SMS-Forbid: An efficient algorithm for Simple Motif Problem

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

[1]Research Unit of Technologies of Information and Communication,
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

Finding common motifs from a set of strings coding biological sequences is an important problem in Molecular Biology. Several versions of the motif finding problem have been proposed in the literature and for each version, numerous algorithms have been developed. However, many of these algorithms fall under the category of heuristics.

In this paper, we concentrate on the Simple Motif Problem (SMP) and we propose an exact algorithm, called SMS-Forbid, for this version of motif finding problem. SMS-Forbid make use of less time and space than the known exact algorithms.

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:

- *Planted (l,d)-Motif Problem* (PMP) [1, 7, 8]
- *Extended (l,d)-Motif Problem* (ExMP) [5, 12]
- *Edited Motif Problem* (EdMP) [9, 10, 13]
- *Simple Motif Problem* (SMP).
  Actually, concerning SMP there are two different versions:
  - the one decribed by Floratos and Rigoutsos [3],
  - and the one decribed by Rajasekaran *et al.* [9]

In this paper, we are interested in a more general version of the SMP. Let us now give some definitions related to the SMP: A *simple motif* has the same definition as in [3, 9], it is a string built from an 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. Symbols of Σ are said to be solid while the wildcard symbol `?` is said to be non-solid. A string of u solid symbols is called a u-mer. The length of a simple motif is the number of the symbols that constitute this motif, including the wildcard symbols. The class of the simple motifs where each simple motif is of length u and has exactly v wildcard symbols will be denoted by (u, v)-class. Now let us formulate our version of SMP: Let $Y = \{y_1, y_2, \dots, y_n\}$ be a set of strings built from an alphabet Σ, $p > 0$ be an integer and $q \leq n$ be a quorum, find all the simple motifs of length at most p that occurs in at least q sequences of Y.

Despite many efforts, the motif finding problem remains a challenge for both Computer Scientists and Biologists. Indeed, on one hand, the general version of this problem is NP-hard [9]. On the other hand, our incomplete and fuzzy understanding of a number of biological mechanisms does not help us to provide good models for this problem. In this paper, we propose a new approach to search motifs and we present an efficient algorithm for the SMP. Our algorithm, called SMS-Forbid, contains techniques that reduce the number of patterns to be searched for. Hence, it should help the biologist to identify important motifs.

The rest of this paper is organized as follows: In section 2, we present some related works. In section 3, we explain a new approach related to SMP. In section 4, we give a detailed algorithm based on the approach presented in the previous section. In section 5, we compute the complexity of the algorithm by giving the worst case time complexity, the worst case space complexity and the average complexity. In section 6, we make a conclusion to this paper.

2 Related works

In this section, we present two algorithms, Teiresias [3] and SMS [9].

Algorithm Teiresias [3] addresses a problem that is close to our version of SMP. So, let us first give some definitions related to this problem: A simple motif m is called $\langle \ell, d\rangle$-motif if every simple motif of m of length at least ℓ contains at least d symbols belonging to Σ. An elementary $\langle \ell, d\rangle$-motif is a substring of length ℓ which contains exactly d symbols belonging to Σ. A simple motif m' is said to be more specific than a simple motif m if m' can be obtained from m by changing one or more ?'s of m into symbols belonging to Σ and/or by adding one or more symbols belonging to Σ to the extremities of m. A simple motif m is said to be maximal if there is no simple motif m' that is more specific than m and which appears in more strings of S than m. Now let us formulate the problem addressed by algorithm Teiresias [3]: Let $Y = \{y_1, y_2, \ldots, y_n\}$ be a set of strings built from an alphabet Σ and ℓ, d and q be three positive integers, find all maximal $\langle \ell, d\rangle$-motifs in Y that occur in at least q distinct strings of Y. Teiresias algorithm operates in two steps.

- During the first step, it identifies the elementary $\langle \ell, d\rangle$-motifs in the strings of Y.
- Then, during the second step, it superposes overlapping elementary $\langle \ell, d\rangle$-motifs, identified during the first step, to obtain larger $\langle \ell, d\rangle$-motifs ones. The obtained $\langle \ell, d\rangle$-motifs in Y that are maximal and that occur in at least q distinct strings of Y are solutions to the addressed problem.

The time complexity of algorithm Teiresias is $\Omega(\ell^d N \log N)$.

As we said earlier, SMS algorithm [9] does not address the same version of SMP. The version of SMP defined in [9] is as follows: Let $Y = \{y_1, y_2, \ldots, y_n\}$ be a set of strings built from an alphabet Σ and $\ell > 0$ be an integer, find all the simple motifs of length at most ℓ with anywhere from 0 to $\lfloor \ell/2 \rfloor$?'s and for each simple motif give its number of occurrences in the strings of Y.

SMS algorithm operates as follows: For each (u, v)-class, $0 \le |u| \le \ell$ and $0 \le v \le \lfloor \ell/2 \rfloor$:

- First, it extracts all the substrings of length u in the strings of Y.
- Then, it sorts all the substrings of length u extracted during the first step only with respect to the non-wildcard positions. The authors employ in this phase the radix sort [4].
- Finally, it scans through the sorted list and count the number of times each simple motif appears.

The time complexity of this algorithm is $O(\ell^{\ell/2} N)$, where $N = \sum_{i=1}^{n} |y_i|$.

Next, we present a new algorithm for the SMP. The generation of u-mers in the input sequences is performed in a first step as in the SMS algorithm, but it is useless to generate them all and to sort them in order to eliminate duplicates. We propose to make a more clever generation which allows us to store all existing patterns in the input sequences without sorting. Moreover, we propose a more efficient approach to count these patterns.

3 A new approach

The inputs of the algorithm are a set Y of n sequences, a quorum $q \le n$ and an integer p.

The algorithm outputs the set of motifs of length at most p that occurs in at least q sequences.

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

For each position on the input sequences, we use all the ℓ-windows for $3 \le \ell \le p$. Each ℓ-window defines an ℓ-mer.

Each ℓ-mer x defines a set of ℓ-patterns X. At each position of each 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,

$$z[i] = \begin{cases} x[i] \\ \text{or} \\ \texttt{?} \end{cases}$$

for $2 \le i \le \ell - 2$ and $z[1] = x[1]$ and $z[\ell] = x[\ell]$.

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. Formally $z_2[i] = \texttt{?} \Rightarrow z_1[i] = \texttt{?}$ for $1 \le i \le \ell$.

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

Each node of the lattice is thus represented by 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

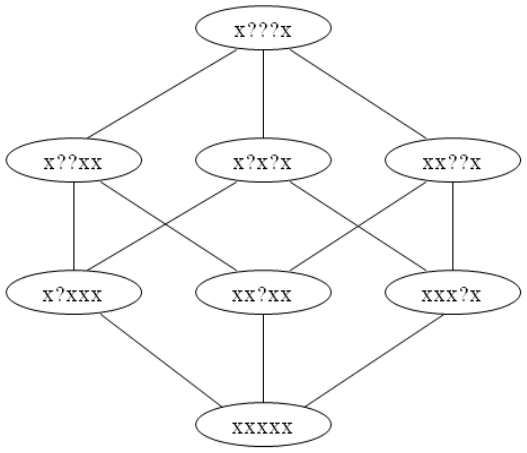

Figure 1: 5-mer lattice

- it is more general than an output pattern

then it is disregarded otherwise it is searched in every sequences of Y. Then if it occurs in at least q sequences 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 sequences 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 [9], the new approach does not search for all the ℓ-patterns generated from the n sequences 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 sequences 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 sequence y of Y is done using an indexing structure of y which can be done in a proportional time to ℓ.

Furthermore the new approach only outputs the more specific motifs that occur in at least q sequences of Y. This should help the biologist to identify important motifs.

The algorithm together with all the different data structures will be presented in details in the next section.

```
SMS-Forbid($Y, p, q$)
1 $Res \leftarrow \emptyset$
2 $\mathcal{T} \leftarrow \emptyset$
3 for $j \leftarrow 1$ to $n$ do
4    for $i \leftarrow 1$ to $|y_j| - 2$ do
5       for $\ell \leftarrow 3$ to $\min\{p, |y_j| - i\}$ do
6          for $k \leftarrow 0$ to $\ell - 2$ do
7             Breadth-First-Search(
                 $y_j[i \mathinner{.\,.} i + \ell - 1], 2, k$)
8 return $Res$
```

Figure 2: The main algorithm.

4 Detailed algorithm

We will give a top-down detailed presentation of the algorithm introduced in Section 3. The main algorithm is given in Figure 2. It builds the set *Res* of searched motifs of length at most p contained in at least q sequences and uses a set $\mathcal{T}$ of minimal patterns that are not contained in at least q sequences. It scans the n sequences of Y. For each position of each sequence it considers all the ℓ-windows for $3 \leq \ell \leq p$ (at the end of the sequences it may be less than p). For each ℓ-mer x defined by each ℓ-window, the breadth-first search of the lattice is performed level by level by a recursive algorithm fed by x, the first position where a wildcard symbol can be inserted and the number of wildcard symbols (originally 0).

The breadth-first search of the lattice is performed by the recursive algorithm given in Figure 3. Its inputs are an ℓ-pattern x, two integers *pos* and i where i is the number of wildcard symbols to be inserted from position *pos* to position $|x| - 1$.

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 (see Figure 4).

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. Since there does not exist any efficient solution to this problem when the sum of the lengths of the patterns in the set is large (see [6], p. 96) we use the best known algorithm for searching a single pattern with wildcard symbols in a text with wildcard

```
Breadth-First-Search($x, pos, i$)
  ▷ set $i$ wildcard symbols in $x$
  ▷ from position $pos$ to $|x| - 1$
if $i > 0$ then
  if $|x| - pos > i$ then
    Breadth-First-Search($x, pos + 1, i$)
  $c \leftarrow x[pos]$
  $x[pos] \leftarrow$ ?
  Breadth-First-Search($x, pos + 1, i - 1$)
  $x[pos] \leftarrow c$
else if $x$ is not marked then
    if $x \in Res$ then
      Depth-First-Search($x$)
    else if $\mathcal{T} \not\subseteq x$ then
        $k \leftarrow$ Count($x, Y$)
        if $k \geq q$ then
          $Res \leftarrow Res \cup \{x\}$
          Depth-First-Search($x$)
        else $\mathcal{T} \leftarrow \mathcal{T} \cup \{x\}$
```

Figure 3: Generation of the ℓ-pattern corresponding to an ℓ-mer using a breadth-first search of the associated lattice.

```
Depth-First-Search($x$)
if $x$ is not marked then
  mark $x$
  for each successor $s$ of $x$ do
    Depth-First-Search($s$)
```

Figure 4: Mark all the successors of the ℓ-pattern x in the lattice.

symbols [11] for every pattern in $\mathcal{T}$.

If x does not contain any minimal forbidden pattern then it is searched in all the sequences of Y (see Figures 5 and 6). If it occurs in at least q sequences, it is added to Res and all its successors are marked using a depth-first search (see Figure 4). 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 $x[1\,..\,\ell-1]$ 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 algorithm for counting the number of sequences that contain an ℓ-pattern is given in Figure 5. For every sequence of Y, it calls a procedure Search given in Figure 6. This procedure takes as input an

```
Count($x, Y$)
$k \leftarrow 0$
for $j \leftarrow 1$ to $n$ do
  $k \leftarrow k +$ Search($x, y_j$)
  if $k + n - j < q$ then
    break
return $k$
```

Figure 5: Count the number of strings of Y that contain motif x.

```
Search($x, y$)
$u_1 v_1 u_2, v_2 \cdots u_{m-1} v_{m-1} u_m \leftarrow x$
$R \leftarrow Pos(u_1, y)$
$i \leftarrow 1$
while $i < m$ and $R \neq \emptyset$ do
  $i \leftarrow i + 1$
  $R \leftarrow$ Merge($R, Pos(u_i, y), |u_1 \cdots v_{i-1}|$)
if $i < m$ then
  return 0
else return 1
```

Figure 6: Search motif x in string y.

ℓ-pattern x and a sequence y of Y. It considers a factorization of an ℓ-pattern x as follows:

$$x = u_1 v_1 u_2, v_2 \cdots u_{m-1} v_{m-1} u_m$$

where $u_i \in \Sigma^*$ for $1 \leq i \leq m$ and $v_j \in \{?\}^*$ for $1 \leq j \leq m-1$ (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 (see Figure 7). 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 linked lists. 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, see [2]).

5 Complexities

5.1 Time complexity

The algorithm SMS-Forbid given Figure 2 scans all the positions of the n sequences of Y. For each

```
Merge($R, T, d$)
 1  $V \leftarrow \emptyset$
 2  $r \leftarrow$ Dequeue($R$)
 3  $t \leftarrow$ Dequeue($T$)
 4  while $R \neq \emptyset$ and $T \neq \emptyset$ do
 5      if $r + d < t$ then
 6          $r \leftarrow$ Dequeue($R$)
 7      else if $r + d > t$ then
 8              $t \leftarrow$ Dequeue($T$)
 9          else $\triangleright$ $r + d = t$
10              $V \leftarrow V \cup \{r\}$
11              $r \leftarrow$ Dequeue($R$)
12              $t \leftarrow$ Dequeue($T$)
13  return $V$
```

Figure 7: Merge the two lists R and T according to length d.

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 (see algorithm Count in Figure 5) consists in:

1. looking if x is in *Res*;

2. checking if x contains minimal forbidden patterns;

3. 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 $O(|\mathcal{T}| \times |x|)$.

The search of one ℓ-pattern x in one sequence y of Y (see algorithm Search in Figure 6) consists in spelling all the solid factors of x into the indexing structure of y. This is realized by the different calls to function *Pos* and can be done in $O(\ell)$ time overall. Each call to function *Pos* can return a list of positions of size $O(|y|)$. Thus the time complexity of algorithm Search is $O(|x| + |y|)$.

The time complexity for building the indexing structures for all the n sequences of Y is $O(N)$ where N is the total length of the n sequences of Y.

Altogether the time complexity of the algorithm SMS-Forbid is $O(N \times 2^p \times |\Sigma|^p \times (p + m))$ where m is the maximal length of the sequences of Y.

5.2 Space complexity

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 $O(N)$.

Each list of positions used by the algorithms Search(x, y) and Merge requires a linear space with respect to the length of y.

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 $O(N + 2^p + |\Sigma|^p)$.

6 Conclusion

In this paper, we have presented a new algorithm, called SMS-Forbid, to solve the Simple Motif Problem (SMP).

On one hand, SMS-Forbid does not search all the ℓ-patterns generated from the input sequences but it searches the more specific patterns. By using this technique, we avoid the sorting step of SMS algorithm presented in [9].

On the other hand, it maintains a set of minimal forbidden patterns that do not occur in at least q sequences in order to not search for any pattern that contains a factor that has already been unsuccessfully searched.

Moreover the new approach only outputs the more specific motifs and so that it identifies important motifs.

SMS-Forbid have the potential of performing well in practice by reducing the number of patterns to be searched for.

References

[1] 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.

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

[3] 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.

[4] E. Horowitz, S. Sahni, and S. Rajasekaran. *Computer Algorithms.* W. H. Freeman Press, 1998.

[5] 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.

[6] G. Navarro and M. Raffinot. *Flexible Pattern Matching in Strings.* Cambridge University Press, 2002.

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

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

[9] 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.

[10] M. F. Sagot. Spelling approximate repeated or common motifs using a suffix tree. In *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.

[11] W. F. Smyth and S. Wang. An adaptive hybrid pattern-matching algorithm on indeterminate strings. *International Journal on Foundations of Computer Science*, 2009. to appear.

[12] 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.

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