Bases of Motifs for Generating Repeated Patterns with Wild Cards

Nadia Pisanti, Maxime Crochemore, Roberto Grossi, and Marie-France Sagot

Abstract—Motif inference represents one of the most important areas of research in computational biology, and one of its oldest ones. Despite this, the problem remains very much open in the sense that no existing definition is fully satisfying, either in formal terms, or in relation to the biological questions that involve finding such motifs. Two main types of motifs have been considered in the literature: matrices (of letter frequency per position in the motif) and patterns. There is no conclusive evidence in favor of either, and recent work has attempted to integrate the two types into a single model. In this paper, we address the formal issue in relation to motifs as patterns. This is essential to get at a better understanding of motifs in general. In particular, we consider a promising idea that was recently proposed, which attempted to avoid the combinatorial explosion in the number of motifs by means of a generator set for the motifs. Instead of exhibiting a complete list of motifs satisfying some input constraints, what is produced is a *basis* of such motifs from which all the other ones can be generated. We study the computational cost of determining such a basis of repeated motifs with wild cards in a sequence. We give new upper and lower bounds on such a cost, introducing a notion of basis that is provably contained in (and, thus, smaller) than previously defined ones. Our basis can be computed in less time and space, and is still able to generate the same set of motifs. We also prove that the number of motifs in all bases defined so far grows exponentially with the quorum, that is, with the minimal number of times a motif must appear in a sequence, something unnoticed in previous work. We show that there is no hope to efficiently compute such bases unless the quorum is fixed.

Index Terms—Motifs basis, repeated motifs.

1 Introduction

Identifying motifs in biological sequences is one of the oldest fields in computational biology. Yet, it remains also very much an open problem in the sense that no currently existing definition of a "motif" is fully satisfying for the purposes of accurately and sensitively identifying the biological features that such motifs are supposed to represent. Among the most difficult to model are binding sites, as they are often quite degenerate. Indeed, variability may be considered part of their function. Such variability translates itself into changes in the motif, mostly substitutions, that do not affect the biological function. Two main schools of thought on how to define motifs in biology have coexisted for years, each valid in its own way. The first works with a statistical representation of motifs, usually given in the form of what is called in the literature a PSSM ("Position Specific Scoring Matrix" [9], [11], [13], [12] or a profile which is one type of PSSM). Interesting PSSMs are those that have a high information value (measured, for instance, by the relative entropy of the corresponding matrix). The second school defines a motif as a consensus [4], [24]. A motif is therefore a pattern that appears repeatedly, in general, approximately, that is, up to a certain number of differences (most often substitutions only) in a sequence or set of sequences of interest.

It is generally accepted that PSSMs are more appropriate for modeling an already known (in the sense of well-characterized) biological feature for the purpose of then identifying other occurrences of the feature, even though the false positive rate of this further identification remains very high. Identifying the PSSM itself *ab initio* is still, however, a difficult problem, particularly for large data sets or when the amount of noise may be high. The methods used are also no guarantee heuristics, leaving an uncertainty as to whether motifs that are statistically as meaningful as those reported have not been missed.

On the other hand, formulating the problem of identifying approximate motifs as patterns enables one to address the motif identification problem in an exhaustive fashion, even though the algorithmic complexity of the problem remains relatively high, and the model may appear more limited than PSSMs. Because of the lower algorithmic complexity of identifying repeated patterns, the model may, however, be made more complex and biologically pertinent in other ways. One could think of introducing motifs composed of various different submotifs separated by variable-length distances that may then also be found in a relatively efficient way [14]. Motifs presenting such a high level of combinatorial complexity are indeed frequent, particularly in eukaryotes. Exhaustively seeking for approximately repeated patterns may however have the drawback of producing many "solutions," that is, many motifs. In fact, the number of motifs identified with this model may be so high (e.g., exponential in the size of the input) that it is as impossible to manage as the initial input sequence(s), even though they provide a first way of

- *N. Pisanti and R. Grossi are with the Dipartimento di Informatica, Università di Pisa, Italy. E-mail: {pisanti, grossi}@di.unipi.it.*
- *M. Crochemore is with the Institut Gaspard-Monge, University of Marne-la-Vallée, France and King's College London. E-mail: maxime.crochemore@univ-mlv.fr.*
- *M.-F. Sagot is with INRIA Rhône-Alpes, Laboratoire de Biométrie et Biologie Évolutive, Université Claude Bernard Lyon 1, France and King's College London. E-mail: marie-france.sagot@inria.fr.*

Manuscript received 14 Mar. 2004; revised 2 Dec. 2004; accepted 16 Feb. 2005; published online 30 Mar. 2005.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-0036-0304.

 Published by the IEEE CS, CI, and EMB Societies & the ACM

structuring such input. Yet, it appeared clear also to any computational biologist working with motifs as patterns that there was further structure to be extracted from the set of motifs found, even when such a set is huge. Furthermore, such a structure could reflect some additional biological information, thus providing additional motivation for inferring it. Doing this is generally addressed by means of clustering, or even by attempting to bring together the two types of motif models (PSSMs and patterns). Indeed, recently researchers have been using pattern detection as a first filter-flavored step toward inferring PSSMs from biological sequences [6]. This seems very promising although much work remains to be done to precisely determine the relation between the two types of models, and to fully explore the biological implications this may have.

Again, each of the two above approaches is valid, but the question remained open whether or not the inner structure of a set of motifs could be expressed in a manner that would be more satisfying from both the mathematical and the biological points of view. Then, in 2000, a paper by Parida et al. [17] seemed to present a way of extracting such an inner structure in a very elegant and powerful way for a particular type of motif. The power of their proposal resided in the fact that the above mentioned structure corresponded to a well-known and precisely defined mathematical object and, moreover, guaranteed that no solution would be lost. Exhaustiveness in relation to the chosen type of motif is also preserved, thus enabling a biologist to draw some conclusions even in the face of negative answers (i.e., when no motifs, or no a priori "expected" motifs are found in a given input), something which PSSM-detecting methods do not allow. The structure is that of a *basis of motifs*. Informally speaking, it is a subset of all the motifs satisfying some input parameters (related, for instance, to which differences between a pattern and its occurrences are allowed) from which it is possible to recover all the other motifs, in the sense that all motifs not in the basis are a combination of some (in general, a few only) motifs in the basis. Such a combination is modeled by simple rules to systematically generate the other motifs with an output sensitive cost [18]. A basis would therefore also provide a way of characterizing the input, which then might be used to compare different inputs without resorting to the traditional alignment methods with all the pitfalls they present. The idea of a basis would fulfill such expectations if its size could be proven to be small enough. The argument [17] seemed to be that, for the type of motifs considered, a compact enough basis could always be found.

The motifs considered in [17] were *patterns* with *wild card* symbols occurring in a given sequence s of n symbols drawn over an alphabet Σ. A wild card symbol is a special symbol "$\circ$" matching any other element[1] For example, the pattern $\texttt{T}\circ\texttt{G}$ matches both TTG and TGG inside $s = \texttt{TTGG}$. Parida et al. focused on patterns which appear at least q times in s for an input parameter $q \geq 2$, called the *quorum*. This may, at first sight, seem an even more restrictive type of motif than patterns in general. It, however, has the merit of capturing one aspect of biological features that current PSSMs in general ignore, or address only in an indirect way. This aspect often concerns isolated positions *inside* a motif that are not part of the biological feature being captured. This is the case, for instance, with some binding sites, particularly at the protein level. Studying patterns with wild cards has a further very important motivation in biology, even when no differences (such as substitutions) are allowed. Indeed, motifs such as these or closely related ones can be used as seeds for finding long repeats and for aligning, pairwise or multiple-wise, a set of sequences or even whole genomes [15], [23].

The basis introduced by Parida et al. had interesting features, but presented some unsatisfying properties. In particular, as we show in this paper, there is an infinite family of strings for which the authors' basis contains $\Omega(n^2)$ motifs for $q = 2$. This contradicts the upper bound of $3n$ for any $q \geq 2$ given in [17]. As a result, the algorithm taking $O(n^3 \log n)$ time, mentioned in [17], for finding the basis of motifs does not hold since it relies on the upper bound of $3n$, thus leaving open the problem of efficiently discovering a basis. A refinement of the definition of basis and an incremental construction in $O(n^3)$ time has recently been described by Apostolico and Parida [2]. A comparative survey of several notions of bases can be found in [22].

Closely following previous work, here we introduce a new definition of basis. The condition for the new basis is stronger than that of [17] and, hence, our basis is included in that of [17] (and is thus smaller) while both are able to generate the *same* set of motifs with mechanical rules. Our basis is moreover symmetric: Given a string s, the motifs in the basis for its reverse $\widetilde{s}$ are the reversals of the motifs in the basis for s. Moreover, the number of motifs in our basis can provably be upper bounded in the worst case by $n - 1$ for $q = 2$ and occur in s a total of $2n$ times at most. However, we reveal an *exponential* dependency on q for the number of motifs in all bases defined so far (i.e., including our basis, Parida's and Pelfrene et al.'s [19]), something unnoticed in previous work. Consequently, *no* polynomial-time algorithm can exist for finding one of these bases with *arbitrary* values of $q \geq 2$.

2 Notation and Terminology

We consider strings that are finite sequences of letters drawn from an alphabet Σ, whose elements are also called *solid characters*. We introduce an additional symbol (denoted by $\circ$ and called *wild card*) that does not belong to Σ and matches any letter; a wild card clearly matches itself. The length of a string t, denoted by $|t|$, is the number of letters and wild cards in t, and $t[i]$ indicates the letter or wild card at position i in t for $0 \leq i \leq |t| - 1$ (hence, $t = t[0]t[1]\cdots t[|t| - 1]$ also noted $t[0..|t| - 1]$).

Definition 1 (pattern). *Given the alphabet Σ, a* pattern *is a string in $\Sigma \cup \Sigma(\Sigma \cup \{\circ\})^*\Sigma$ (that is, it starts and ends with a solid character).*

The patterns are related by the following *specificity relation* $\preceq$.

1. In the literature on sequence analysis and pattern matching, the wild card is often referred to as *do not care* (as it is in the literature on bases of motifs). Therefore, we will use this latter term when referring to the sequence analysis and string matching literature.

Definition 2 ($\preceq$). *For individual characters $\sigma_1, \sigma_2 \in \Sigma \cup \{\circ\}$, we have $\sigma_1 \preceq \sigma_2$ if $\sigma_1 = \circ$ or $\sigma_1 = \sigma_2$. Relation $\preceq$ extends to strings in $(\Sigma \cup \{\circ\})^*$ under the convention that each string t is implicitly surrounded by wild cards, namely, letter $t[j]$ is $\circ$ when $j \geq |t|$. Hence, v is* more specific *than u (written $u \preceq v$) if $u[j] \preceq v[j]$ for any integer j.*

We can now formally define the occurrences of patterns x in s and their lists.

Definition 3 (occurrence, $\mathcal{L}$). *We say that u* occurs at position ℓ in v *if $u[j] \preceq v[j+\ell]$, for $0 \leq j \leq |u|-1$ (equivalently, we say that u* matches *$v[\ell..\ell+|u|-1]$). For the input string $s \in \Sigma^*$ with $n = |s|$, we consider the* location list *$\mathcal{L}_x \subseteq \{0..n-1\}$ as the set of all the positions on s at which x occurs.*

When a pattern u occurs in another pattern (or into a string) v, we also say that v contains u. For example, the location list of $x = \mathtt{T} \circ \mathtt{G}$ in $s = \mathtt{TTGG}$ is $\mathcal{L}_x = \{0, 1\}$, hence s contains x.

Definition 4 (motif). *Given a parameter $q \geq 2$, called* quorum, *we say that pattern x is a* motif *in s when $|\mathcal{L}_x| \geq q$.*

Given any location list $\mathcal{L}_x$ and any integer d, we adopt the notation $\mathcal{L}_x + d = \{\ell + d \mid \ell \in \mathcal{L}_x\}$ for indicating the occurrences in $\mathcal{L}_x$ "displaced" by the offset d.

Definition 5 (maximality). *A motif x is* maximal *if for any other motif y that contains x, we have no integer d such that $\mathcal{L}_y = \mathcal{L}_x + d$.*

In other words, making a maximal motif x more specific (thus obtaining y) reduces the number of its occurrences in s. Definition 5 is equivalent to that meant in [17] stating that x is maximal if there exist no other motif y and no integer $d \geq 0$ verifying $\mathcal{L}_x = \mathcal{L}_y + d$, such that $x[j] \preceq y[j+d]$ for $0 \leq j \leq |x|-1$ (that is, x occurs in y at position d in our terminology).[2]

Definition 6 (irredundant motif). *A maximal motif x is* irredundant *if, for any maximal motifs $y_1, y_2, \ldots, y_k$ such that $\mathcal{L}_x = \cup_{i=1}^{k} \mathcal{L}_{y_i}$, motif x must be one of the y_is. Conversely, if all the y_is are different from x, pattern x is said to be* covered *by motifs $y_1, y_2, \ldots, y_k$.*

The *basis of irredundant motifs* for string s is the set of *all* irredundant motifs in s. The definition is given with respect to the set of maximal motifs of the input string which is unique; indeed, such basis is unique and it can be used as a generator for all maximal motifs in s as proved in [17]. The size of the basis is the *number* of irredundant motifs contained in it. We illustrate the notions given so far by employing the example string $s = \mathtt{FABCXFADCYZEADCEADC}$. For this string and $q = 2$ the location list of motif $x_1 = \mathtt{A} \circ \mathtt{C}$ is $\mathcal{L}_{x_1} = \{1, 6, 12, 16\}$, and that of motif $x_2 = \mathtt{FA} \circ \mathtt{C}$ is $\mathcal{L}_{x_2} = \{0, 5\}$. They are both maximal because they lose at least one of their occurrences when extended with solid characters at one side (possibly with wild cards in between), or when their wild cards are replaced by solid characters. However, motif $x_3 = \mathtt{DC}$ having list $\mathcal{L}_{x_3} = \{7, 13, 17\}$ is not maximal. It occurs in $x_4 = \mathtt{ADC}$, where $\mathcal{L}_{x_4} = \{6, 12, 16\}$, and its occurrences can be obtained from those of x_4 by a displacement of $d = 1$ positions. The basis of the irredundant motifs for s is made up of $x_1 = \mathtt{A} \circ \mathtt{C}$, $x_2 = \mathtt{FA} \circ \mathtt{C}$, $x_4 = \mathtt{ADC}$, and $x_5 = \mathtt{EADC}$. The location list of each of them cannot be obtained from the union of any of the other location lists.

3 Irredundant Motifs: The Basis and Its Size for Quorum $q = 2$

In this section, we show the existence of an infinite family of strings s_k ($k \geq 5$) for which there are $\Omega(n^2)$ irredundant motifs in the basis for quorum $q = 2$, where $n = |s_k|$. In this way, we disprove the claimed upper bound of $3n$ [17] mentioned in Section 1. Each string s_k will be constructed from a shorter string t_k, which we now define. For each k, $t_k = \mathtt{A}^k\mathtt{TA}^k$, where $\mathtt{A}^k$ denotes the letter A repeated k times (our argument works, in general, for $z^k w z^k$, where z and w are strings of equal length not sharing any common character). String t_k contains an exponential number of maximal motifs, including those having the form $\mathtt{A}\{\mathtt{A}, \circ\}^{k-2}\mathtt{A}$ with exactly two wild cards. To see why, each such motif x occurs four times in t_k: Specifically, two occurrences of x match the first and the last k letters in t_k while each distinct wild card in x matching the letter T in t_k contributes to one of the two remaining occurrences. Extending x or replacing a wild card with a solid character reduces the number of these occurrences, so x is maximal. The idea of our proof is to obtain strings s_k by prefixing t_k with $O(|t_k|)$ symbols so that these motifs x become irredundant in s_k. Since there are $\Omega(k^2)$ of them, and $n = |s_k| = \Theta(|t_k|) = \Theta(k)$, this leads to the claimed result.

In order to define the strings s_k on the alphabet $\Sigma = \{\mathtt{A}, \mathtt{T}, \mathtt{u}, \mathtt{v}, \mathtt{w}, \mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{k-2}\}$, we introduce some notation. Let $\widetilde{u}$ denote the reversal of u, and let ev_k, od_k, u_k, v_k be the strings thus defined

$$\begin{aligned}
\text{if } k \text{ is even}: \quad ev_k &= \mathtt{a}_2\mathtt{a}_4 \cdots \mathtt{a}_{k-2},\\
od_k &= \mathtt{a}_1\mathtt{a}_3 \cdots \mathtt{a}_{k-3},\\
u_k &= ev_k\, \mathtt{u}\, \widetilde{ev_k}\, \mathtt{vw}\, ev_k,\\
v_k &= od_k\, \mathtt{xy}\, \widetilde{od_k}\, \mathtt{z}\, od_k,
\end{aligned}$$

$$\begin{aligned}
\text{if } k \text{ is odd}: \quad ev_k &= \mathtt{a}_2\mathtt{a}_4 \cdots \mathtt{a}_{k-3},\\
od_k &= \mathtt{a}_1\mathtt{a}_3 \cdots \mathtt{a}_{k-2},\\
u_k &= ev_k\, \mathtt{uv}\, \widetilde{ev_k}\, \mathtt{wx}\, ev_k,\\
v_k &= od_k\, \mathtt{y}\, \widetilde{od_k}\, \mathtt{z}\, od_k.
\end{aligned}$$

The strings s_k are then defined by $s_k = u_k v_k t_k$ for $k \geq 5$. Fig. 1 shows them for $k = 7$.

Fact 1. *The length of $u_k v_k$ is $3k$, and that of s_k is $n = 5k + 1$.*

2. Actually, the definition literally reported in [17] is "Definition 4 (Maximal Motif). Let $p_1, p_2, \ldots, p_k$ be the motifs in a sequence s. Let $p_i[j]$ be "." if $j > |p_i|$. A motif p_i is maximal if and only if there exists no p_l, $l \neq i$ and no integer $0 \leq \delta$ such that $\mathcal{L}_{p_i} + \delta = \mathcal{L}_{p_l}$ and $p_l[\delta + j] \preceq p_i[j]$ hold for $1 \leq j \leq |p_i|$." (The symbols in p_i and p_l are indexed starting from 1 onward.) The corresponding example in the paper illustrates the definition for $s = \mathtt{ABCDABCD}$, stating that $p_i = \mathtt{ABCD}$ is maximal while $p_l = \mathtt{ABC}$ is not. However, p_i does not match the definition because of the existence of its prefix p_l (setting $\delta = 0$); hence, we suspect a minor typo in the definition, for which the definition should read as "... such that $\mathcal{L}_{p_i} = \mathcal{L}_{p_l} + \delta$ and $p_i[j] \preceq p_l[\delta + j]$."

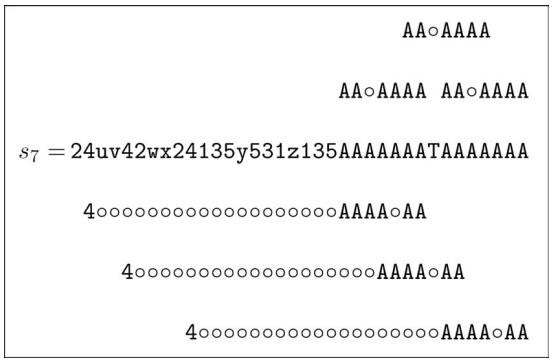

Fig. 1. Example string s_7, ($\mathtt{a}_i$ of the definition is simply denoted by $\mathtt{i}$). Above it, there are the occurrences of w of the Proof of Proposition 1, while the three lines below show the occurrences of motif $x = \mathtt{4}\circ^{19}\mathtt{AAAA}\circ\mathtt{AA}$ in s_7. The letter $\mathtt{4}$ corresponds to position 4 of the wild card in $\mathtt{AAAA}\circ\mathtt{AA}$.

Proof. Whatever the parity of k, the string $u_k v_k$ contains the six letters $\mathtt{u}$, $\mathtt{v}$, $\mathtt{w}$, $\mathtt{x}$, $\mathtt{y}$, $\mathtt{z}$, two occurrences each of ev_k and od_k, and one occurrence each of $\widetilde{ev_k}$ and $\widetilde{od_k}$. Since od_k and ev_k together contain one occurrence of each letter $\mathtt{a}_1$, $\mathtt{a}_2, \ldots, \mathtt{a}_{k-2}$, we have $|od_k| + |ev_k| = k-2$. Moreover, $|\widetilde{ev_k}| = |ev_k|$ and $|\widetilde{od_k}| = |od_k|$, so that $|u_k v_k| = 6 + 3(k-2) = 3k$. This proves the first statement. For the second statement, the total length of s_k follows by observing that $|t_k| = 2k+1$, and so $n = |s_k| = 3k + 2k + 1 = 5k+1$. □

Proposition 1. *For $1 \le p \le k-2$, no motif of the form $\mathtt{A}^p \circ \mathtt{A}^{k-p-1}$ can be maximal in s_k. Also, motif $\mathtt{A}^k$ cannot be maximal in s_k.*

Proof. Let w be an arbitrary motif of the form $\mathtt{A}^p \circ \mathtt{A}^{k-p-1}$, with $1 \le p \le k-2$. Its location list is $\mathcal{L}_w = \{0, k-p, k+1\} + |u_k v_k| = \{3k, 4k-p, 4k+1\}$ since $|u_k v_k| = 3k$ by Fact 1 and w matches the two substrings $\mathtt{A}^k$ of s_k as well as $\mathtt{A}^p\,\mathtt{TA}^{k-p-1}$. The occurrences are shown in Fig. 1 for $k=7$ and $p=2$. No other occurrences are possible. Let us consider the position, say i, of the leftmost appearance of letter $\mathtt{a}_p$ in s_k (recall that there are three positions on s_k at which letter $\mathtt{a}_p$ occurs; we have $i=0$ in our example of Fig. 1 with $p=2$). We claim that motif $y = \mathtt{a}_p \circ^{3k-i-1} w$ satisfies $\mathcal{L}_y = \mathcal{L}_w - (3k-i)$. Since w appears in y, it follows that w cannot be maximal in s_k by Definition 5 (setting $d = -3k+i$). To see why $\mathcal{L}_w = \mathcal{L}_y + (3k-i)$, it suffices to prove that the distance in s_k between the positions of the two leftmost letters $\mathrm{a_p}$ is $k-p$ while that of the leftmost and the rightmost $\mathrm{a_p}$ is $k+1$. The verification is a bit tedious because four cases arise according to the fact that each of k and p can be even or odd. Since the cases are analogous, we detail only two of them, namely, when both k and p are even, and when k is even and p is odd. In the first case, the three occurrences of $\mathtt{a}_p$ are all in u_k. Moreover, the distance between the two leftmost letters $\mathtt{a}_p$ is the length of the substring $\mathtt{a}_p\mathtt{a}_{p+2}\cdots\mathtt{a}_{k-2}\mathtt{u}\mathtt{a}_{k-2}\mathtt{a}_{k-4}\cdots\mathtt{a}_{p+2}$, that is, $2|\mathtt{a}_{p+2}\cdots\mathtt{a}_{k-2}| + 2 = 2(k-2-p)/2 + 2 = k-p$. The distance between the leftmost and rightmost $\mathtt{a}_p$ is the length of $\mathtt{a}_p\mathtt{a}_{p+2}\cdots\mathtt{a}_{k-2}\,\widetilde{ev_k}\,\mathtt{vwa}_2\mathtt{a}_4\cdots\mathtt{a}_{p-2}$. This is also the length of $\mathtt{u}\,\widetilde{ev_k}\,\mathtt{vwa}_2\mathtt{a}_4\cdots\mathtt{a}_{p-2}\mathtt{a}_p\mathtt{a}_{p+2}\cdots\ \mathtt{a}_{k-2} = \mathtt{u}\,\widetilde{ev_k}\,\mathtt{vw}ev_k$, that is, $2(k-2)/2 + 3 = k+1$ as expected. In the second case where k is even and p is odd, the occurrences of $\mathtt{a}_p$ are all in v_k. Analogously to the first case, the distance between the two leftmost letters $\mathtt{a}_p$ is the length of $\mathtt{a}_p\mathtt{a}_{p+2}\cdots\mathtt{a}_{k-3}\mathtt{xya}_{k-3}\cdots\mathtt{a}_{p+2}$, that is, $2|\mathtt{a}_{p+2}\cdots\mathtt{a}_{k-3}| + 3 = 2(k-3-p)/2 + 3 = k-p$. The distance between the leftmost and the rightmost $\mathtt{a}_p$ is the length of the string $\mathtt{a}_p\mathtt{a}_{p+2}\cdots\mathtt{a}_{k-3}\,\mathtt{xy}\widetilde{od_k}\mathtt{z}\mathtt{a}_1\mathtt{a}_3\cdots\mathtt{a}_{p-2}$, which equals $k+1$, the length of $\mathtt{xy}\widetilde{od_k}\mathtt{z}od_k$. The analogous verification of the other two cases yields the fact that w cannot be maximal.

The second part of the lemma for motif $\mathtt{A}^k$ proceeds along the same lines, except that we choose $y = \mathtt{a}_p \circ^{3k-i-1} \mathtt{A}^k$ with i as before (note that y is not required to be maximal and that the motifs in the statement are maximal in t_k). □

Proposition 2. *Each motif of the form $\mathtt{A}\{\mathtt{A}, \circ\}^{k-2}\mathtt{A}$ with exactly two $\circ$s is irredundant in s_k.*

Proof. Let x be an arbitrary motif of the form $\mathtt{A}\{\mathtt{A}, \circ\}^{k-2}\mathtt{A}$ with two $\circ$s, namely, $x = \mathtt{A}^{p_1} \circ \mathtt{A}^{p_2-p_1-1} \circ \mathtt{A}^{k-p_2-1}$ for $1 \le p_1 < p_2 \le k-2$. To prove that x is an irredundant motif, we first show that x is maximal. Its location list is $\mathcal{L}_x = \{0, k-p_2, k-p_1, k+1\} + 3k$ since $|u_k v_k| = 3k$ by Fact 1 and x matches the two substrings $\mathtt{A}^k$ of s_k as well as $\mathtt{A}^{p_1}\,\mathtt{TA}^{k-p_1-1}$ and $\mathtt{A}^{p_2}\,\mathtt{TA}^{k-p_2-1}$. Any other motif y such that x occurs in y can be obtained by replacing at least one wild card (at position p_1 or p_2) in x with a solid character, but this would cause the removal of position $4k-p_1$ or $4k-p_2$ from $\mathcal{L}_x$. Analogously, extending x to the right by putting a solid character at position $|x|$ or larger would eliminate position $4k+1$ from $\mathcal{L}_x$. Finally, extending x to the left by a solid character would eliminate at least one position from $\mathcal{L}_x$ because no symbol occurs four times in $u_k v_k$. In conclusion, for any motif y such that x occurs in y, we have $\mathcal{L}_y \neq \mathcal{L}_x + d$ for any integer d and, thus, x is a maximal motif by Definition 5. We now prove that x is irredundant according to Definition 6. Let us consider an arbitrary set of maximal motifs $y_1, y_2, \ldots, y_h$ such that $\mathcal{L}_x = \cup_{i=1}^{h} \mathcal{L}_{y_i}$. We claim that at least one y_i is of the form $\mathtt{A}\{\mathtt{A}, \circ\}^{k-2}\mathtt{A}$. Indeed, there must exist a location list $\mathcal{L}_{y_i}$ containing position $4k+1$ since that position belongs to $\mathcal{L}_x$. This implies that y_i occurs in the suffix $\mathtt{A}^k$ of s_k. It cannot be that $|y_i| < k$ since y_i would occur also in some position $j > 4k+1$ whereas $j \notin \mathcal{L}_x$, so it is impossible. Consequently, y_i is of length k and matches $\mathtt{A}^k$, thus being of the form $\mathtt{A}\{\mathtt{A}, \circ\}^{k-2}\mathtt{A}$. We observe that y_i cannot contain zero or one $\circ$s, as it would not be maximal by Proposition 1. Also, y_i cannot contain three or more $\circ$s, as each distinct $\circ$ symbol would match the letter $\mathtt{T}$ in s_k giving $|\mathcal{L}_{y_i}| > |\mathcal{L}_x|$, which is impossible. The only possibility is that y_i contains exactly two $\circ$s as x does at the same positions because $\mathcal{L}_y \subseteq \mathcal{L}_x$ and they are maximal. It follows that $y_i = x$ proving the proposition. □

Theorem 2. *The basis for string s_k contains $\Omega(n^2)$ irredundant motifs, where $n = |s_k|$ and $k \ge 5$.*

Proof. By Proposition 2, the number of irredundant motifs in s_k is at least $\binom{k-2}{2} = \Omega(k^2)$, the number of choices of two positions in $\{\mathtt{A}, \circ\}^{k-2}$. Since $|s_k| = 5k+1$ by Fact 1, we get the conclusion. □

4 Tiling Motifs: The Basis and Its Properties

4.1 Terminology and Properties

In this section, we introduce a natural notion of a basis for generating all maximal motifs occurring in a string s of length n.

Definition 7 (tiling motif). *A maximal motif x is* tiling *if, for any maximal motifs $y_1, y_2, \ldots, y_k$ and for any integers $d_1, d_2, \ldots, d_k$ such that $\mathcal{L}_x = \cup_{i=1}^{k}(\mathcal{L}_{y_i} + d_i)$, motif x must be one of the y_is. Conversely, if all the y_is are different from x, pattern x is said to be tiled by motifs $y_1, y_2, \ldots, y_k$.*

The notion of tiling is in general more selective than that of irredundancy. Continuing our example string $s = \texttt{FABCXFADCYZEADCEADC}$, we have seen in Section 2 that motif $x_1 = \texttt{A} \circ \texttt{C}$ is irredundant for s. Now, x_1 is tiled by $x_2 = \texttt{FA} \circ \texttt{C}$ and $x_4 = \texttt{ADC}$ according to Definition 7 since its location list, $\mathcal{L}_{x_1} = \{1, 6, 12, 16\}$, can be obtained from the union of $\mathcal{L}_{x_2} = \{0, 5\}$ and $\mathcal{L}_{x_4} = \{6, 12, 16\}$ with respective displacements $d_2 = 1$ and $d_4 = 0$.

Remark 1. A fairly direct consequence of Definition 7 is that if x is tiled by $y_1, y_2, \ldots, y_k$ with associated displacements $d_1, d_2, \ldots, d_k$, then x occurs at position d_i in y_i for $1 \le i \le k$. As a consequence, we have that $d_i \ge 0$ in Definition 7. Note also that the y_is in Definition 7 are not necessarily distinct and that $k > 1$ for tiled motifs. (It follows from the fact that $\mathcal{L}_x = \mathcal{L}_{y_1} + d_1$ with $x \neq y_1$ would contradict the maximality of both x and y_1.) As a result, a maximal motif x occurring exactly q times in s is tiling as it cannot be tiled by any other motifs because such motifs would occur less than q times.

The *basis of tiling motifs* is the complete set of all tiling motifs for s, and the size of the basis is the number of these motifs. For example, the basis, let us denote it by $\mathcal{B}$, for $\texttt{FABCXFADCYZEADCEADC}$ contains $\texttt{FA} \circ \texttt{C}$, $\texttt{EADC}$, and $\texttt{ADC}$ as tiling motifs. Although Definition 7 is derived from that of irredundant motifs given in Definition 6, the difference is much more substantial than it may appear. The basis of tiling motifs relies on the fact that tiling motifs are considered as invariant by displacement as for maximality. Consequently, our definition of basis is symmetric, that is, each tiling motif in the basis for the reverse string $\tilde{s}$ is the reverse of a tiling motif in the basis of s. This follows from the symmetry in Definition 7 and from the fact that maximality is also symmetric in Definition 5. It is a *sine qua non* condition for having a notion of basis invariant by the left-to-right or right-to-left order of the symbols in s (like the entropy of s), while this property does not hold for the irredundant motifs.

The basis of tiling motifs has further interesting properties for quorum $q = 2$, illustrated in Sections 4.2, 4.3, and 4.4. In Section 4.2, we show that our basis is linear (that is, its size is at most $n - 1$). In Section 4.3, we show that the total size of the location lists for the tiling motifs is less than $2n$, describing how to find them in $O(n^2 \log n \log |\Sigma|)$ time. In Section 4.4, we discuss some applications such as generating all maximal motifs with the basis and finding motifs with a constraint on the number of undefined symbols.

4.2 A Linear Upper Bound for the Tiling Motifs with Quorum $q = 2$

Given a string s of length n, let $\mathcal{B}$ denote its basis of tiling motifs for quorum $q = 2$. Although the number of maximal motifs may be exponential and the basis of irredundant motifs may be at least quadratic (see Section 3), we show that the size of $\mathcal{B}$ is always less than n. For this, we introduce an operator $\oplus$ between the symbols of Σ to define the *merges*, which are at the heart of the properties of $\mathcal{B}$. Given two letters $\sigma_1, \sigma_2 \in \Sigma$ with $\sigma_1 \neq \sigma_2$, the operator satisfies $\sigma_1 \oplus \sigma_2 = \circ$ and $\sigma_1 \oplus \sigma_1 = \sigma_1$. The operator applies to any pair of strings $x, y \in \Sigma^*$, so that $u = x \oplus y$ satisfies $u[j] = x[j] \oplus y[j]$ for all integers j.

Definition 8 (Merge). *For $1 \le k \le n - 1$, let s_k be the (infinite) string whose character at position i is $s_k[i] = s[i] \oplus s[i + k]$. If s_k contains at least one solid character, $Merge_k$ denotes the motif obtained by removing all the leading and trailing $\circ$s in s_k (that is, those appearing before the leftmost solid character and after the rightmost solid character).*

For example, $\texttt{FABCXFADCYZEADCEADC}$ has $Merge_4 = \texttt{EADC}$, $Merge_5 = \texttt{FA} \circ \texttt{C}$, $Merge_6 = Merge_{10} = \texttt{ADC}$, and $Merge_{11} = Merge_{15} = \texttt{A} \circ \texttt{C}$. The latter is the only merge that is not a tiling motif.

Lemma 1. *If $Merge_k$ exists, it must be a maximal motif.*

Proof. Motif $x = Merge_k$ occurs at positions, say, i and $i + k$ in s. Character $s_k[i]$ is solid by Definitions 4 and 8. We use the fact that x at occurs at least twice in s for showing that it is maximal. Suppose it is not maximal. By Definition 5, there exists $y \neq x$ such that x occurs in y and $\mathcal{L}_y = \mathcal{L}_x + d$ for some integer d (in this case $d \le 0$). Since y is more specific than x displaced by d, there must exist at least one position j with $0 \le j < |y|$ such that $x[j + d] = \circ$ and $y[j] = \sigma \in \Sigma$. Hence, $x[j + d] = s[i + (j + d)] \oplus s[i + k + (j + d)] = \circ$, and so $s[(i + d) + j] \neq s[(i + k + d) + j]$. Since $y[j]$ cannot match both of the latter symbols in s, at least one of $i + d$ or $i + k + d$ is not a position of y in s. This contradicts the hypothesis that $\mathcal{L}_y = \mathcal{L}_x + d$, whereas both $i, i + k \in \mathcal{L}_x$. □

Lemma 2. *For each tiling motif x in the basis $\mathcal{B}$, there is at least one k for which $Merge_k = x$.*

Proof. As mentioned in Remark 1, a maximal motif occurring exactly twice in s is tiling. Hence, if $|\mathcal{L}_x| = 2$, say $\mathcal{L}_x = \{i, j\}$ with $j > i$, then $x = Merge_k$ with $k = j - i$ by the maximality of x and that of the merges by Lemma 1. Let us now consider the case where $|\mathcal{L}_x| > 2$. For any pair $i, j \in \mathcal{L}_x$, we denote by u_{ij} the string $s[i..i + |x| - 1] \oplus s[j..j + |x| - 1]$ obtained by applying the operator $\oplus$ to the two substrings of s matching x at positions i and j, respectively. We have $x \preceq u_{ij}$ since x occurs at positions i and j, and $\mathcal{L}_x = \bigcup_{i,j \in \mathcal{L}_x} \mathcal{L}_{u_{ij}}$ since we are taking all pairs of occurrences of x. Letting $k = |j - i|$ for $i, j \in \mathcal{L}_x$, we observe that u_{ij} is a substring of $Merge_k$ occurring at position, say, δ_k in it. Thus,

$$\bigcup_{i,j \in \mathcal{L}_x} \mathcal{L}_{u_{ij}} = \bigcup_{k=|j-i| \,:\, i,j \in \mathcal{L}_x} \left(\mathcal{L}_{Merge_k} + \delta_k\right) = \mathcal{L}_x.$$

By Definition 7, the fact that x is tiling implies that x must be one $Merge_k$, proving the lemma. □

We now state the main property of tiling bases that follows directly from Lemma 2.

Theorem 3 (linearity of the basis). *Given a string s of length n and the quorum $q = 2$, let $\mathcal{M}$ be the set of $Merge_k$, for $1 \leq k \leq n-1$ such that $Merge_k$ exists. The basis $\mathcal{B}$ of tiling motifs for s satisfies $\mathcal{B} \subseteq \mathcal{M}$ and, therefore, the size of $\mathcal{B}$ is at most $n-1$.*

A simple consequence of Theorem 3 implies a tight bound on the number of tiling motifs for periodic strings. If $s = w^e$ for a string w repeated $e > 1$ times, then s has at most $|w|$ tiling motifs.

Corollary 1. *The number of tiling motifs for s is at most p, the smallest period of s.*

The bound in Corollary 1 is not valid for irredundant motifs. String $s = \texttt{ATATATATA}$ has period $p = 2$ and only one tiling motif ATATATA, while its irredundant motifs are A, ATA, ATATA, and ATATATA.

4.3 A Simple Algorithm for Computing Tiling Motifs with Quorum $q = 2$

We describe how to compute the basis $\mathcal{B}$ for string s when $q = 2$. A brute-force algorithm generating first all maximal motifs of s takes exponential time in the worst case. Theorem 3 plays a crucial role in that we first compute the motifs in $\mathcal{M}$ and then discard those being tiled. Since $\mathcal{B} \subseteq \mathcal{M}$, what remains is exactly $\mathcal{B}$. To appreciate this approach, it is worth noting that we are left with the problem of selecting $\mathcal{B}$ from $n-1$ maximal motifs in $\mathcal{M}$ at most, rather than selecting $\mathcal{B}$ among *all* the maximal motifs in s, which may be exponential in number. Our simple algorithm takes $O(n^2 \log n \log |\Sigma|)$ time and is faster than previous (and more complicated) methods discussed in Section 1.

Step 1. Compute the multiset $\mathcal{M}'$ of merges. Letting $s_k[i]$ be the leftmost solid character of string s_k in Definition 8, we define $occ_x = \{i, i+k\}$ to be the positions of the two occurrences of x whose superposition generates $x = Merge_k$. For $k = 1, 2, \ldots, n-1$, we compute string s_k in $O(n-k)$ time. If s_k contains some solid characters, we compute $x = Merge_k$ and occ_x in the same time complexity. As a result, we compute the multiset $\mathcal{M}'$ of merges in $O(n^2)$ time. Each merge x in $\mathcal{M}'$ is identified by a triplet $\langle i, i+k, |x| \rangle$, from which we can recover the jth symbol of x in constant time by simple arithmetic operations and comparisons.

Step 2. Transform the multiset $\mathcal{M}'$ into the set $\mathcal{M}$ of merges. Since there can be two or more merges in $\mathcal{M}'$ that are identical and correspond to the same merge in $\mathcal{M}$, we put together all identical merges in $\mathcal{M}'$ by radix sorting them. The total cost of this step is dominated by radix sorting, giving $O(n^2)$ time. As a byproduct, we produce the temporary location list $T_x = \bigcup_{x'=x\,:\,x' \in \mathcal{M}'} occ_{x'}$ for each distinct $x \in \mathcal{M}$ thus obtained.

Lemma 3. *Each motif $x \in \mathcal{B}$ satisfies $T_x = \mathcal{L}_x$.*

Proof. For a fixed $x \in \mathcal{B}$, the fact that x is equal to at least one merge by Lemma 2 implies that T_x is well defined, with $|T_x| \geq 2$. Since $T_x \subseteq \mathcal{L}_x$, let us assume by contradiction that $\mathcal{L}_x - T_x \neq \emptyset$. For each pair $i \in \mathcal{L}_x - T_x$ and $j \in T_x$, let $m_{ij} = Merge_{|j-i|}$, which is maximal by Lemma 1. Note that each $m_{ij} \neq x$ by our assumption as otherwise i would belong to T_x; however, x must occur in m_{ij}, say, at position δ_{ij} in m_{ij}. Consequently, $\bigcup_{i \in \mathcal{L}_x - T_x, j \in T_x} (\mathcal{L}_{m_{ij}} + \delta_{ij}) = \mathcal{L}_x$ since any occurrence of x is either $i \in \mathcal{L}_x - T_x$ or $j \in T_x$. At this point, we apply Definition 7 to the tiling motif x, obtaining the contradiction that x must be equal to one m_{ij}. □

Notice that the conclusion of Lemma 3 does not necessarily hold for the motifs in $\mathcal{M} - \mathcal{B}$. For the previous example string FADABCXFADCYZEADCEADCFADC, one such motif is $x = \texttt{ADC}$ with $\mathcal{L}_x = \{8, 14, 18, 22\}$ while $T_x = \{8, 18\}$.

Step 3. Select $\mathcal{M}^* \subseteq \mathcal{M}$, where $\mathcal{M}^* = \{x \in \mathcal{M} : T_x = \mathcal{L}_x\}$. In order to build $\mathcal{M}^*$, we employ the Fischer-Paterson algorithm based on convolution [8] for string matching with don't cares to compute the whole list of occurrences $\mathcal{L}_x$ for each merge $x \in \mathcal{M}$. Its cost is $O((|x| + n) \log n \log |\Sigma|)$ time for each merge x. Since $|x| < n$ and there are at most $n-1$ motifs $x \in \mathcal{M}$, we obtain $O(n^2 \log n \log |\Sigma|)$ time to construct all lists $\mathcal{L}_x$. We can compute $\mathcal{M}^*$ by discarding the merges $x \in \mathcal{M}$ such that $T_x \neq \mathcal{L}_x$ in additional $O(n^2)$ time.

Lemma 4. *The set $\mathcal{M}^*$ satisfies the conditions $\mathcal{B} \subseteq \mathcal{M}^*$ and $\sum_{x \in \mathcal{M}^*} |\mathcal{L}_x| < 2n$.*

Proof. The first condition follows from the fact that the motifs in $\mathcal{M} - \mathcal{M}^*$ are surely tiled by Lemma 3. The second condition follows from the definition of $\mathcal{M}^*$ and from the observation that

$$\sum_{x \in \mathcal{M}^*} |\mathcal{L}_x| = \sum_{x \in \mathcal{M}^*} |T_x| \leq \sum_{x \in \mathcal{M}} |occ_x| < 2n,$$

since $|occ_x| = 2$ (see Step 1) and there are less than n of them. □

The property of $\mathcal{M}^*$ in Lemma 4 is crucial in that $\sum_{x \in \mathcal{M}} |\mathcal{L}_x| = \Theta(n^2)$ when many lists contain $\Theta(n)$ entries. For example, $s = \texttt{A}^n$ has $n-1$ distinct merges, each of the form $x = \texttt{A}^i$ for $1 \leq i \leq n-1$, and so $|\mathcal{L}_x| = n - i + 1$. This would be a sharp drawback in Step 4 when removing tiled motifs as it may turn into a $\Theta(n^3)$ algorithm. Using $\mathcal{M}^*$ instead, we are guaranteed that $\sum_{x \in \mathcal{M}^*} |\mathcal{L}_x| = O(n)$; hence, we may still have some tiled motifs in $\mathcal{M}^*$, but their total number of occurrences is $O(n)$.

Step 4. Discard the tiled motifs in $\mathcal{M}^*$. We can now check for tiling motifs in $O(n^2)$ time. Given two distinct motifs $x, y \in \mathcal{M}^*$, we want to test whether $\mathcal{L}_x + d \subseteq \mathcal{L}_y$ for some integer d and, in that case, we want to mark the entries in $\mathcal{L}_y$ that are also in $\mathcal{L}_x + d$. At the end of this task, the lists having *all* entries marked are tiled (see Definition 7). By removing their corresponding motifs from $\mathcal{M}^*$, we eventually obtain the basis $\mathcal{B}$ by Lemma 4. Since the meaningful values of d are as many as the entries of $\mathcal{L}_y$, we have only $|\mathcal{L}_y|$ possible values to check. For a *given* value of d, we avoid to merge $\mathcal{L}_x$ and $\mathcal{L}_y$ in $O(|\mathcal{L}_x| + |\mathcal{L}_y|)$ time to perform the test, as it would contribute to a total of $\Theta(n^3)$ time. Instead, we exploit the fact that each list has values ranging from 1 to n, and use two bit-vectors of size n to perform the above check in $O(|\mathcal{L}_x| \times |\mathcal{L}_y|)$ time for *all* values of d. This gives $O(\sum_y \sum_x |\mathcal{L}_x| \times |\mathcal{L}_y|) = O(\sum_y |\mathcal{L}_y| \times \sum_x |\mathcal{L}_x|) = O(n^2)$ by Lemma 4.

We therefore detail how to perform the above check with $\mathcal{L}_x$ and $\mathcal{L}_y$ in $O(|\mathcal{L}_x| \times |\mathcal{L}_y|)$ time. We use two bit-vectors V_1 and V_2 of length n initially set to all zeros. Given $y \in \mathcal{M}^*$, we set $V_1[i] = 1$ if $i \in \mathcal{L}_y$. For each $x \in \mathcal{M}^* - \{y\}$ and for each $d \in (\mathcal{L}_y - m)$ (where m is the smallest entry of $\mathcal{L}_x$), we then perform the following test. If *all* $j \in \mathcal{L}_x + d$ satisfy $V_1[j] = 1$, we set $V_2[j] = 1$ for all such j. Otherwise, we take the next value of d, or the next motif if there are no more values of d, and we repeat the test. After examining all $x \in \mathcal{M}^* - \{y\}$, we check whether $V_1[i] = V_2[i]$ for all $i \in \mathcal{L}_y$. If so, y is tiled as its list is covered by possibly shifted location lists of other motifs. We then reset the ones in both vectors in $O(|\mathcal{L}_y|)$ time.

Summing up Steps 1-4, we have that the dominant cost is that of Step 3 and that we have proved the following result.

Theorem 4. *Given an input string s of length n over the alphabet Σ, the basis of tiling motifs with quorum $q = 2$ can be computed in $O(n^2 \log n \log |\Sigma|)$ time. The total number of motifs in the basis is less than n, and the total number of their occurrences in s is less than $2n$.*

We have implemented the algorithm underlying Theorem 4, and we report here the lessons learned from our experiments. Step 1 requires, in practice, less than the predicted $O(n^2)$ running time. If $p = 1/|\Sigma|$ denotes the probability that two randomly chosen symbols of Σ match in the uniform distribution, the probability of finding the first solid character in a merge follows the binomial distribution, and so the expected number of examined characters in s is $O(1/p) = O(|\Sigma|)$, yielding $O(n|\Sigma|)$ time on the average to locate the *first* (scanning s from the beginning) and the *last* (scanning s from the end backward) solid character in each merge. A similar approach can be followed in Step 2 for finding the distinct merges. In this case, the merges are first partially sorted using hashing and exploiting the fact that the input is *almost* sorted. Insertion sort is then the best choice and works very efficiently in our experiments (at least 50 percent faster than Quicksort). We do not compute yet the *full* merges at this stage, but we delay this expensive part to a later stage on a small set of buckets that require explicit representation of the merges. As a result, the average case is almost linear. For example, executing Steps 1 and 2 on chromosome V of *C.elegans* containing more than 21 million bases took around 15 minutes on a machine with 512Mb of RAM running Linux on a 1Ghz AMD Athlon processor. Step 3 is expensive also in practice and the worst case predicted by theory shows up in the experiments. Running this step on sequences much shorter than chromosome V of *C.elegans* took many hours. Step 4 is not much of a problem. As a result, an alternative way of selecting $\mathcal{M}^*$ from $\mathcal{M}$ in Step 3 working fast in practice, would improve considerably the overall performance.

4.4 Some Applications

Checking whether a pattern is a motif. The main property underlying the notion of basis is that it is a generator of all motifs. The generation can be done as follows: First select segments of motifs in the basis that start and end with solid characters, then replace any number of internal solid characters by wild cards. However, since the number of motifs, and even maximal motifs, can be exponential, this is not really meaningful unless this number is small and the time complexity of the algorithm is proportional to the total size of the output. An attempt in this direction is done in [18]. The dual problem concerns testing only one pattern. We show how, given a pattern x, it can be tested whether x is a motif for string s, that is, if pattern x occurs at least q times in s. There are two possible ways of performing such a test, depending on whether we test directly on the string or on the basis. The answer relies on iterative applications of the observation made in Remark 1, according to which any tiled motif must occur in at least one tiling motif. The next two statements deal with the alternative. In both cases, we assume that integer k comes from the decomposition of pattern x in the form $u_0 \circ^{\ell_0} u_1 \circ^{\ell_1} \cdots u_{k-1} \circ^{\ell_{k-1}} u_k$, where the subwords u_i contain no wild cards ($u_i \in \Sigma^*$, $0 \le i \le k$) and ℓ_j are positive integers, $0 \le j \le k-1$. The next proposition states a well-known fact on matching such a pattern in a text without any wild card that we report here because it is used in the sequel.

Proposition 3. *The positions of the occurrences of a pattern x in a string of length n can be computed in time $O(kn)$.*

Proof. This is a mere application of matching a pattern with do not cares inside a text without do not cares. Using, for instance, the Fischer and Paterson's algorithm [8] is not necessary. Instead, the positions of the subwords u_i are computed by a multiple string-matching algorithm, such as the Aho-Corasick algorithm [1]. For each position p, a counter associated with position $p - \ell$ on s is incremented, where ℓ is the position of u_i in x (ℓ is the offset of u_i in x). Counters whose value is $k+1$ correspond then to occurrences of x in s. It remains to check if x occurs at least q times in s. The running time is governed by the string-matching algorithm, which is $O(kn)$ (equivalent to running k times a linear-time string matching algorithm).□

Proposition 4. *Given the basis $\mathcal{B}$ of string s, testing if pattern x is a motif or a maximal motif can be done in $O(kb)$ time, where $b = \sum_{y \in \mathcal{B}} |y|$.*

Proof. From Remark 1, testing if x is a maximal motif requires only finding if x occurs in an element y of the basis. To do this, we can apply the procedure of the previous proof because wild cards in y should be viewed as extra characters that do not match any letter of Σ. The time complexity of the procedure is thus $O(kb)$. Since a nonmaximal motif occurs in a maximal motif, the same procedure applies to test if x is a general motif. □

As a consequence of Propositions 3 and 4, we get an upper bound on the time complexity for testing motifs.

Corollary 2. *Testing whether or not pattern $u_0 \circ^{\ell_0} u_1 \circ^{\ell_1} \cdots u_{k-1} \circ^{\ell_{k-1}} u_k$ is a motif in a string of length n having a basis of total size b can be done in time $O(k \cdot \min\{b, n\})$.*

Remark 2. Inside the procedure described in the proofs of Propositions 3 and 4, it is also possible to use bit-vector pattern matching methods [3], [16], [25] to compute the occurrences of x. This leads to practically efficient solutions running in time proportional to the length of

the string n or the total size of the basis b, in the bit-vector model of machine. This is certainly a method of choice for short patterns.

Finding the longest motif with bounded number of wild cards. We address an interesting question concerning the computation of a longest motif occurring repeated in a string. Given an integer $g \geq 0$, let $LM_g(s)$ be the maximal length of motifs occurring in a string s of length n with quorum $q = 2$, and containing no more than g wild cards. If $g = 0$, the value can be computed in $O(n \log |\Sigma|)$ time with the help of the suffix tree of s (see [5] or [10]). For $g > 0$, we can show that $LM_g(s)$ can be computed in $O(gn^2)$ time using the suffix tree augmented (in linear time) to accept longest common ancestor (LCA) queries as follows: For each possible pair (i, j) of positions on s for which $s[i] = s[j]$, we compute the longest common prefix of $s[i..n-1]$ and $s[j..n-1]$ in constant time through an LCA query on the suffix tree. If ℓ is the length of the prefix, we get the first part $s[i..i+\ell-1] \circ$ of a possible longest motif. The second part is found similarly by considering the pair of positions $(i+\ell+1, j+\ell+1)$. The process is iterated g times (or less) and provides a longest motif containing at most g wild cards and occurring at positions i and j. Length $LM_g(s)$ is obtained by taking the maximum length of motifs for all pairs of positions (i, j). This yields the next result.

Proposition 5. *Using the suffix tree, $LM_g(s)$ can be computed in $O(gn^2)$ time.*

What makes the use of the basis of tiling motifs interesting is that computing $LM_g(s)$ becomes a mere pattern matching exercise because of the strong properties of the basis. This contrasts with the previous result grounded on the deep algorithmic technique for LCA queries.

Proposition 6. *Using the basis $\mathcal{B}$ of tiling motifs, $LM_g(s)$ can be computed in time $O(b)$, where $b = \sum_{y \in \mathcal{B}} |y|$.*

Proof. Let x be a motif yielding $LM_g(s)$ (i.e., x is of length $LM_g(s)$); hence, x occurs at least twice in s. Let y be a maximal motif in which x occurs (we have $y = x$ if x is itself maximal). Let z be a tiling motif in which y occurs (again we may have $z = y$ if y is a tiling motif). The word x then occurs in z that belongs to the basis. Let us say that it matches $z[i..j]$. Assume that x is not a tiling motif, that is $x \neq z$. Certainly, $i = 0$ or $z[i-1] = \circ$, otherwise, x would not be the longest with its property. For the same reason, $j = |z| - 1$ or $z[j+1] = \circ$. But, indeed, x occurs exactly in z, which means that the wild card symbols do not match any solid symbol. Because, otherwise, $z[i..j]$ would contain less than g do not cares and could be extended by at least one symbol to the left or to the right because $x \neq z$, yielding a contradiction with the definition of x. Therefore, either x is a tiling motif or it matches exactly a segment of one of the tiling motifs. Searching for x thus reduces to finding a longest segment of a tiling motif in $\mathcal{B}$ that contains no more than g wild cards. The computation can be done in linear time with only two pointers on s, which proves the result. □

By Proposition 6, it is clear that a small basis $\mathcal{B}$ leads to an efficient computation once $\mathcal{B}$ is given. If we have to build $\mathcal{B}$ from scratch, we can observe that no (maximal) motif can give a larger value of $LM_g(s)$ if it does not belong to $\mathcal{B}$. With this observation, we have $O(n^2)$ running time, which always beats the $O(g \times n^2)$ cost of using the suffix tree. In particular, it is interesting to notice that the running time of the algorithm using the basis is independent of the parameter g.

5 Pseudopolynomial Bases for Higher Quorum

We now discuss the general case of quorum $q \geq 2$ for finding the basis of a string of length n. Differently from previous work, we show in Section 5.1 that *no* polynomial-time algorithm can exist for *any* arbitrary value of q in the worst case, both for the basis of irredundant motifs and for the basis of tiling motifs. The size of these bases provably depends *exponentially* on suitable values of $q \geq 2$, that is, we give a lower bound of $\binom{\frac{n-1}{2}-1}{q-1} = \Omega\left(\frac{1}{2^q}\binom{n-1}{q-1}\right)$. In practice, this size has an exponential growth for increasing values of q up to $O(\log n)$, but larger values of q are theoretically possible in the worst case. Fixing $q = (n-1)/4 + 1$ in our lower bound, we get a size of $\Omega(2^{(n-1)/4})$ motifs in the bases. On the average, $q = O(\log_{|\Sigma|} n)$ by extending the argument after Theorem 4, namely, using the fact that on the average the number of simultaneous comparisons to find the first solid character of a merge is $O(|\Sigma|^{q-1})$, which must be less than n.

We show a further property for the basis of tiling motifs in Section 5.2, giving an upper bound of $\binom{n-1}{q-1}$ on its size with a simple proof. Since we can find an algorithm taking time proportional to the square of that size, we can conclude that a worst-case polynomial-time algorithm for finding the basis of tiling motifs exists if and only if the quorum q satisfies either $q = O(1)$ or $q = n - O(1)$ (the latter condition is hardly meaningful in practice).

5.1 A Lower Bound of $\binom{\frac{n-1}{2}-1}{q-1}$ on the Bases

We show the existence of a family of strings for which there are at least $\binom{\frac{n-1}{2}-1}{q-1}$ tiling motifs for a quorum q. Since a tiling motif is also irredundant, this gives a lower bound for the irredundant motifs to be combined with that in Section 3 (note that the lower bound in Section 3 still gives $\Omega(n^2)$ for $q \geq 2$). For $q > 2$, this gives a lower bound of $\Omega\binom{\frac{n-1}{2}-1}{q-1} = \Omega\left(\frac{1}{2^q}\binom{n-1}{q-1}\right)$ for the number of both tiling and irredundant motifs.

The strings are this time of the form $t_k = \mathtt{A}^k\mathtt{T}\mathtt{A}^k$ ($k \geq 5$), without the left extension used in the bound of Section 3. The proof proceeds by exhibiting $\binom{k-1}{q-1}$ motifs that are maximal and have each exactly q occurrences, from when it follows immediately that they are tiling. Indeed, Remark 1 for tiling motifs holds for any $q \geq 2$. Namely, all maximal motifs that occur exactly q times in a string are tiling.

Proposition 7. *For $2 \leq q \leq k$ and $1 \leq p \leq k - q + 1$, any motif $\mathtt{A}^p \circ \{\mathtt{A}, \circ\}^{k-p-1} \circ \mathtt{A}^p$ with exactly q wild cards is tiling (and so irredundant) in t_k.*

Proof. Let x be an arbitrary motif $\mathtt{A}^p \circ \{\mathtt{A}, \circ\}^{k-p-1} \circ \mathtt{A}^p$ with $1 \le p \le k-q+1$ and q wild cards; namely, $x = \mathtt{A}^{p_1} \circ \mathtt{A}^{p_2-p_1-1} \circ \cdots \circ \mathtt{A}^{p_{q-1}-p_{q-2}-1} \circ \mathtt{A}^{k-p_{q-1}-1} \circ \mathtt{A}^{p_1}$ for $1 \le p_1 < p_2 < \cdots < p_{q-1} \le k-1$ and $p = p_1$. We first have to prove that x is a maximal motif according to Definition 5. Its length is $k+1+p_1$ and its location list is $\mathcal{L}_x = \{0,\ k-p_{q-1}, \ldots, k-p_2, k-p_1\}$. Observe that the number of its occurrences is exactly the number of times the wild card appears in x, which is equal to q. A motif y different from x such that x occurs in y can be obtained by replacing the wild card at position p_i with a solid symbol, for $1 \le i \le q-1$, but this eliminates $k-p_i$ from the location list of y. Also, y can be obtained by extending x to the right by a solid symbol (at any position $\ge |x|$), but then position $k-p_1$ is not in $\mathcal{L}_y$ because the last symbol in that occurrence of y occupies position $(k-p_1)+|y|-1 \ge (k-p_1)+|x| = (k-p_1)+(k+1+p_1) > |t_k|-1$ in t_k, which is impossible. Analogously, y can be obtained by extending x to the left by a solid symbol (at any position $d<0$), but position 0 is no longer in $\mathcal{L}_y$. Consequently, for any motif y more specific than x, we have $\mathcal{L}_y \neq \mathcal{L}_x + d$, implying that x is maximal. As previously mentioned, x is tiling because it has exactly q occurrences. □

Theorem 5. *String t_k has $\binom{\frac{n-1}{2}-1}{q-1} = \Omega\left(\frac{1}{2^q}\binom{n-1}{q-1}\right)$ tiling (and irredundant) motifs, where $n = |t_k|$ and $k \ge 2$.*

Proof. By Proposition 7, the tiling or irredundant motifs in t_k are at least $\binom{k-1}{q-1}$, the number of choices of $q-1$ positions on $\mathtt{A}^{k-1}$. Since $n = 2k+1$, we obtain the statement. □

5.2 An Upper Bound of $\binom{n-1}{q-1}$ Tiling Motifs

We now prove that $\binom{n-1}{q-1}$ is an upper bound for the size of a basis of tiling motifs for a string s and quorum $q \ge 2$. Let us denote as before such a basis by $\mathcal{B}$. To prove the upper bound, we use again the notion of a merge, except that it now involves q strings. The operator $\oplus$ between the elements of Σ extends to more than two arguments, so that the result is a $\circ$ if at least two arguments differ. Let k denote now an *array* of $q-1$ positive values $k_1, \ldots, k_{q-1}$ with $1 \le k_i < k_j \le n-1$ for all $1 \le i < j \le q-1$.

Definition 9. *Let s_k denote the string such that its jth character is $s_k[j] = s[j] \oplus s[j+k_1] \oplus \cdots \oplus s[j+k_{q-1}]$ for all integers j. $Merge_k$ is the pattern obtained by removing all the leading and trailing $\circ$s in s_k (that is, appearing before the leftmost solid character and after the rightmost solid character).*

Lemmas 5 and 6 reported below extend Lemmas 1 and 2 for $q > 2$.

Lemma 5. *If $Merge_k$ exists for quorum q, then it must be a maximal motif.*

Proof. Let $x = Merge_k$ denote the (nonempty) pattern, and let $s_k[i]$ be its first character, which is solid by Definition 9. Since x occurs at least q times in s, at positions $i, i+k_1, \ldots, i+k_{q-1}$, then x is a motif for quorum q. We show that x is maximal. Suppose it is not maximal. By Definition 5, there exists $y \neq x$ s.t. x occurs in y and $\mathcal{L}_y = \mathcal{L}_x + d$ for some integer d. This implies there exists at least one position j with $0 \le j < |y|$ such that $y[j] = \sigma \in \Sigma$ and $x[j+d] = \circ$. Since

$$x[j+d] = s[i+j+d] \oplus s[i+j+k_1+d] \oplus \cdots \oplus s[i+j+k_{q-1}+d],$$

then at least one among $i+d, i+k_1+d, \ldots, i+k_{q-1}+d$ is not an occurrence of y, contradicting the hypothesis that $\mathcal{L}_y = \mathcal{L}_x + d$ (since $i, i+k_1, \ldots, i+k_{q-1} \in \mathcal{L}_x$). □

Lemma 6. *For each tiling motif x in the basis $\mathcal{B}$ with quorum q, there is at least one k for which $Merge_k = x$.*

Proof. If $|\mathcal{L}_x| = q$ and $\mathcal{L}_x = \{i_1, \ldots, i_q\}$ with $i_1 < \cdots < i_q$, then $x = Merge_k$ where k is the array of values $i_2 - i_1, i_3 - i_1, \ldots, i_q - i_1$. Let us now consider the case where $|\mathcal{L}_x| > q$. Given any q-tuple $i_1, \ldots, i_q \in \mathcal{L}_x$, let u_k denote $s[i_1..i_1 + |x| - 1] \oplus \cdots \oplus s[i_q..i_q + |x| - 1]$, which is a substring of $Merge_k$ introduced in Definition 9. We have that $x \preceq u_k$ and $\mathcal{L}_x = \bigcup_{i_1, i_2, \ldots, i_q \in \mathcal{L}_x} \mathcal{L}_{u_k}$. Since each u_k for $i_1, i_2, \ldots, i_q \in \mathcal{L}_x$ is a substring of $Merge_k$, we infer that $\mathcal{L}_x = \bigcup_{i_1, i_2, \ldots, i_q \in \mathcal{L}_x} (\mathcal{L}_{Merge_k} + \delta_k)$ where the δ_ks are non-negative integers. By Definition 7, if $Merge_k$ were different from x, then x would not be tiling, which is a contradiction. Therefore, at least one $Merge_k$ is x. □

The following property of tiling bases follows from Lemma 5 and 6.

Theorem 6. *Given a string s of length n and a quorum $q \ge 2$, let $\mathcal{M}$ be the set of $Merge_k$, for any of the $\binom{n-1}{q-1}$ possible choices of k for which $Merge_k$ exists. The basis $\mathcal{B}$ of tiling motifs for s satisfies $\mathcal{B} \subseteq \mathcal{M}$ and, therefore, the size of $\mathcal{B}$ is at most $\binom{n-1}{q-1}$.*

The tiling motifs in our basis appear in s for a total of $q\binom{n-1}{q-1}$ times at most. A variation of the algorithm given in Section 4.3 gives a pseudopolynomial-time complexity of

$$O\left(q^2\binom{n-1}{q-1}^2\right).$$

When this upper bound is combined with the lower bound of Section 5.1, we obtain that there exists a polynomial-time algorithm for finding the basis if and only if either $q = O(1)$ or $q = n - O(1)$.

6 Conclusions

The work presented in this paper is theoretical in nature, but it should be clear by now that its practical consequences, particularly—but not exclusively—for computational biology, are relevant. Whether motifs as patterns are used for inferring binding sites or repeats of any length, for characterizing sequences or as a filtering step in a whole genome comparison algorithm or before inferring PSSMs: We show that wild cards alone are not enough for a biologically satisfying definition of the patterns of interest. Simply throwing away the pattern-type of motif detection is not a good way to address the problem. This is confirmed by various biological publications [24], [7] as well as by the not yet published—but already publicly available—results of a first

motif detection competition http://bio.cs.washington.edu/assessment/. Even if patterns are not the best way of modeling biological features, they deserve an important function in any future improved algorithm for inferring motifs *ab initio* from biological sequences. As such, the purpose of this paper is to shed some further light on the inner structure of one important type of motif.

ACKNOWLEDGMENTS

Many suggestions from the anonymous referees greatly improved the original form of this paper. The authors are thankful to them for this and to M.H.ter Beek for improving the English. A preliminary version of the results in this paper has been described in the technical report IGM-2002-10, July 2002 [20], and in [21]. Work was partially supported by the French program bioinformatique EPST 2002 "Algorithms for Modelling and Inference Problems in Molecular Biology." N. Pisanti and R. Grossi were partially supported by the Italian PRIN project "ALINWEB: Algorithmics for Internet and the Web." M.-F. Sagot was partially supported by CNRS-INRIA-INRA-INSERM action BioInformatique and the Wellcome Trust Foundation. M. Crochemore was partially supported by CNRS action AlBio, NATO Science Programme grant PST.CLG.977017, and the Wellcome Trust Foundation.

REFERENCES

[1] A. Aho and M. Corasick, "Efficient String Matching: An Aid to Bibliographic Search," *Comm. ACM,* vol. 18, no. 6, pp. 333-340, 1975.
[2] A. Apostolico and L. Parida, "Incremental Paradigms of Motif Discovery," *J. Computational Biology,* vol. 11, no. 1, pp. 15-25, 2004.
[3] R. Baeza-Yates and G. Gonnet, "A New Approach to Text Searching," *Comm. ACM,* vol. 35, pp. 74-82, 1992.
[4] A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert, "Approaches to the Automatic Discovery of Patterns in Biosequences," *J. Computational Biology,* vol. 5, pp. 279-305, 1998.
[5] M. Crochemore and W. Rytter, *Jewels of Stringology.* World Scientific Publishing, 2002.
[6] E. Eskin, "From Profiles to Patterns and Back Again: A Branch and Bound Algorithm for Finding Near Optimal Motif Profiles," *RECOMB'04: Proc. Eighth Ann. Int'l Conf. Computational Molecular Biology,* pp. 115-124, 2004.
[7] E. Eskin, U. Keich, M. Gelfand, and P. Pevzner, "Genome-Wide Analysis of Bacterial Promoter Regions," *Proc. Pacific Symp. Biocomputing,* pp. 29-40, 2003.
[8] M. Fischer and M. Paterson, "String Matching and Other Products," *SIAM AMS Complexity of Computation,* R. Karp, ed., pp. 113-125, 1974.
[9] M. Gribskov, A. McLachlan, and D. Eisenberg, "Profile Analysis: Detection of Distantly Related Proteins," *Proc. Nat'l Academy of Sciences,* vol. 84, no. 13, pp. 4355-4358, 1987.
[10] D. Gusfield, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge Univ. Press, 1997.
[11] G.Z. Hertz and G.D. Stormo, "*Escherichia Coli* Promoter Sequences: Analysis and Prediction," *Methods in Enzymology,* vol. 273, pp. 30-42, 1996.
[12] C.E. Lawrence, S.F. Altschul, M.S. Boguski, J.S. Liu, A.F. Neuwald, and J.C. Wooton, "Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment," *Science,* vol. 262, pp. 208-214, 1993.
[13] C.E. Lawrence and A.A. Reilly, "An Expectation Maximization (EM) Algorithm for the Identification and Characterization of Common Sites in Unaligned Biopolymer Sequences," *Proteins: Structure, Function, and Genetics,* vol. 7, pp. 41-51, 1990.
[14] L. Marsan and M.-F. Sagot, "Algorithms for Extracting Structured Motifs Using a Suffix Tree with an Application to Promoter and Regulatory Site Consensus Identification," *J. Computational Biology,* vol. 7, pp. 345-362, 2000.
[15] W. Miller, "Comparison of Genomic DNA Sequences: Solved and Unsolved Problems," *Bioinformatics,* vol. 17, pp. 391-397, 2001.
[16] G. Myers, "A Fast Bit-Vector Algorithm for Approximate String Matching Based on Dynamic Programming," *J. ACM,* vol. 46, no. 3, pp. 395-415, 1999.
[17] L. Parida, I. Rigoutsos, A. Floratos, D. Platt, and Y. Gao, "Pattern Discovery on Character Sets and Real-Valued Data: Linear Bound on Irredundant Motifs and Efficient Polynomial Time Algorithm," *Proc. SIAM Symp. Discrete Algorithms (SODA),* 2000.
[18] L. Parida, I. Rigoutsos, and D. Platt, "An Output-Sensitive Flexible Pattern Discovery Algorithm," *Combinatorial Pattern Matching,* A. Amir and G. Landau, eds., pp. 131-142, Springer-Verlag, 2001.
[19] J. Pelfrne, S. Abdeddaïm, and J. Alexandre, "Extracting Approximate Patterns," *Combinatorial Pattern Matching,* pp. 328-347, Springer-Verlag, 2003.
[20] N. Pisanti, M. Crochemore, R. Grossi, and M.-F. Sagot, "A Basis for Repeated Motifs in Pattern Discovery and Text Mining," Technical Report IGM 2002-10, Institut Gaspard-Monge, Univ. of Marne-la-Vallée, July 2002.
[21] N. Pisanti, M. Crochemore, R. Grossi, and M.-F. Sagot, "A Basis of Tiling Motifs for Generating Repeated Patterns and Its Complexity for Higher Quorum," *Math. Foundations of Computer Science (MFCS),* B. Rovan and P. Vojtás, eds., pp. 622-631, Springer-Verlag, 2003.
[22] N. Pisanti, M. Crochemore, R. Grossi, and M.-F. Sagot, *String Algorithmics,* chapter: A Comparative Study of Bases for Motif Inference, pp. 195-225, KCL Press, 2004.
[23] D. Pollard, C. Bergman, J. Stoye, S. Celniker, and M. Eisen, "Benchmarking Tools for the Alignment of Functional Noncoding DNA," *BMC Bioinformatics,* vol. 5, pp. 6-23, 2004.
[24] A. Vanet, L. Marsan, and M.-F. Sagot, "Promoter Sequences and Algorithmical Methods for Identifying Them," *Research in Microbiology,* vol. 150, pp. 779-799, 1999.
[25] S. Wu and U. Manber, "Path-Matching Problems," *Algorithmica,* vol. 8, no. 2, pp. 89-101, 1992.

Nadia Pisanti received the laurea degree in computer science in 1996 from the University of Pisa (Italy), the French DEA in fundamental informatics with applications to genome treatment in 1998 from the University of Marne-la-Vallee (France), and the PhD degree in computer science in 2002 from the University of Pisa. She has been postdoctorate at INRIA and at the University of Paris 13 and she is currently a research fellow in the Department of Computer Science of the University of Pisa. Her interests are in computational biology and, in particular, in motifs extraction and genome rearrangement.

Maxime Crochemore received the PhD degree in 1978 and the Doctorat d'etat in 1983 from the University of Rouen. He received his first professorship position at the University of Paris-Nord in 1975 where he acted as President of the Department of Mathematics and Computer Science for two years. He became a professor at the University Paris 7 in 1989 and was involved in the creation of the University of Marne-la-Vallee where he is presently a professor. He also created the Computer Science Research Laboratory of this university in 1991. Since then, he has been the director of the laboratory, which now has around 45 permanent researchers. Professor Crochemore has been a senior research fellow at King's College London since 2002. He has been the recipient of several French grants on string algorithmics and bioinformatics. He participated in a good number of international projects in algorithmics and supervised 20 PhD students.

Roberto Grossi received the laurea degree in computer science in 1988, and the PhD degree in computer science in 1993, at the University of Pisa. He joined the University of Florence in 1993 as an associate researcher. Since 1998, he has been an associate professor of computer science in the Dipartimento di Informatica, University of Pisa. He has been visiting several international research institutions. His interests are in the design and analysis of algorithms and data structures, namely, dynamic and external memory algorithms, graph algorithms, experimental and algorithm engineering, fast lookup tables and dictionaries, pattern matching algorithms, text indexing, and compressed data structures.

Marie-France Sagot received the BSc degree in computer science from the University of Sao Paulo, Brazil, in 1991, the PhD degree in theoretical computer science and applications from the University of Marne-la-Vallee, France, in 1996, and the Habilitation from the same university in 2000. From 1997 to 2001, she worked as a research associate at the Pasteur Institute in Paris, France. In 2001, she moved to Lyon, France, as a research associate at the INRIA, the French National Institute for Research in Computer Science and Control. Since 2003, she has been director of research at the INRIA. Her research interests are in computational biology, algorithmics, and combinatorics.