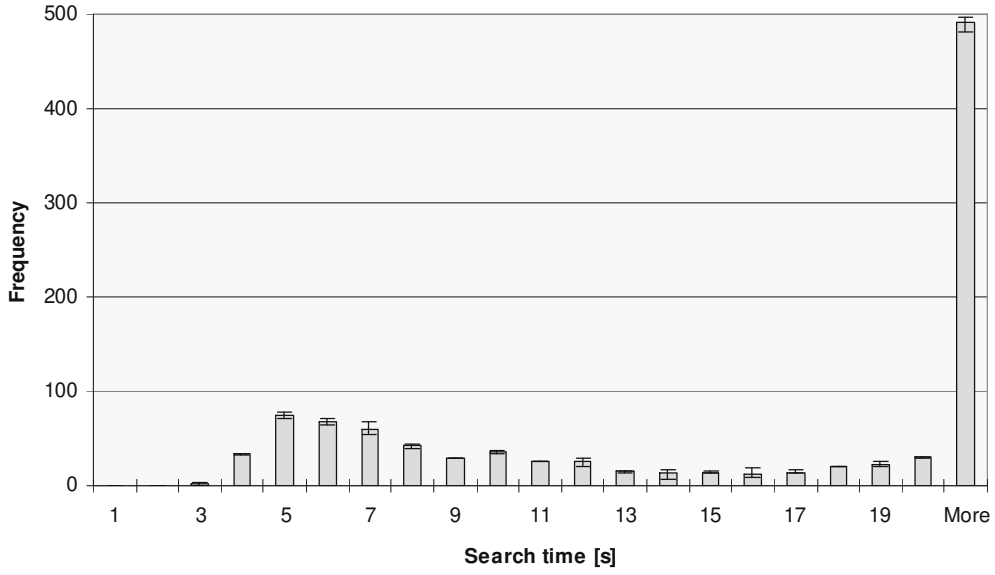

Figure 8. Search time frequency distribution for the grep program. Search times were obtained by searching the PIR database for 1025 pattern entries from the ProSite database. The search was performed 3 times. Error bars reflect the minimum and maximum frequencies over the 3 tests sets.

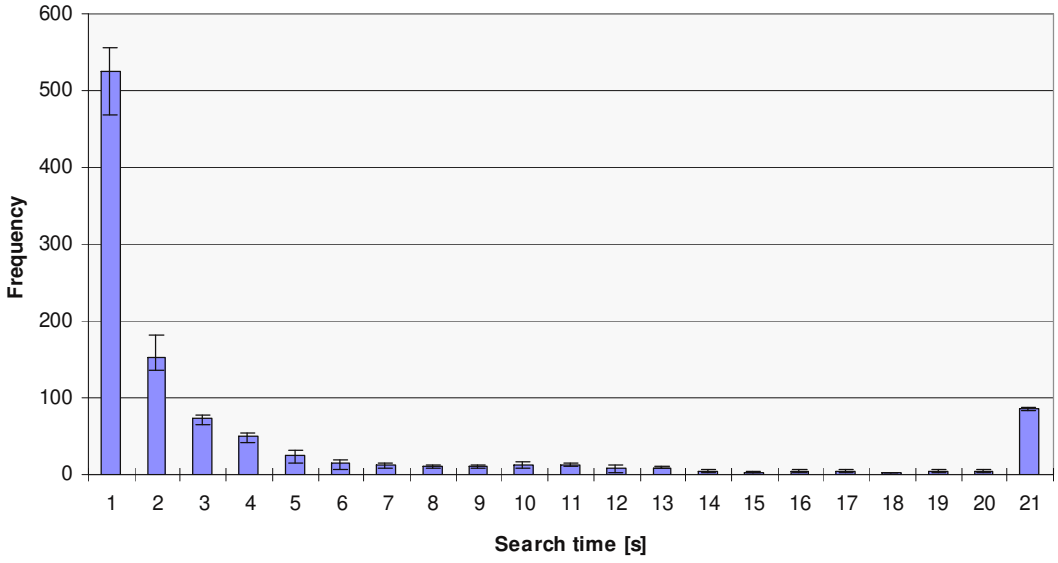

Figure 9. Search time frequency distribution for the Motif Explorer tool. Search times were obtained by searching the PIR database for 1025 pattern entries from the ProSite database. The search was performed 3 times. Error bars reflect the minimum and maximum frequencies over the 3 tests sets.

rithm and a direct searching algorithm prior to performing the search. An effective predictor would allow the best search strategy for a given regular expression to be chosen. Table 3 presents a comparison of three different predicted search time cut-

Table 3: Comparison of actual search time statistics for different predicted search time cutoffs and an optimal search. The optimal search times were obtained by performing each search using both Motif Explorer and grep and choosing the method with shorter search time.

Predicted search time cutoff [steps]	Number of patterns (% of all, 100%=1,025)			Search time	
	Below cutoff	Performing better with Motif Explorer	Performing better with grep	Average [s]	Median [s]
50,000	932 (91%)	923 (90%)	102 (10%)	11.98	1.16
100,000	961 (94%)	941 (92%)	84 (8%)	10.29	1.15
150,000	975 (95%)	946 (92%)	79 (8%)	10.50	1.15
Optimal	-	946 (92%)	79 (8%)	10.02	1.15

off values and their effectiveness in choosing correctly between a GST-based and a grep-based search. Figure 10 presents a comparison of the optimal distribution with

a distribution resulting from a 150k cutoff. It shows that the predictor-based cutoff method of choosing the search method is very close to optimal.

5.2.4 Summary

The experimental results presented above indicate that in the majority of cases grep, a linearly scanning algorithm, performs much worse than the GST-based Motif Explorer (median search time of 18.9 vs. 1.06 seconds). The short search times of Motif Explorer allow it to be applied interactively to the study of sequence motifs. Occasionally, high-redundancy patterns can cause the Motif Explorer's performance to degrade below that of a linearly scanning algorithm. We have shown that it is possible to use a search time prediction technique to detect such cases prior to the search and to choose the fastest algorithm.

6.0 Conclusions and future work

Our experiences with the Motif Explorer tool indicate that the GST-based regular expression search algorithm can be successfully applied to the exploration of protein sequence motifs. The Motif Explorer tool provides the performance necessary

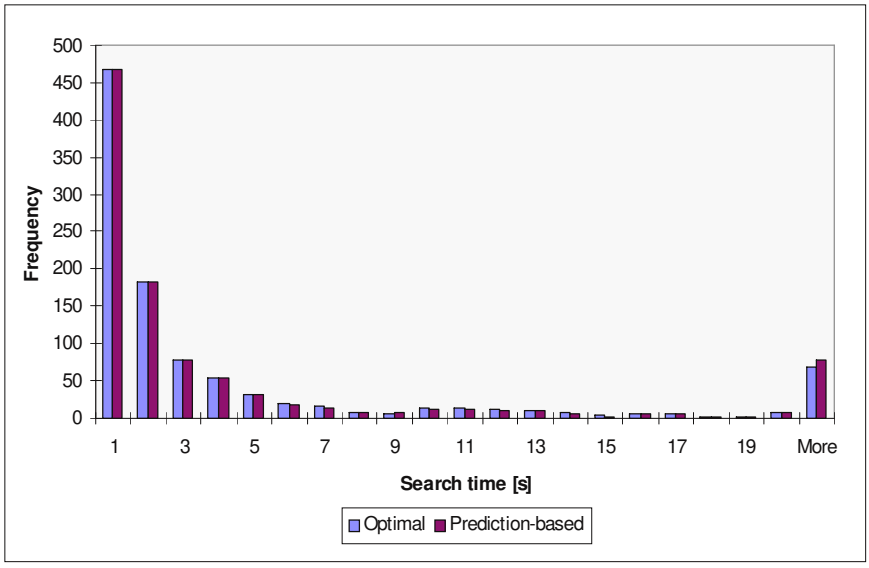

Figure 10. Comparison of optimal and prediction-based search time distributions. The optimal distribution has been obtained by choosing the experimentally determined minimum search time. The prediction-based distribution has been obtained by using a Motif Explorer search time for all patterns with predicted search times of 150,000 steps or less and the grep search time otherwise.

to achieve response times acceptable for interactive exploration. Our experiments with patterns from the ProSite database indicate that a GST-based search tool is faster than the best linear scanning tools in the vast majority of cases. For some high redundancy patterns, a linear scanning method may, however, perform better. Using the GST regular expression search performance predictor we have shown that it is possible to construct a hybrid system combining a GST-based system with a linear scanning system. A prototype of the Motif Explorer tool has been successfully used during mapping of an opioid receptor [2].

We are currently investigating the application of the GST-based regular expression search engine used by Motif Explorer as a part of a motif search tool capable of searching for higher-level motif networks of motifs and spacers, similar to those used by ANREP [20]. A nucleic acid motif version of Motif Explorer is being used to explore DNA sequence patterns in EST sequences produced by the *Arabidopsis Thaliana* genome project [http://lenti.med.umn.edu]. The latest version of Motif Explorer and the latest work-in-progress report can be accessed via the World Wide Web at http://lenti.med.umn.edu/gst/MotifExplorer.html.

7.0 Bibliography

[1] Apostolico, A., *The Myriad Virtues of Suffix Trees*, Combinatorial Algorithms on Words, pp. 85-96, Springer-Verlag, 1985.

[2] Arvidsson, U., Dado R. J, Law, P.Y., Loh, H.L. Elde, R. and Wessendorf, M.W., *Delta-opioid Receptor Immunoreactivity: Distribution in Brain Stem and Spinal Cord and Relationship to Biogenic Amines and Enkephalin*, Journal of Neuroscience, February 1994.

[3] Bairoch, A., *PROSITE: a Dictionary of Sites and Patterns in Proteins,* Nucleic Acids Research, 19:2241-2245, 1991.

[4] Bays, J. C., *The Complete PATRICIA*, Ph.D. dissertation, University of Oklahoma, 1974.

[5] Bieganski, P., Riedl, J., Carlis, J. and Retzel, E., *Generalized Suffix Trees for Biological Sequence Data: Applications and Implementation*, Proceedings of the 27th Hawaii International Conference on Systems Science, Maui, January 4-7, 1994.

[6] Bieganski, P., *Genetic Sequence Data Retrieval and Manipulation based on Generalized Suffix Trees*, Ph.D. dissertation, University of Minnesota, 1995.

[7] Boyer, R.S., Moore, J.S., *A Fast String Searching Algorithm*, Communications of the ACM, Vol. 20, pp. 762-772, October 1977.

[8] Chan, S.C., Wong, A. K. C. and Chiu, D. K. Y., *A Survey of Multiple Sequence Comparison Methods*, Bulletin of Mathematical Biology, Vol. 54, No. 4, pp. 563-598, 1992.

[9] Chao, Kun-Mao, Hardison, Ross C. and Miller, Webb, *Constrained Sequence Alignment*, Bulletin of Mathematical Biology, Vol. 55, No. 3, pp. 503-524, 1993.

[10] Chi Kwong Lui, Lucas, *Color Set Size Problem with Applications to String Matching*, Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching, Tucson, Arizona, USA, April/May 1992.

[11] McCreight, E. M., *A Space Economical Suffix Tree Construction Algorithm*, JACM, 23, 262-272, 1976.

[12] Haertel, M., *Gnugrep 2.0*, Usenet archive comp.sources.reviewed, Vol. 3, July, 1993.

[13] Gonnet, G.H. and Baeza-Yates, R., *Handbook of Algorithms and Data Structures*, 2nd edition, Addison-Wesley, Reading, Massachusetts, 1991.

[14] Grassberger, Peter, *Estimating the Information Content of Symbol Sequences and Efficient Codes*, IEEE Transactions on Information Theory, Vol. 35, No. 3, May 1989

[15] Gribskov, M., McLachlan, M., Eisenberg, D., *Profile Analysis: Detection of Distantly Related Proteins*, Proceedings of National Academy of Science, 84:4355-4358, 1987.

[16] Helgesen, Carsten and Sibbald, Peter R., *PALM - A pattern language for molecular biology*, Proceedings of the First International Conference on Intelligent Systems for Molecular Biology, Bethesda, Maryland, pp. 172-180, 1993.

[17] Hopcroft, J.E., and Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Massachusetts, 1979.

[18] Horowitz, Ellis and Sahni, Sartaj, *Fundamentals of Data Structures*, Computer Science Press, 1987.

[19] Landau, Gad M. and Vishkin, Uzi, *Fast Parallel and Serial Approximate String Matching*, Journal of Algorithms, 10, pp.157-169, 1989.

[20] Mehldau, Gerhard and Myers, Gene, *A System for Pattern Matching Applications on Biosequences*, Technical Paper, University of Arizona, 1992.

[21] National Biomedical Research Foundation, *Protein Information Resource (PIR)*, Release 41.00, June 30, 1994.

[22] Sun Microsystems, *System 4 Users Manual*, 1993.

[23] Szpankowski, Wojciech. *Probabilistic Analysis of Generalized Suffix Trees*, Combinatorial Pattern Matching, Third Annual Symposium, Proceedings, Springer Verlag, 1992, 1-14.

[24] Taylor, W. R., *Patterns in Protein Sequence and Structure*, Springer-Verla, Heidelberg, 1992.

[25] Woodson, W.E., *Human Engineering Guide*, University of California Press, 1954.