Experimental Evaluation of Fast Solution Algorithms for the Simple Motif Problem

Tarek El Falah[1,2], Thierry Lecroq[2], and Mourad Elloumi[1]

[1] Research Unit of Technologies of Information and Communication,
Higher School of Sciences and Technologies of Tunis, 1008 Tunis, Tunisia
`Tarek.Elfalah@etu.univ-rouen.fr`,
`Thierry.Lecroq@univ-rouen.fr`
[2] University of Rouen, LITIS EA 4108,
76821 Mont-Saint-Aignan Cedex, France
`Mourad.Elloumi@fsegt.rnu.tn`

Abstract. The *Simple Motif Problem* (SMP) is: given a set of strings $Y = \{y_0, y_1, \ldots, y_{n-1}\}$ built from a finite alphabet Σ, $p > 0$ an integer and $q \leq n$ a quorum, find all the simple motifs of length at most p that occurs in at least q strings of Y.

This paper presents an experimental evaluation of algorithms dealing with SMP and using a minimal forbidden pattern approach. The experiments are concerned both with running times and space consumption.

Keywords: algorithms, strings, motifs, SMP, complexities.

1 Introduction

The motif finding problem consists in finding substrings that are more or less conserved in a set of strings. This problem is a fundamental one in both Computer Science and Molecular Biology. Indeed, when the concerned strings code biological sequences, i.e., DNA, RNA and proteins, extracted motifs offer to Biologists many tracks to explore and help them to deal with many challenging problems. In the literature, several versions of the motif finding problem have been identified [4]:

– *Planted (l,d)-Motif Problem* (PMP) [2,11];
– *Extended (l,d)-Motif Problem* (ExMP) [9,14];
– *Edited Motif Problem* (EdMP) [12,13];
– *Simple Motif Problem* (SMP) [12,5].

A *simple motif* is a string built from a finite alphabet $\Sigma \cup \{\texttt{?}\}$ that cannot begin or end with `?`, where Σ is a set of symbols and $\texttt{?} \notin \Sigma$ is a wildcard symbol, it can be replaced by any symbol from Σ.

Let $Y = \{y_0, y_1, \ldots, y_{n-1}\}$ be a set of strings built from an alphabet Σ, $p > 0$ be an integer and $q \leq n$ be a quorum, the *Simple Motif Problem* (SMP) is to find all the simple motifs of length at most p that occurs in at least q strings of Y.

© Springer-Verlag Berlin Heidelberg 2011

Actually, concerning SMP the main existing solutions are:

- the algorithm given by Floratos and Rigoutsos [8],
- the one described by Rajasekaran *et al.* [12].
- the SMS-Forbid and SMS-H-Forbid algorithms by El Falah *et al.* [5,6].

In this paper, we focus on SMP and we propose an experimental evaluation of algorithms SMS-Forbid and SMS-H-Forbid. Both algorithms are based on a minimal forbidden pattern approach. Algorithm SMS-Forbid uses indexing structures (either suffix trees or suffix arrays) while algorithm SMS-H-Forbid uses an hash table. The experiments are concerned both with running times and space consumption. These experiments were conducted on pseudo-randomly generated data on two alphabet sizes: 4 (to simulate DNA sequences) and 20 (to simulate protein sequences).

We organize the rest of the paper as follows: In section 2, we present some useful definitions and notations. In section 3, we review algorithms SMS-Forbid and SMS-H-Forbid. Section 4 presents experimental results. In section 5, we give our conclusion and perspectives.

2 Preliminaries

A *simple motif* is a string built from an alphabet $\Sigma \cup \{?\}$ that cannot begin or end with ?, where Σ is a finite set of symbols and $? \notin \Sigma$ is a wildcard symbol, it can be replaced by any symbol from Σ. Symbols of Σ are said to be *solid* while the wildcard symbol ? is said to be *non-solid*. The length of a simple motif is the number of the symbols that constitute this motif, including the wildcard symbols.

A string of ℓ symbols from Σ is called a ℓ-mer. A string of ℓ symbols from $\Sigma \cup \{?\}$ is called a ℓ-pattern. A ℓ-pattern z_1 is equivalent to a ℓ-pattern z_2 ($z_1 \cong z_2$), if at every position where z_1 and z_2 contains both solid symbols these symbols are equal. Formally, $z_1 \cong z_2$ if for $1 \leq i \leq \ell$: $z_1[i] = z_2[i]$ or $z_1[i] = ?$ or $z_2[i] = ?$

A ℓ-pattern z_1 is more general than a ℓ-pattern z_2 if a position in z_2 contains the wildcard symbol implies that the same position in z_1 contains the wildcard symbol and if a position in z_2 contains a solid symbol then at the same position in z_1 there could be either the same symbol or a wildcard symbol. Formally $z_2[i] = ? \Rightarrow z_1[i] = ?$ and $z_2[i] = a \Rightarrow (z_1[i] = a$ or $z_1[i] = ?)$ for $1 \leq i \leq \ell$ and $a \in \Sigma$.

Let $Y = \{y_0, y_1, \ldots, y_{n-1}\}$ be a set of strings built from an alphabet Σ and let $N = \sum_{i=0}^{n-1} |y_i|$. Let m be the maximal length of the strings in Y. Let σ be the size of the alphabet Σ.

3 Minimal Forbidden Pattern Approach

In this section, we present two algorithms SMS-Forbid and SMS-H-Forbid.

3.1 SMS-Forbid Algorithm

Given a set Y of n strings, a quorum $q \leq n$ and an integer p. The algorithms output the set of motifs of length at most p that occurs in at least q strings.

A pattern z of length at most p is said to be a minimal forbidden pattern if it occurs in less than q strings but all its proper factors beginning and ending with a solid symbol occur in at least q strings.

For each position on the input strings, we use all the windows of length ℓ for $3 \leq \ell \leq p$. Each window defines an ℓ-mer. Each ℓ-mer x defines a set of ℓ-patterns X. At each position of each ℓ-pattern z of X, the symbol of z is either the symbol at the same position of x or the wildcard symbol except for the first and the last symbols of z that are necessarily non-wildcard symbols. Formally, $z[i] = x[i]$ or $z[i] = \texttt{?}$ for $1 \leq i \leq \ell - 2$ and $z[0] = x[0]$ and $z[\ell - 1] = x[\ell - 1]$.

These ℓ-patterns together with the generality relation form a lattice. The minimal element of the lattice is x itself and the maximal element is $x[0]\texttt{?}^{\ell-2}x[\ell-1]$. (see example in Fig. 1: the minimal element is $x = \texttt{CAGAT}$ and the maximal element is $\texttt{C?}^{\ell-2}\texttt{T}$).

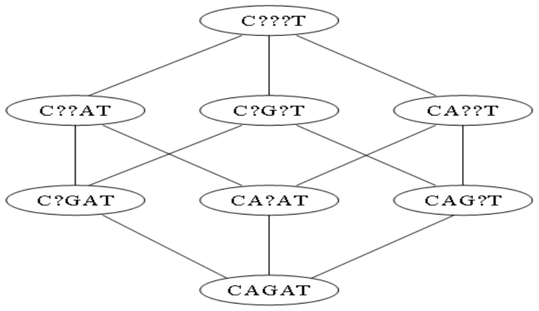

Fig. 1. A 5-mer $x = \texttt{CAGAT}$ lattice

Each node of the lattice represents an ℓ-pattern.

The ℓ-patterns are scanned by doing a breadth-first search of the lattice beginning from the minimal element.

When a ℓ-pattern z is considered, if it has already been output or it contains minimal forbidden patterns as factors or it is more general than an output pattern, then it is disregarded otherwise it is searched in the strings of Y. Then if it occurs in at least q strings it is output and all its successors in the lattice are not considered since they are more general. On the contrary if it does not occur in at least q strings it is added to the set of minimal forbidden patterns.

The generation of the ℓ-patterns is performed using a breadth-first search of the lattice for the following reason. When a ℓ-pattern is discovered all its successors in the lattice, that are more general, do not have to be considered. They are thus marked using a depth-first search of the lattice from the discovered ℓ-pattern. During the remaining of the breadth-first search, marked ℓ-patterns are not considered.

Contrary to the algorithm presented in [12], the new approach does not search for all the ℓ-patterns generated from the n strings of Y but it begins by searching the more specific patterns i.e. the less general patterns which avoids the sorting step. Moreover it maintains a set of minimal forbidden patterns that do not occur in at least q strings in order to not search for any pattern that contains a factor that has already been unsuccessfully searched. This two techniques reduce the number of patterns to be searched for. The search of one ℓ-pattern in one string y of Y is done using an indexing structure of y which can be done in a time proportional to ℓ. Furthermore the new approach only outputs the more specific motifs that occur in at least q strings of Y. This should help the biologist to identify important motifs.

An illustrative example: Let $Y = \{y_0 = \texttt{ACGAAGACT}, y_1 = \texttt{CGTAGCTAC}, y_2 = \texttt{CAGTACTAT}, y_3 = \texttt{ACGTACAAA}, y_4 = \texttt{CCTACTGCT}\}$ be a set of strings built from the alphabet $\Sigma = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$. The set of most specific simple motifs having length at most $p = 6$ and occuring in at least $q = 3$ strings of Y is

$$\{\texttt{TAC}, \texttt{ACT}, \texttt{GTA}, \texttt{CTA}, \texttt{C?TA}, \texttt{CG?A}\}.$$

We will give a top-down detailed presentation of the algorithm. It builds the set *Res* of searched motifs of length at most p contained in at least q strings and uses a set $\mathcal{T}$ of minimal patterns that are not contained in at least q strings. It scans the n strings of Y from y_0 to y_{n-1}. For each position of each string it considers all the windows of length ℓ for $3 \leq \ell \leq p$ (at the end of the strings it may be less than p). For each ℓ-mer x defined by each window of length ℓ, the breadth-first search of the lattice is performed level by level.

During the breadth-first search of the lattice, when a non-marked ℓ-pattern x is considered, it is searched in the set *Res* of resulting ℓ-motifs. For that, the set *Res* is implemented using a trie: looking if an ℓ-pattern x belongs to *Res* simply consists in spelling x from the root of the trie. If x belongs to *Res* then all it successors are marked using a depth-first search.

If x does not belong to *Res* then the algorithm checks if it contains minimal forbidden patterns. This consists in searching for a finite set of patterns $\mathcal{T}$ with wildcard symbols in a text with wildcard symbols x, where a wildcard symbol in the text only matches a wildcard symbol in the set of patterns while a wildcard in the set of patterns may match any symbol in the text.

If x does not contain any minimal forbidden pattern then it is searched in all the strings of Y from y_{j+1} to y_{n-1} since it has not been considered before dealing with y_j and it has at least one occurrence in y_j. If it occurs in at least q strings, it is added to *Res* and all its successors are marked using a depth-first search. Otherwise it is added to the set $\mathcal{T}$ of minimal forbidden patterns.

The lattice is completely traversed in a breadth-first search in every cases.

Each ℓ-pattern x in the lattice is associated with an integer from 0 to $2^{\ell-2}$ whose binary representation is given by $x[1 \mathinner{.\,.} \ell-2]$ where each solid symbol is replaced by 0 and each wildcard symbol is replaced by 1. For example `ab?ba` is associated to 2 whose binary representation is 010. This enables to mark easily the nodes of the lattice.

The candidate patterns are generated by considering the strings from y_0 to y_{n-1}. When a pattern is generated from y_j it occurs at least once in y_j then it is searched in the following strings: $y_{j+1}, y_{j+2}, \ldots, y_{n-1}$.

This procedure considers a factorization of an ℓ-pattern x as follows:

$$x = u_0 v_0 u_1, v_1 \cdots u_{m-1} v_{m-1} u_m$$

where $u_i \in \Sigma^*$ for $0 \leq i \leq m$ and $v_j \in \{?\}^*$ for $0 \leq j \leq m-1$ (remember that an ℓ-pattern begins and ends with a solid symbol).

Then the search of x in y is performed by successively searching for the u_i in y and merging sets of positions. Assume that R contains all the positions of $u_1 \cdots u_i$ in y, then the procedure computes the set T of all the positions of u_{i+1} in y. The two sets are merge in order to keep only the positions of R that are compatible with positions of T. A position g of u_i is compatible with a position of T if there exists a position h in T such that $g = h + |u_1 \cdots v_i|$. The merge can be easily realized if the two sets are implemented as ordered arrays. The set of positions of u_i in y can be computed efficiently by using any indexing structures of y (such as suffix trees or suffix arrays).

A problem remains when a more specific patterns is found after a more general pattern. Then the set of resulting patterns has to be filtered.

3.2 SMS-H-Forbid Algorithm

In order to easily find the candidate patterns we define a table H for every couple of solid symbols and every integer k from 0 to $p-3$ as follows:

$$H[a, b, k] = \{(i, j) \mid y_i[j] = a \text{ and } y_i[j + k + 2] = b\}.$$

When a candidate ℓ-pattern is generated from position j in string y_i, if

- it has not already been output or
- it does not contain minimal forbidden patterns as factors or
- it is not more general than an output pattern

its potential occurrences are only searched at the positions in $H[y_i[j], y_i[j + \ell - 1], \ell - 2]$. In practice, the elements of $H[a, b, k]$ are sorted in decreasing order of the index of the strings.

The algorithm scans the strings of Y in the same order than algorithm SMS-Forbid. The bread-first-search is performed in the same manner. The only changes appear for counting the number of strings containing an ℓ-pattern x generated from y_j. The occurrences of x are searched using the list of pairs in $H[x[0], x[\ell - 1], \ell - 3]$. Furthermore those pairs (ind, pos) are sorted in decreasing order thus only pairs where $ind > j$ are considered.

3.3 Complexities

We will now give the complexities of both algorithms SMS-Forbid and SMS-H-Forbid.

The algorithm SMS-Forbid scans all the positions of the n sequences of Y. For each position it considers all the ℓ-patterns defined by the corresponding ℓ-mer for $3 \leq \ell \leq p$. The number of elements of all the corresponding lattices is bounded by 2^{p+1}. Processing one ℓ-pattern x consists in looking if x is in *Res*, checking if x contains minimal forbidden patterns and searching x in the n sequences of Y. Looking if x is included in *Res* can be done in $O(|x|)$ time using a trie for *Res*. Checking if x contains minimal forbidden patterns consists in using an algorithm for searching a single pattern with wildcard symbols in a text with wildcard symbols for every pattern in $\mathcal{T}$. This can be done in $O(|\mathcal{T}||x|)$. The search of one ℓ-pattern x in one string y of Y consists in spelling all the solid factors of x into the indexing structure of y. When a solid factor u of length $|u|$ is searched in y, it can be done in $O(|u| \log \sigma)$ with the suffix tree of y and in $O(|u| + \log |y|)$ with the suffix array and the Longest Common Prefix array of y (see [3]). However the search for each solid factor can return a list of positions of size $O(|y|)$.

Thus the time complexity of this step is $O(|x| \log \sigma + p|y|)$ with suffix trees and is $O(|x| + \log |y| + p|y|)$ with suffix arrays.

The time complexity for building the indexing structures for all the n sequences of Y is $O(N)$. Altogether the time complexity of the algorithm SMS-Forbid is $O(N2^p\sigma^p((p + \log m) + pm)))$ where m is the maximal length of the sequences of Y.

The algorithm requires to build and traverse all the lattices corresponding to ℓ-patterns. An array of size $2^{\ell-2}$ is used to mark the nodes of each lattice. Thus the space complexity for the lattices is $O(2^p)$. The space complexity of the indexing structures for all the n sequences of Y is $O(N)$. In the worst case the size of *Res* and $\mathcal{T}$ is bounded by $|\Sigma|^p$. Altogether the space complexity of the algorithm SMS-Forbid is $O(N + 2^p + |\Sigma|^p)$.

A similar analysis can be done for algorithm SMS-H-Forbid. The complexities are summarized in table 1.

Table 1. Time and space complexities of algorithms SMS-Forbid and SMS-H-Forbid

Algorithm	Time	Space
SMS-Forbid (suffix trees)	$O(N2^p\sigma^p(p \log \sigma + pm)))$	$O(N + 2^p + \sigma^p)$
SMS-Forbid (suffix arrays)	$O(N2^p\sigma^p((p + \log m) + pm))$	$O(N + 2^p + \sigma^p)$
SMS-H-Forbid	$O(N2^p\sigma^p(pm))$	$O(\sigma^2 p + 2^p + \sigma^p)$

Next, we will denote by SMS-ST the SMS-Forbid algorithm using suffix trees, SMS-SA the SMS-Forbid algorithm using suffix arrays and SMS-Hash the SMS-H-Forbid algorithm.

4 Experimental Results

We implemented our algorithms in C. Experimental results were conducted on pseudo-randomly generated data using the KISS generator [10] on two alphabet sizes 4 (to simulate DNA sequences) and 20 (to simulate protein sequences). All the experiments were performed on a P4.2 GHz machine with 3GB memory.

We measured the computing time and the space consumption of these algorithms for different values of m, n, p and q. All these results have been obtained through computing an average on 15 draws. Fig. 2 and 3 show the variation of computing time in function of m and n for $p = 7$ and $q = 10$ on DNA alphabet. The X-axis represents m and the Y-axis represents the computing time t in seconds. We note that we have used a nonlinear method to implement the suffix arrays for the input strings but which is fast in practice.

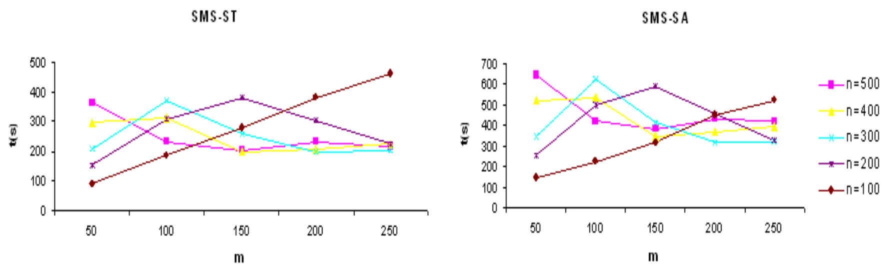

Fig. 2. Computing time in seconds of SMS–FORBID using Suffix Trees (left) and using Suffix Arrays (right)

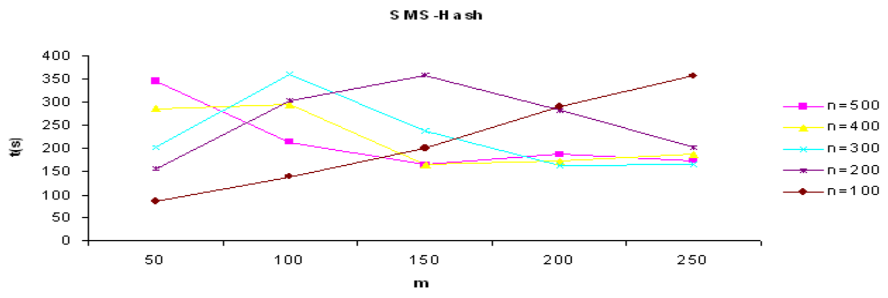

Fig. 3. Computing time in seconds of SMS-H-FORBID

As we can notice, SMS-H-FORBID performs better than SMS-FORBID in terms of running time since it uses a hash table instead of an indexing structure for the strings in Y (see Fig. 4). Indeed, the hash table used in SMS-H-FORBID makes easier the search of candidate patterns.

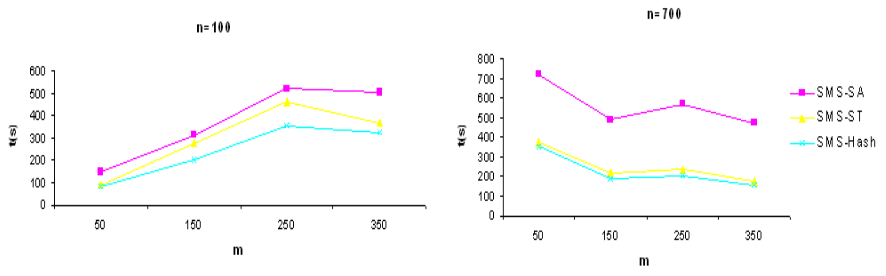

Fig. 4. Computing time for $n = 100$ (left) and $n = 700$ (right) with DNA alphabet

We have also computed the memory consumption in MB of the specific data structures used by each algorithm (see table 2). This table show that the SMS-Forbid algorithm using suffix arrays consumes less memory than the other algorithms.

We have employed our algorithms on various pseudo-randomly generated protein sequences (see Fig. 5). The experimental study show that SMS-H-Forbid algorithm is a faster solution to SMP when the input strings are built with protein alphabet, despite the fact that its worst case time complexity is bigger than the one of SMS-Forbid.

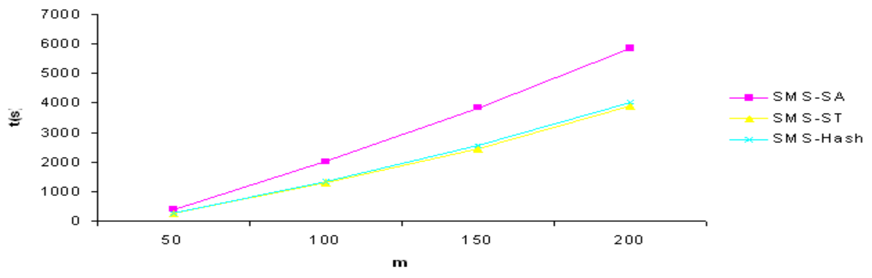

Fig. 5. Computing time in seconds with protein alphabet for $n = 100$

Fig. 6 shows the results obtained for SMS-H-Forbid in function of q and n for $m = 100$. As we can notice, the curves have a bell-like shape. Indeed, for a low value of q, the possibility of finding quickly the motifs is higher. When q is getting closer to n, the number of detected minimal forbidden motifs increases. So, the possibility to reduce the computing time is also higher.It is also remarkable to note that the peak of the curves obtained with protein alphabet is for a low value of the quorum. Indeed, the size of the alphabet makes the possibility of finding simple motifs that have occurrences in at least q strings decreases rapidly as q increases.

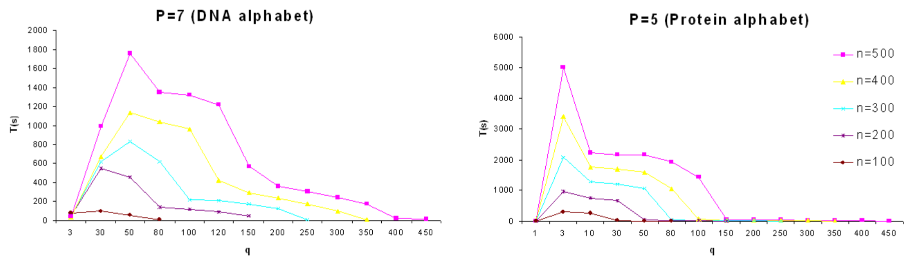

Fig. 6. Computing time in seconds of SMS-H-FORBID in function of q and n with DNA alphabet for $p = 7$ (left) and with protein alphabet for $p = 5$ (right)

Table 2. space consumption in MB with DNA alphabet for $n = 500$

Algorithm \ m	50	150	250	350
SMS-SA	0.29	0.73	1.17	1.61
SMS-ST	1.6	4.8	8.02	11.23
SMS-HASH	0.47	1.5	2.52	3.55

5 Conclusion

In this paper, we have proposed an experimental evaluation of algorithms dealing with SMP. It is remarkable to note that our algorithms have the ability to identify the most specific motifs earlier in the search process, which allows less specific motifs to be pruned. By using this technique, we avoid the sorting step of SMS algorithm presented in [12] and we filter motifs respecting the generality relation between patterns. Experimental results show that our algorithms are well performing in practice. To have a more efficient solution in terms of space consumption, we are studying how to determine a new algorithm using the FM-Index [7] which is a data structure based on the Burrows-Wheeler Transform [1].

References

1. Adjeroh, D., Bell, T., Mukherjee, A.: The Burrows-Wheeler Transform. Springer, Heidelberg (2008)
2. Chin, F.Y.L., Leung, H.C.M.: Voting algorithm for discovering long motifs. In: Proceedings of Asia-Pacific Bioinformatics Conference, pp. 261–272 (2005)
3. Crochemore, M., Hancart, C., Lecroq, T.: Algorithms on Strings. Cambridge University Press, Cambridge (2007)
4. El Falah, T., Elloumi, M., Lecroq, T.: Motif finding algorithms in biological sequences. In: Algorithms in Computational Molecular Biology: Techniques, Approaches and Applications, Wiley Book Series on Bioinformatics: Computational Techniques and Ingeneering, pp. 387–398. Wiley-Blackwell, John Wiley and Sons Ltd., New Jersey, USA (2011)

5. El Falah, T., Lecroq, T., Elloumi, M.: SMS-Forbid: an efficient algorithm for simple motif problem. In: Proceedings of the ISCA 2nd International Conference on Bioinformatics and Computational Biology, Honolulu, Hawai, pp. 121–126 (2010)
6. El Falah, T., Lecroq, T., Elloumi, M.: An efficient motif search algorithm based on a minimal forbidden patterns approach. In: Proceedings of the 5th International Conference on Practical Applications of Computational Biology and Bioinformatics. AISC, pp. 175–182. Springer, Heidelberg (2011)
7. Ferragina, P., Manzini, G.: Opportunistic data structures with applications. In: Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS), Redondo Beach, CA, USA, pp. 390–398 (2000)
8. Floratos, A., Rigoutsos, I.: On the time complexity of the teiresias algorithm. Technical report, Research Report RC 21161 (94582), IBM T.J. Watson Research Center (1998)
9. Leung, H.C.M., Chin, F.Y.L.: An efficient algorithm for the extended (l,d)-motif problem, with unknown number of binding sites. In: Proceedings of the Fifth IEEE Symposium on Bioinformatics and Bioengineering (BIBE 2005), pp. 11–18 (2005)
10. Marsaglia, G., Zaman, A.: The kiss genrator. Technical report, Department of Statistics, Florida State University, Tallahassee, FL, USA (1993)
11. Rajasekaran, S., Balla, S., Huang, C.H.: Exact algorithms for planted motif challenge problems. Journal of Computational Biology 12(8), 1117–1128 (2005)
12. Rajasekaran, S., Balla, S., Huang, C.-H., Thapar, V., Gryk, M., Maciejewski, M., Schiller, M.: High-performance exact algorithms for motif search. Journal of Clinical Monitoring and Computing 19, 319–328 (2005)
13. Sagot, M.-F.: Spelling approximate repeated or common motifs using a suffix tree. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 111–127. Springer, Heidelberg (1998)
14. Styczynski, M.P., Jensen, K.L., Rigoutsos, I., Stephanopoulos, G.N.: An extension and novel solution to the (l,d)-motif challenge problem. Genome Informatics 15(2), 63–71 (2004)