Extraction of Infrequent Simple Motifs from a Finite Set of Sequences using a Lattice Structure

Tarek El Falah[1,2] Thierry Lecroq[2]
Mourad Elloumi[1]
[1]*Laboratory of Technologies of Information and Communication and Electrical Engineering, Higher School of Sciences and Technologies of Tunis, 1008 Tunis, Tunisia*
[2]*University of Rouen, LITIS EA 4108, 76821 Mont-Saint-Aignan Cedex, France*

Abstract

In this paper we present a method for finding infrequent simple motifs in a finite set of sequences. The method uses a lattice structure and minimal forbidden patterns. It is based on a method for solving the Simple Motif Problem.

keywords: lattice, algorithms, strings, infrequent motifs, patents

INTRODUCTION

One of the problems arising in the analysis of sequences is the discovery of sequence similarity by finding common motifs. Several versions of the *motif finding problem*, which consists in finding substrings that are more or less conserved in a set of strings, have been proposed for dealing with this problem [5]. For each version, numerous algorithms have been developed [2, 8, 11, 12, 13, 14, 15, 16, 18]. These algorithms share in common the fact that they all extract the frequent motifs. 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.

The SMS-FORBID and SMS-H-FORBID algorithms by El Falah *et al*. [7, 6] are fast solutions to SMP. 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 general approach on which are based these algorithms, maintains a set of minimal forbidden patterns that occur in less than q sequences in order to not search for any pattern that contains a factor that has already been unsuccessfully searched.

In this paper, we address the extraction of infrequent simple motifs. An infrequent simple motif is a simple motif that occurs in less than q sequences. We use a similar approach as in SMS-FORBID and SMS-H-FORBID algorithms based on a lattice structure. This lattice with a generality relation between patterns is a fundamental structure to obtain the rare motifs. It is remarkable to note that the set of forbidden patterns that occur in at least q sequences represents the complementary of the set of infrequent simple motifs. Thus our method finds in the same time the most general infrequent motifs and the most specific frequent ones.

When dealing with biological sequences, this could help biologists to understand mechanisms of the biological processes in which these motifs are involved. For example, searching rare motifs is an important task to identify absent words in genomic sequences [9, 10, 20]. Moreover, to identify proteins involved in parasitism, it is important to search the infrequent motifs [19]. Indeed, given a set of protein sequences known to be involved in a common biological system (positive set) and a set of

protein sequences known not to be involved in that system (negative set) our method is able to identify motifs that are frequent in positive sequences while infrequent in negative ones.

In a similar way to our approach, we note that there are some studies in data mining addressing the extraction of rare itemsets and have medicine and biology as fields of usage [17] despite the fact that most studies have concentrated on frequent itemsets.

We organize the rest of the paper as follows: In the first section, we present some useful definitions and notations. In the second section, we present the lattice based approach to find rare motifs. In the third section, we present the algorithm in more details. In the fourth section, we give our conclusion and present the current and future development.

PRELIMINARIES

A *simple motif* is a string built from an alphabet $\Sigma \cup \{\texttt{?}\}$ that cannot begin or end with `?`, where Σ is a finite set of symbols and $\texttt{?} \notin \Sigma$ 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 $\Sigma \cup \{\texttt{?}\}$ 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 $z_1[i] = \texttt{?}$ or $z_2[i] = \texttt{?}$

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 $z_2[i] = \texttt{?} \Rightarrow z_1[i] = \texttt{?}$ and $z_2[i] = a \Rightarrow (z_1[i] = a$ or $z_1[i] = \texttt{?})$ 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.

A LATTICE BASED APPROACH TO FIND INFREQUENT MOTIFS

In this section, we present the lattice based approach to find infrequent simple motifs.

Given a set Y of n strings, a quorum $q \leq n$ and an integer p, a pattern of length at most p is called rare (or infrequent) motif when it occurs in less than q strings. The algorithm output is the set of the more general infrequent motifs.

A pattern z of length at most p is said to be a minimal forbidden pattern if it occurs in at least q strings and all the patterns beginning and ending with a solid symbol and containing z as factor do not occur in at least q strings.

Our approach is based on maintaining a set of minimal forbidden patterns that occur in at least q strings in order to not search for any pattern that contains a factor that has already been unsuccessfully searched. In fact, when we search the infrequent motifs, we maintain also the minimal forbidden patterns. Hence our lattice based approach allows to find at the same time the frequent and also the rare motifs.

Let us explain the general approach based on a lattice structure: 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 Figure 1: the minimal element is $x = \texttt{CAGAT}$ and the maximal element is $\texttt{C?}^{\ell-2}\texttt{T}$).

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.

The infrequent patterns are counted as follows: when a ℓ-pattern z is considered, if it has already been output or

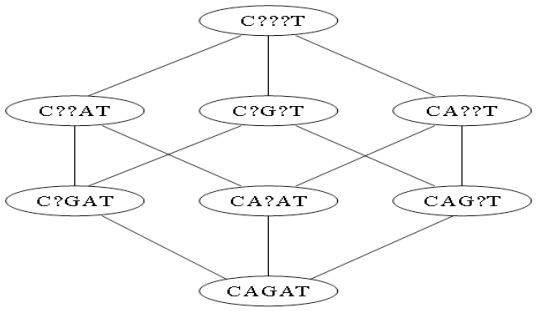

Figure 1: A 5-mer $x =$ `CAGAT` lattice

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 does not occur in at least q strings it is considered as an infrequent pattern and all its predecessors in the lattice are not considered since they are more general. On the contrary if it occurs in at least q strings it is added to the set of minimal forbidden patterns. By adopting this approach when we count a number of occurrences of z less than q, the considered ℓ-pattern z is an infrequent one. Then we respect the generality relation between patterns to output the most general infrequent motifs.

It is remarkable to note that 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 predecessors in the lattice, that are more specific, 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.

The techniques used in this approach 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 outputs both the more general rare motifs and the frequent motifs which occur in at least q strings of Y.

ALGORITHM

We will give a top-down detailed presentation of the algorithm. It builds the set *Res* of searched infrequent motifs of length at most p contained in less than q strings and uses a set $\mathcal{T}$ of minimal patterns that are 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 predecessors 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 less than 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 $x[1\,..\,\ell-2]$ 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 merged 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 general patterns is found after a more specific pattern. Then the set of resulting patterns has to be filtered.

An alternative for using an indexing structure is to use an hash table as follows. 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. 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.

COMPLEXITIES

We will now give the complexities of both methods.

The algorithm using an indexing structure called RARE-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 RARE-FORBID is $O(N 2^p \sigma^p((p + \log m) + pm)))$ 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 RARE-FORBID is $O(N + 2^p + |\Sigma|^p)$.

A similar analysis can be done for algorithm using an hash table called RARE-H-FORBID. The complexities are summarized in table 1.

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

Table 1: Time and space complexities of algorithms RARE-FORBID and RARE-H-FORBID.

CURRENT AND FUTURE DEVELOPMENT

In this paper, we have presented an approach based on a lattice structure to search infrequent motifs in a set of strings. This approach is based on maintaining a set of minimal forbidden patterns which allows less specific motifs to be pruned.

Our approach allows to find in the same time the frequent and also the infrequent patterns. When dealing with biological sequences, it could help biologists to have the most pertinent motifs and understand mechanisms of the biological processes in which the rare motifs are involved.

As future work, we aim to develop new algorithms dealing with the problem of finding absent words in a set of biological sequences [1, 4, 10, 20].

ACKNOWLEDGEMENT

The authors would like to thank all colleagues who have done work in this field, and apologize to the colleagues whose work in the field was not directly cited.

CONFLICT OF INTEREST

The author has no conflicts of interest that are directly relevant to the content of this review.

References

[1] M. P. Beal, F. Mignosi, and A. Restivo. Minimal forbidden words and symbolic dynamics. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science, STACS96, Vol.1046 of Lecture Notes in Computer Science*, pages 555–566, Grenoble, France, 1996.

[2] F. Y. L. Chin and H. C. M. Leung. Voting algorithm for discovering long motifs. In *Proceedings of Asia-Pacific Bioinformatics Conference*, pages 261–272, 2005.

[3] M. Crochemore, C. Hancart, and T. Lecroq. *Algorithms on Strings*. Cambridge University Press, 2007.

[4] M. Crochemore, F. Mignosi, and A. Restivo. Automata and forbidden words. *Information Processing Letters*, 67:111–117, 1998.

[5] T. El Falah, M. Elloumi, and T. Lecroq. 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*, pages 387–398. Wiley-Blackwell, John Wiley and Sons Ltd., New Jersey, USA, 2011.

[6] T. El Falah, T. Lecroq, and M. Elloumi. 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, Advances in Intelligent and Soft-Computing Seies, Springer-Verlag, Berlin, Heidelberg, Germany*, Salamanca, Spain.

[7] T. El Falah, T. Lecroq, and M. Elloumi. SMS-FORBID: an efficient algorithm for simple motif problem. In *Proceedings of the ISCA 2nd International Conference on Bioinformatics and Computational Biology*, pages 121–126, Honolulu, Hawai, 2010.

[8] A. Floratos and I. Rigoutsos. On the time complexity of the teiresias algorithm. Technical report, Research Report RC 21161 (94582), IBM T.J. Watson Research Center, 1998.

[9] S.P. Garcia, A.J Pinho, J.M.O.S. Rodrigues, C.A.C. Bastos, and P.J.S.G Ferreira. Minimal absent words in prokaryotic and eukaryotic genomes. *PLoS ONE*, 6(1):e16065, 2011.

[10] J. Herold, S. Kurtz, and R. Giegerich. Efficient computation of absent words in genomic sequences. *BMC Bioinformatics*, 9:167, 2008.

[11] H. C. M. Leung and F. Y. L. Chin. 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'05)*, pages 11–18, 2005.

[12] A. Price, S. Ramabhadran, and P. A. Pevzner. Finding subtle motifs by branching from sample strings. *Bioinformatics*, 1(1):1–7, 2003.

[13] S. Rajasekaran, S. Balla, and C. H. Huang. Exact algorithms for planted motif challenge problems. *Journal of Computational Biology*, 12(8):1117–1128, 2005.

[14] S. Rajasekaran, S. Balla, C.-H. Huang, V. Thapar, M. Gryk, M. Maciejewski, and M. Schiller. High-performance exact algorithms for motif search. *Journal of Clinical Monitoring and Computing*, 19:319–328, 2005.

[15] M. F. Sagot. Spelling approximate repeated or common motifs using a suffix tree. In *C. L. Lucchesi and A. V. Moura, editors, LATIN'98: Theoretical Informatics, volume 1380 of Lecture Notes in Computer Science, Springer-Verlag*, pages 111–127, 1998.

[16] M.P. Styczynski, K. L. Jensen, I. Rigoutsos, and G.N. Stephanopoulos. An extension and novel solution to the (l,d)-motif challenge problem. *Genome Informatics*, 15(2):63–71, 2004.

[17] L. Szathmary, S. Maumus, P. Petronin, Y. Toussaint, and A. Napoli. Vers l'extraction de motifs rares. In *EGC*, pages 499–510, 2006.

[18] S. Thota, S. Balla, and S. Rajasekaran. Algorithms for motif discovery based on edit distance. Technical report, BECAT/CSE-TR-07-3, 2007.

[19] C. Vens, E. Danchin, and M.N. Rosso. Identifying proteins involved in parasitism by discovering degenerated motifs. In *4th International Workshop on Machine Learning in Systems Biology*, pages 81–84, 2010.

[20] Z.D. Wu, T. Jiang, and W.J. Su. Efficient computation of shortest absent words in a genomic sequence. *Inf. Process. Lett.*, 110:596–601, 2010.