Modeling local repeats on genomic sequences

Jacques Nicolas, Christine Rousseau, Anne Siegel, Pierre Peterlongo, François Coste, Patrick Durand, Sébastien Tempel, Anne-Sophie Valin, Frédéric Mahé

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Modeling local repeats on genomic sequences

Jacques Nicolas — Christine Rousseau — Anne Siegel
— Pierre Peterlongo — François Coste — Patrick Durand

— Sébastien Tempel — Anne-Sophie Valin — Frédéric Mahé

N° 6802

Décembre 2008

Thème BIO

ISSN 0249-6399 ISRN INRIA/RR--6802--FR+ENG

INRIA
RENNES

Modeling local repeats on genomic sequences

Jacques Nicolas* [†] , Christine Rousseau [†] , Anne Siegel [†] , Pierre Peterlongo [†] , François Coste [†] , Patrick Durand [‡] , Sébastien Tempel [§] , Anne-Sophie Valin [¶] , Frédéric Mahé [‖]

Thème BIO — Systèmes biologiques
Équipes-Projets Symbiose

Rapport de recherche n° 6802 — Décembre 2008 — 40 pages

Abstract: This paper deals with the specification and search of repeats of biological interest, i.e. repeats that may have a role in genomic structures or functions. Although some particular repeats such as tandem repeats have been well formalized, models developed so far remain of limited expressivity with respect to known forms of repeats in biological sequences. This paper introduces new general and realistic concepts characterizing potentially useful repeats in a sequence: Locality and several refinements around the Maximality concept. *Locality* is related to the distribution of occurrences of repeated elements and characterizes the way occurrences are clustered in this distribution. The associated notion of *neighborhood* allows to indirectly exhibit words with a distribution of occurrences that is correlated to a given distribution. *Maximality* is related to the contextual delimitation of the repeated units. We have extended the usual notion of maximality, working on the inclusion relation between repeats and taking into account larger contexts. Mainly, we introduced a new repeat concept, largest maximal repeats, looking for the existence of a subset of maximal occurrences of a repeated word instead of a global maximization.

We propose algorithms checking for local and refined maximal repeats using at the conceptual level a suffix tree data structure. Experiments on natural and artificial data further illustrate various aspects of this new setting. All programs are available on the genouest platform, at `http://genouest.org/modulome`.

Key-words: local repeats, largest maximal repeats, repeats, genomes, bioinformatics

* to whom correspondence should be addressed
† Irisa/Inria, Campus de Beaulieu, 35042 Rennes Cedex, France
‡ Korilog SARL, BP 34, 56190 Muzillac, France
§ Giri, 1925 Landings Dr. Mountain View, CA 94043, USA
¶ Ligue Nationale Contre le Cancer, 14 Rue Corvisart, 75013 Paris, France
‖ Ecobio, CNRS UMR 6553, Campus de Beaulieu, 35042 Rennes Cedex, France

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Modélisation de répétitions locales dans les séquences génomiques

Résumé : Cet article étudie la modélisation et la recherche de répétitions particulières ayant un intérêt biologique, c'est à dire pouvant jouer un rôle dans les structures ou les fonctions génomiques. Même si, à l'image des répétitions en tandem, certains types de répétitions ont déjà été bien formalisées, les modèles développés jusqu'alors souffrent d'une expressivité limitées par rapport aux formes connues des répétitions ayant un sens biologique.

Ce papier introduit de nouveaux concepts génériques et réalistes qui caractérisent des répétitions d'intérêt dans les séquences: la localité et plusieurs raffinements autour de la notion de la maximalité. La *localité* est liée à la distribution des occurrences des éléments répétés et caractérise la façon dont les occurrences sont groupées dans cette distribution. En outre, la notion associée de *voisinage* permet d'exhiber des corrélations entre distributions d'occurrences et ainsi de mettre à jour d'autres répétitions. La notion de *Maximalité* est liée à la délimitation des unités répétées. Nous avons étendu la notion communément admise de maximalité, par une approche basée sur l'inclusion entre répétitions et en considérant des contextes plus large qu'un unique caractère. En particulier, nous avons proposé un nouveau concept de répétitions, appelées plus grandes répétitions maximales, qui cherche à vérifier l'existence d'un sous ensemble d'occurrences maximales d'une répétition plutôt que de s'appuyer sur la recherche d'une maximisation globale.

Pour tous les nouveaux concepts introduits, nous proposons des algorithmes de détection dans les séquences basés au niveau conceptuel sur l'arbre des suffixes. Des résultats expérimentaux sur des données réelles et simulées illustrent l'intérêt de notre approche. Tous les programmes sont disponibles sur la plateforme genouest `http://genouest.org/modulome`.

Mots-clés : répétitions locales, plus grandes répétitions maximales, répétitions, génomes, bioinformatique

1 Introduction

A wide number of studies has revealed that genome sequences contain repeated sub-sequences playing major roles in the structure, the function, the dynamics and the evolution of genomes [18, 22, 11].

There exists a large literature covering the problem of finding repeats which can be mainly divided into three categories depending on the type of targeted repeats: exact repeats, repeats with errors and structured repeats.

Exact repeats have been extensively studied, leading to various concepts such as longest repeats [10, 16], maximal repeats [8, 12, 13, 23] and super-maximal repeats [8, 1]. The concept of maximal repeat is quite attractive and simply focuses on sequences present in at least two largest common blocks, without possible left or right extension, and without any biological *a priori*.

A second category of algorithms introduces an error model in the specification of repeated units, such as longest repeats with a block of don't cares [6], maximal pairs with bounded gap [5, 14], tandem repeats [25, 26, 4] and repeats with edit distance [15]. These kinds of algorithms are more adapted to analyze the many repeats contained in genome sequences that usually contain copies with multiple variations.

Finally, the third kind of approach targets the search of structured motifs that consist of an ordered collection of $p > 1$ parts separated from one another by constrained spacers [17, 9, 19, 20]. These algorithms are of particular interest in studying gene expression and gene regulation.

Our concern in this paper deals with the specification and search of repeats of biological interest, i.e. repeats that may have a role in genomic structures or functions. This problem is already addressed by algorithms from the two last categories. Algorithms allowing the treatment of errors can be used to locate genes n-plication ($n > 1$), and various types of tandem repeats which are, among others, constituents of centromeres and telomeres. Algorithms looking for structured repeats have been proposed to tackle the difficult problem of locating the set of short motifs constituting the regulatory factors involved in gene transcription.

However, all models developed so far remain of limited expressivity with respect to known forms of repeats in biological sequences. Transposable elements for instance exhibit complex copying patterns that are only partially understood so far. Further studies are needed to develop more realistic formal settings, while preserving the generality of the concepts. This paper is a contribution towards this goal. We introduce some new variations on repeats that capture important characteristics of observed repeats in the context of molecular biology. We examine the problem of defining and matching repeats appearing at particular positions or in particular contexts.

The search for repeats is always based on the detection of elementary units with several copies occurring in the studied sequence. *Maximal repeats*[1] have largely been used for this purpose throughout the literature since they can represent all other repeats and have a strong mathematical structure. Maximal repeats contain longest repeats and have a well defined structure of inclusion, their number is linear (at most n exact maximal repeats in a sequence of size

[1] A maximal repeat in a string S is a substring w such that there are two substrings awc and bwd of $\$S\$$ such that $a \neq b$ and $c \neq d$ (\$ is a special character that does not appear in S).

n), they can be computed in linear time using a suffix-tree based algorithm, and they can be used as basic blocks to compute error-prone repeats.
However, they suffer from important drawbacks with respect to "real" repeats.

First of all, it is hard in practice to distinguish maximal repeats from background noise. Pointing at large repeated words leads generally to meaningful units because the probability that they appear by chance is very low. In contrast, short words such as those that appear in gene regulation can occur at a frequency that is comparable to the frequency of random words of similar size. Maximality is a global concept that is defined with respect to all occurrences of a word. What makes short copies relevant has generally a local nature. This explain why a large part of this work is dedicated to the simple but powerful concept of *locality of repeats*. The clustering of occurrences in compact structures in the neighborhood of an initial position is described itself using a particular constraint of locality. We thus introduce a complementary notion, *neighbor repeats*, that is necessary to take into account the presence of elements indirectly local because they are in the vicinity of local units—some of those being eventually degenerated and non observable. Moreover, repeated units have locally an inclusion structure and it is generally the largest ones that are of interest. Within this idea, we have introduced a formal notion of *largest maximal repeat* that is a restriction of maximal repeats to those whose at least one of the occurrence is not covered by a bigger repeat. Furthermore it is worth to notice that no level of noise or variation is allowed between two copies. In practical cases, most of copies share a very similar sequence but are not fully identical. To tackle this problem, we extended the notion of *context* of the maximal repeats, usually limited to one single nucleotide. It permits to take into account small variations like SNP (Single Nucleotide Polymorphism). Another way to look at micro-variations on a set of occurrences is to observe the existence of a set of overlapping maximal repeats. A notion of *unit* reflects this structure. A repeated unit may either be a single word or be made of an overlapping assembly of more elementary units.

In addition to modeling these new concepts about repeats and their locality in genomic sequences, we propose methods and provide algorithms and their proof for their identification in a sequence S.

The paper is organized as follows: the next section provides a formalization of the algorithmic framework. Section 3 details one by one new concepts, presents their identification algorithm and provides results on artificial and biological sequences. Before concluding, Section 4 briefly discusses the choices that have been made in this study, emphasizing a more general research track on flexibility. The paper offers supplementary material at the end for detailed algorithmic or mathematical aspects.

2 Approach

We propose in this work several refinements around the concept of repeats in sequences and about their locality. For each concept, an algorithm is provided for their detection in genomic sequences. It is worth noticing that tools were actually developed; both for validating models on real genomic sequences and for making our approach available to the community. Please refer to the web site `http://genouest.org/modulome` for downloading codes.

We use the suffix tree data structure to describe the search for particular repeats. We recall that a (compact) suffix tree for a sequence S is a tree whose edges are labeled with non empty words; all internal nodes have at least two children and each suffix of S corresponds to exactly one path from the tree's root to a leaf. Each node may be associated with the word made of letters read on the path from the root to this node. Constructing such a tree for S can be achieved in time and space linear in the length of S (see [7] for a review).

We introduce a basic generic procedure computing an attribute on each node of this structure. This procedure, presented in Algorithm 1 (in the supplementary material), is called **Attributes**. It is a simple depth first recursive visit of the tree, computing on each node a value synthesizing the values of its children. It is linear in time and space with respect to the size of the analyzed sequence. People familiar with the computation of maximal repeats will find its description within this framework after proposition 3 in section 3.4. The procedure Attributes allows giving a more abstract presentation of algorithms while ensuring the linear basic complexity. Although such a procedure does not introduce a fundamentally new algorithmic scheme, it requires some non trivial choices in its description and it seems to be the first time that such an explicit formal description is provided for a systematic suffix tree data structure exploiting. This may help describing and extending "the myriad virtues" of this data structure [3] with compact and precise codes.

Of course, the tree can be considered only at the logical level and enhanced suffix arrays used instead [2] in practical implementations. However, its usage being more intuitive, we prefer using suffix tree for didactic purpose.

Our algorithms are all based on Attributes function calls, using each time specific procedures on nodes of the tree.

In order to illustrate the interest of newly introduced concepts, experiments were conducted on several genomes. Sequence data were made of complete genomes of *Archaea* and *Bacteria* with a size in the range $[2.4Mb, 3.5Mb]$ and of random sequences resulting from shuffled versions of these genomes. Shuffling was achieved via the shuffleseq function of the Emboss 4.0 package [24].

3 Enhancing repeats concepts

In this section we introduce one by one the new concepts we propose. The *Locality* notion, applicable to any kind of repeat seen later, is first exposed (Section 3.1). Then we explore various maximal repeats refinements in Sections 3.2, 3.3 and 3.4. In each section we provide first the context and the formalization, and second, we propose some experimental results.

3.1 Local repeats

The first property to be introduced in this paper concerns the distribution of occurrences of repeats. *Locality* is a simple restriction on repeats introducing a bounded size on the range of their occurrences. This range is formalized using a notion of scope.

The *scope* of a repeat gives access to the size of the regions where the repeat occurs in the sequences. Typically, it will get a low value for clustered repeats. Let us start with a very simple definition.

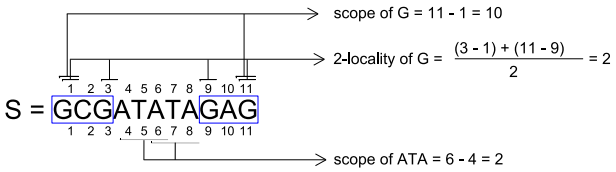

Figure 1: Locality of repeats and notion of scope.

Definition 1 (**scope**) The *scope* of a set of occurrences of words in a sequence is the difference between the last and the first occurrences positions.

The *scope* is a simple index related to the randomness of a distribution: a narrow distribution is typical of particularly interesting words. At first estimate, one could define the scope of a word in a sequence as the scope of its set of occurrences. Biological sequences exhibit in fact a more refined notion of locality since a same repeat may be found in different clusters—see for instance the CRISPR structure in [21]. If the distribution of a word is multimodal and if two clusters are far away, then its scope may be artificially large. This is why we introduce a parameter, the *number of modes* denoted by μ, that limits the maximum number of allowed clusters corresponding to a same repeat: allowing μ modes expresses the fact that occurrences may be clustered in 1 to μ groups. Then, we define the locality of a repeated word as the average scope of all groups.

Definition 2 (**μ-locality**) Let $\mu \in \mathbb{N}^+$ and w be a repeat with occurrences positions *pos*. Given an integer μ, a *μ-partition* is a partition $P = \{P_1, \ldots, P_{|P|}\}$ of *pos* that contains at most μ blocks (mutually exclusive subsets), each one with at least two elements. For each block P_i, we define $\min P_i$ (resp. $\max P_i$) as the smallest (resp. biggest) occurrence position contained in P_i.

We denote by $scope(P)$ the sum of scopes of all clusters of repeats, $scope(P) = \sum_{i=1}^{|P|}(\max P_i - \min P_i)$, and we denote by P^* the minimal μ-partition with respect to scope, then with respect to size. The *μ-locality of the repeat* is defined as the mean value of the scopes of clusters in this optimal partition:

$$\mu\_locality(w) = \frac{scope(P^*)}{|P^*|}.$$

Blocks of a partition represent clusters of repeats with an associated scope—each block has at least two occurrences in order to avoid considering locally isolated elements with null scope. The locality is calculated for an optimal partition from the point of view of the set of positions of all clusters. The 1-locality is just the scope, that is, the maximal difference between positions of occurrences.

Example 1 Let $S = GCGATATAGAG$. The scope of G is 10, the scope of A is 6, the scope of ATA, T or TA is 2 (see figure 1). The 2-locality of G is $(2 + 2)/2 = 2$, corresponding to the partition $\{\{1, 3\}, \{9, 11\}\}$ of its set of occurrences.

Increasing the number of modes μ never increases the value of the μ-locality since it is computed using a minimum over all partitions into 1 to μ subsets.

It has a noticeable effect on the sum of scopes only if clearly separated regions of occurrences exist. In practical applications, the user can fix the maximum number of modes by looking at the stabilization of this sum. In fact, if the number of modes is sufficiently large, the value of the locality converges to the mean interval between two occurrences and reflects the way occurrences are grouped together. We thus define the *asymptotic locality* or simply *locality of a repeat* as its μ-locality when μ tends towards its maximal value (the number of occurrences divided by two, since each block contains at least two elements).

Algorithm for computing the locality of a repeat

Enumerating all possible partitions of occurrences in order to get the μ-locality renders its computation unfeasible in most practical cases. We have established a nice property that allows to drastically reduce the set of interesting partitions and results in a quadratic algorithm, applying a dynamic programming approach.

Proposition 1 *Let $\mu \in \mathbb{N}$ and w be a repeat with n occurrences in a sequence whose positions are stored in $pos[1..n]$ ($n \geq 2$). The* μ_locality *of w is equal to*

$$\mu\_locality(w) = \frac{pos[n] - pos[1] - opt(\mu, n)}{|P|},$$

where there exists a partition $\{P_1, \cdots P_{|P|}\}$ of $pos[1..n]$, $|P| \leq \mu$ such that:

- *each block P_i corresponds to an interval on $pos[1..n]$ containing at least two positions, a_i and b_i: $a_i = \min P_i < b_i = \max P_i < a_{i+1}$;*
- *the extremes of blocks satisfy the relation $\sum_{i=1}^{|P|-1}(a_{i+1} - b_i) = opt(\mu, n)$;*
- *the function opt satisfies the following recurrent formulae:*

$$\begin{aligned} &opt(1, j) = opt(k, 2) = opt(k, 3) = 0 \\ &opt(k+1, j+2) = \max \left\{ \begin{array}{l} opt(k+1, j+1), \\ opt(k, j) + pos[j+1] - pos[j] \end{array} \right. \\ &\qquad\qquad (k \geq 1,\, j \geq 2). \end{aligned} \tag{1}$$

Proof 1 *of proposition 1:*
We first need to prove that the μ-locality of a repeat is obtained for a partition of its occurrences in at most μ clusters of consecutive *occurrences that each contains at least two elements. Consider a partition of the set of occurrences $\{P_1, \cdots P_{|P|}\}$ with $|P| \leq \mu$. The scope of each block is denoted by $scope(P_i) = \max P_i - \min P_i$.*

The partition is ordered so that $\min P_i < \min P_{i+1}$ for every $i \leq |P|$. Suppose now that there exists an index i such that $\max P_i > \min P_{i+1}$.
We show that swapping these two elements leads to a better partition.
Let $P'_i = P_i \cup \{\min P_{i+1}\} \setminus \{\max P_i\}$ and $P'_{i+1} = P_{i+1} \cup \{\max P_i\} \setminus \{\min P_{i+1}\}$. Then $\max P'_i = \max\{(P_i \setminus \{\max P_i\}), \min P_{i+1}\} < \max P_i$ and $\min P'_i = \min\{(P_i \setminus \{\max P_i\}), \min P_{i+1}\} \geq \min P_{i+1} \geq \min P_i$.
Similarly, $\min P'_{i+1} > \min P_{i+1}$ and $\max P'_{i+1} \leq \max P_{i+1}$. Hence $scope(P'_i) <$

$scope(P_i)$, $scope(P'_{i+1}) < scope(P_{i+1})$ and $(P_1, \ldots, P_i, P_{i+1} \ldots P_{|P|})$ is not an optimal partition for the sum of scopes.

We deduce that the smallest scope is reached for partitions that satisfy the relation $\max P_i < \min P_{i+1}$, that is, partitions I of $pos[1..n]$ into $|P|$ intervals $\{[a_1, b_1] \cap I, \cdots, [a_{|P|}, b_{|P|}] \cap I\}$ where a_i and b_i are elements of $pos[1..n]$ such that $a_i < b_i < a_{i+1}$. In particular, $a_1 = \min I = pos[1]$ and $b_{|P|} = \max I = pos[n]$. The μ-locality is $(\sum_{i=1}^{|P|}(b_i - a_i))/|P| = (pos[n] - pos[1] - \sum_{i=1}^{|P|-1}(a_{i+1} - b_i))/|P|$.

A consequence of this property is that minimizing the sum of scopes of blocks is equivalent to maximizing the quantity

$$\begin{array}{ll} V(\{P_1, \cdots, P_{|P|}\}) = \sum_{i=1}^{|P|-1}(\min P_{i+1} - \max P_i) & (\textit{if } |P| \geq 2) \\ V(\{P_{|P|}\}) = 0 & (\textit{if } |P| = 1) \end{array} .$$

Let us denote by $opt(\mu, n)$ the maximum of $V(\{P_1, \cdots, P_{|P|}\})$ for μ-partitions of $pos[1..n]$. We now have to prove that $opt(k, j)$ satisfies Eq. (1).

Assume that $j \geq 2$ and $k \geq 1$. Consider first an optimal partition of $pos[1..j]$ into at most k blocks $\{P_1, \cdots, P_{|P|}\}$, $|P| \leq k$. We have $V(\{P_1, \cdots, P_{|P|}\}) = opt(k, j)$. Add to this partition the block that contains two additional positions $P_{|P|+1} = \{pos[j+1], pos[j+2]\}$. We obtain a partition of the set $pos\{1..j+2\}$ into at most $k+1$ blocks, for which $V(\{P_1, \cdots, P_{|P|+1}\}) = V(\{P_1, \cdots, P_{|P|}\}) + \min P_{|P|+1} - \max P_{|P|} = opt(k, j) + pos[j+1] - pos[j]$. Since $opt(k+1, j+2)$ is greater than $V(\{P_1, \cdots, P_{|P|+1}\})$, we deduce that

$$opt(k+1, j+2) \geq opt(k, j) + pos[j+1] - pos[j].$$

Consider now an optimal partition of $pos\{1..j+1\}$ into at most $k+1$ blocks $\{P_1, \cdots, P_{|P|}\}$, $|P| \leq k$. Add to the last block the element $pos[j+2]$, that is: $P'_{|P|} = P_{|P|} \cup \{pos[j+2]\}$. We thus obtain a partition of $pos[0..j+2]$ into at most $k+1$ blocks, for which $V(\{P_1, \cdots, P_{|P|-1}, P'_{|P|}\}) \geq V(\{P_1, \cdots, P_{|P|-1}, P_{|P|}\}) = opt(k+1, j+1)$. Hence

$$opt(k+1, j+2) \geq opt(k+1, j+1).$$

We deduce that
$opt(k+1, j+2) \geq \max\{opt(k+1, j+1), opt(k, j) + pos[j+1] - pos[j]\}$.

Conversely, consider an optimal partition of $pos[0..j+2]$ into at most $k+1$ blocks $\{P_1, \cdots, P_{|P|}\}$. If $P_{|P|}$ contains exactly two elements, then we have $P_{|P|} = \{pos[j+2], pos[j+1]\}$ and by removing the last block we obtain a partition of $pos[0..j]$ with at most k blocks. Hence $V(\{P_1, \cdots, P_{|P|-1}\}) \leq opt(k, j)$, that is, $opt(k+1, j+2) - (pos[j+1] - pos[j]) \leq opt(k, j)$.
Otherwise, $P_{|P|}$ contains at least three elements. Removing the last element from $P_{|P|}$ leads to an admissible partition of $pos[0..j+1]$. Hence, $V(\{P_1, \cdots, P_{|P|} \setminus \{pos[j+1]\}\}) \leq opt(k+1, j+1)$. Removing the last element does not change the value of V, hence $opt(k+1, j+2) \leq opt(k+1, j+1)$. Since one of the cases occurs, we deduce that

$$\begin{aligned} opt(k+1, j+2) \leq \max\{&opt(k+1, j+1), \\ &opt(k-1, j) + pos[j+1] - pos[j]\}. \end{aligned}$$

Consequently, equation (1) is satisfied when $j \geq 2$ and $k \geq 1$. It remains to compute $opt(k, j)$ when $k = 1$, $j = 3$ or $j = 2$. In each case, the partition contains exactly one block, implying that $V(\{P_1\}) = 0$ so that $opt(k, 3) = opt(k, 2) = opt(1, j) = 0$.

The asymptotic locality may be computed by a similar technique.

Proposition 2 *Let $\mu \in \mathbb{N}$ and w be a repeat with n occurrences in a sequence stored in $pos[1..n]$ ($n \geq 2$). The* asymptotic-locality *of w is equal to*

$$asymptoticLocality(w) = \frac{pos[n] - pos[1] - opt(n)}{|P|},$$

where there exists a partition $\{P_1, \cdots P_{|P|}\}$ of $pos[1..n]$, $|P| \leq \mu$ such that:

- *each block P_i corresponds to an interval on $pos[1..n]$ containing at least two positions, a_i and b_i: $a_i = \min P_i < b_i = \max P_i < a_{i+1}$;*
- *the extremes of blocks satisfy the relation $\sum_{i=1}^{|P|-1}(a_{i+1} - b_i) = opt(n)$;*
- *the function opt satisfies the following recurrent formulae:*

$$\begin{aligned} &opt(2) = opt(3) = 0; opt(4) = pos[3] - pos[2] \\ &opt(j+3) = \max \left\{ \begin{array}{l} opt(j) + pos[j+1] - pos[j] \\ opt(j+1) + pos[j+2] - pos[j+1] \end{array} \right. \\ &\qquad\qquad (j \geq 2). \end{aligned} \tag{2}$$

Proof 2 *of proposition 2 With a proof similar to the one of Proposition 1, it is easy to prove that function opt satisfies the relation*

$$\begin{aligned} &opt(2) = opt(3) = 0 \\ &opt(j+2) = \max(opt(j+1), opt(j) + pos[j+1] - pos[j]),\ (j \geq 2) \end{aligned}$$

Eq. (2) is a one step unfolding of this equation and is equivalent to it for every $j \geq 2$.

It remains to compute $opt(4)$. $opt(4) = \max(opt(3), opt(2) + pos[3] - pos[2]) = pos[3] - pos[2]$.

Algorithms computing μ-locality from a suffix tree representation of a sequence directly follow from these propositions.

Algorithm for computing the locality of all repeated factors of a sequence

The **1-locality** of factors of a sequence may be computed in linear time and space, using the following function call:
$Attributes(root, init, updateMinMax, position, oneScope)$ that stores in $Synth$ (see Algorithm 1) the minimal and maximal positions of the occurrences of a factor, using the following functions:

- *init* gives value $+\infty$ to $Synth.min$ and 0 to $Synth.max$;
- $updateMinMax$ updates $Synth$ on a parent node with respect to min and max positions of its children (see Algorithm 2);
- *position* returns leaves positions in $Synth.min$ and $Synth.max$;

- *oneScope* computes $Node.scope := Synth.max - Synth.min$.

Computing the μ-locality of factors of a sequence for arbitrary values of μ is more complex. With $\mu \geq 2$, the μ-locality of a factor depends on all its occurrences positions and those are managed in Synth (see Algorithm 1) as a sorted list. The locality is computed with the following call:

$$Attributes(root, emptyList, mergeLists, getPos, muLocality)$$

- *emptyList*: returns in *Synth* an empty list;
- *mergeLists*: merges the sorted positions lists of the current node and one of its children;
- *getPos* returns the position of a leaf in *Synth*;
- *muLocality* computes, given the positions list in *Synth*, the μ-locality of the considered node. *muLocality* is a dynamic programming scheme directly based on the recurrence equation given in proposition 1 (see algorithm 3).

*C*omplexity: μ gets generally low values—the μ-locality converges in practice towards the limit locality for values of the order of 20—and can be considered as a constant. Algorithm 3 requires a computation in $O(n)$ steps. The *Synth* structure requires $O(n)$ space. Computing the μ-locality of all factors of a sequence requires thus $O(n^2)$ in time and $O(n)$ in space.

The same way, the asymptotic locality may be computed with function call

$$Attributes(root, emptyList, mergeLists, getPos, asymptoticLocality)$$

where asymptoticLocality is given in Algorithm 4, using the equation given in proposition 2.

Complexity: Computing the limit-locality of all factors of a sequence does not depend on μ but requires the same complexity than computing the μ-locality—i.e. $O(n^2)$ in time and $O(n)$ in space.

Locality experiments

We present first the experimental setting we have used for all our results.

All repeats and all maximal repeats with size varying between 18 and 80 nucleotides have been considered. These bounds have been chosen from preliminary studies on the distribution of occurrences in the set of genomic sequences, solely for presentation purpose: it avoids a too large dynamics in the presented curves while maintaining a large number of repeats. Mode μ is varying from 1 to 10. Greater values have a significant impact only for words with numerous occurrences and these are negligible in the given range of sizes.

In all 3D graphics, the background level for random sequences remains flat (blue curves), simply showing that the distribution of repeated words is independant of μ_locality when sequences have no particular structure. The hilly landscapes (in red) are curves for real genomic sequences. Contrary to random sequences having the same composition, genomic sequences exhibit various behaviors that heavily depend on the species at hand. The distribution of maximal

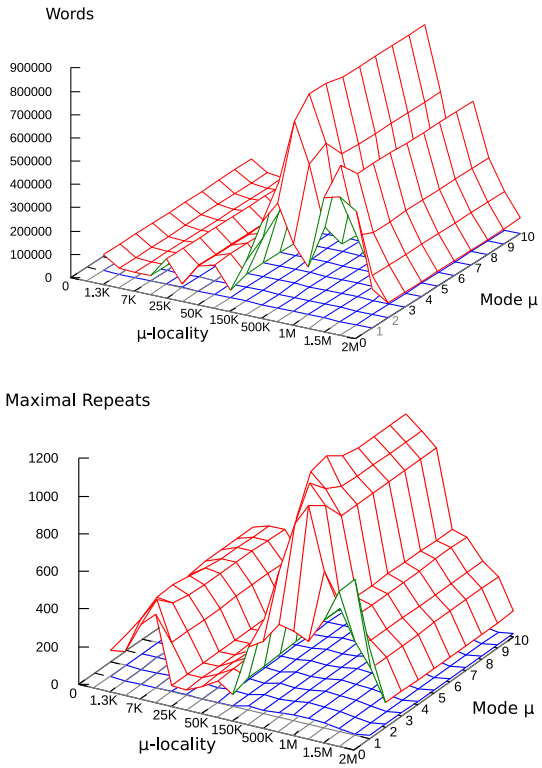

Figure 2: Study of archaeon NC_002754 (*Sulfolobus solfataricus*). Number of repeats, number of maximal repeats with respect to various μ_locality values.

repeats and of their occurrences and of largest maximal repeats are given with respect to mode for normal and shuffled sequences.

Thus Figure 2 and Figure 3 display an overview of the distribution of repeated words for various intervals of μ_locality for two different archaea.

This demonstrates clearly that the 3D profiles for the whole set of words and for the reduced set of maximal repeat may be rather similar for some species or on the contrary sharply differ. In figure 2 that displays results for the archaeon *Sulfolobus solfataricus*, maximality is an efficient filter and absolute numbers of words to be considered differ by a factor of 750, but the proportion of repeats is almost the same for a given range of mode and μ_locality if the maximality criterion is required or not. A small supplementary peak seems to exist for a locality value in the range 5-10K but it remains a minor difference. Figure 3, computed on the genome of the archaeon *Methanoculleus marisnigri*, exhibits a very different scheme where most of words occur very locally (25K) and are not maximal repeats and where maximal repeats have a wider scope, even for large values of mode.

The same way, Figure 4 displays an overview of the value of μ_locality in different contexts for two bacteria, *Desulfotalea psychrophila* and *Geobacillus thermodenitrificans*, the first exhibiting a clear peak in the range of small locality and the second lacking this property. Globally, our empirical studies show a clear species-dependant effect that allows to filter potentially meaningful words on the

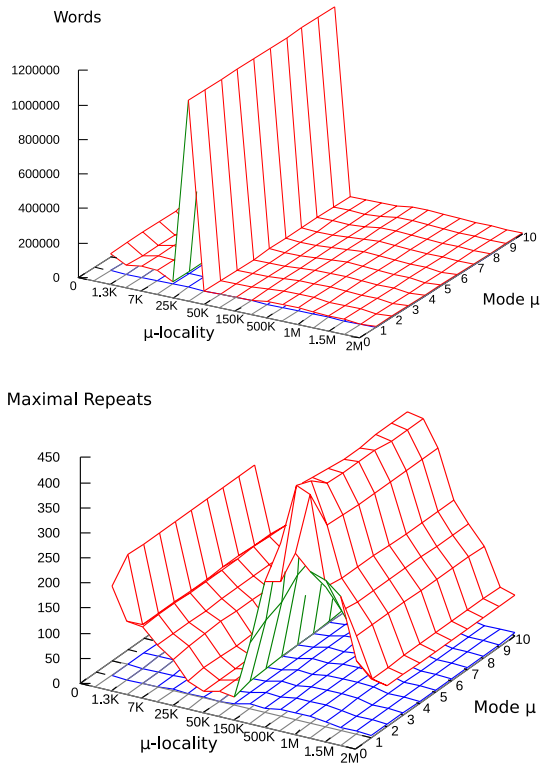

Figure 3: Study of archaeon NC_009051 (*Methanoculleus marisnigri* JR1). Number of repeats, number of maximal repeats with respect to various μ_locality values.

basis of the locality and also points at the interest of studying maximality, a concept that will be further extended in section 3.3 of this paper.

A more precise view of the occurrences of main maximal repeats is given in Figure 5 that displays for three bacteria an overview of the distribution of occurrences of repeats and maximal repeats in a defined range of locality for different mode values.

In all these 2D curves, the locality as been fixed in an interval corresponding to a maximum number of observations of maximal repeats. The number of maximal repeats is represented by red squares ($M1$ curves) for normal sequences and blue circles ($M2$ curves) for their shuffled counterpart. The number of occurrences of maximal repeats is represented by green diagonal crosses ($O1$ curves) for normal sequences and purple squares ($O2$ curves) for their shuffled counterpart. Note that the scale is given on the left for maximal repeats and on the right for occurrences. The effect of μ is best viewed on such curves, which display the variation of frequencies with respect to mode. Very different profiles are observed for different species (see Figure 5 for a comparison of three bacteria). Generally, the number of words that are maximal repeats stabilizes rapidly after a small peak (mode value greater than 3). In contrast, the number of occurrences may vary much more, as it is illustrated in the figure for the *Geobacillus kaustophilus* genome. Note that a variation is also observed for random sequences, showing a likely bias with respect to composition. The real impact of the mode parameter has to be further evaluated in such practical cases, but extrema may nevertheless points to interesting values of modes, that is, interesting number of regions with clustered repeats. The restriction of maximal repeat to largest maximal repeats will provide a more direct effect of the number of modes on the number of words (see figure 7).

3.2 Vicarious local repeats

We stated in the introduction that there exists another way to look at constrained distributions of words. Some words are particularly interesting in an indirect way, because they only occur in association with local words (words with a given μ_locality). We define below this possibility in the most general context. It aims at searching for words whose positions are correlated with positions of a given set of words. It is thus a powerful tool in every application where one is looking for word dependencies inside a sequence.

Definition 3 (δ-neighbor of a language in a sequence) A δ-neighbor (with relative support τ) of a language L in a sequence S is a word such that all its occurrences in S occupy positions that do not intersect with positions of words in L and all (or more than τ percent) of its occurrences are separated by at most δ letters from an occurrence of a word in L.

Example 2 Let $S = CTCCCCTTACCTTATTCATTCCTC$, $L = \{AT, TA\}$. Words T or TT are not 1-neighbors of L because some of their occurrences intersect with occurrences of words of L. Word C is not a 1-neighbor of L since it has 5 occurrences at the right maximal distance from L and 6 occurrences that are too far from words of L. So, it is a 1-neighbor of L only for a support τ that is below 45%. Words CC and CCT are the 1-neighbors repeats with support 50% of L (CCT is also a maximal repeat, see next section).

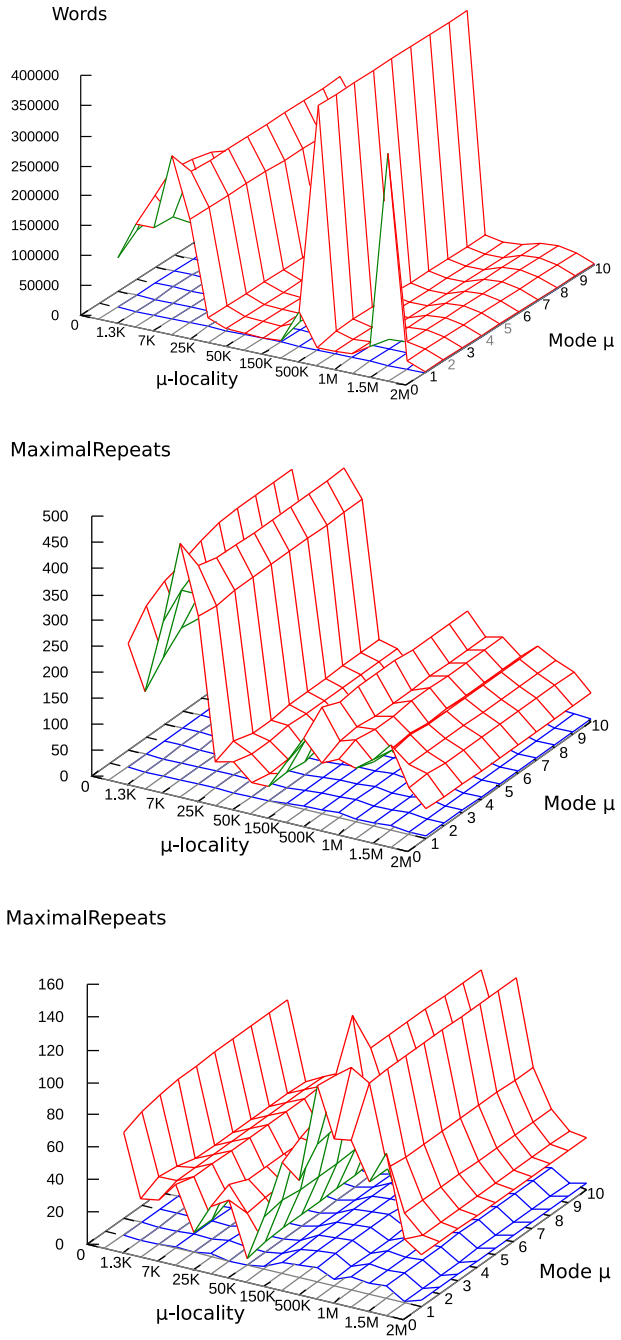

Figure 4: Study of bacterium NC_006138 (*Desulfotalea psychrophila*). Number of repeats, number of maximal repeats with respect to various μ_locality values. Study of bacterium NC_009328 (*Geobacillus thermodenitrificans* NG80-2). Number of maximal repeats with respect to various μ_locality values.

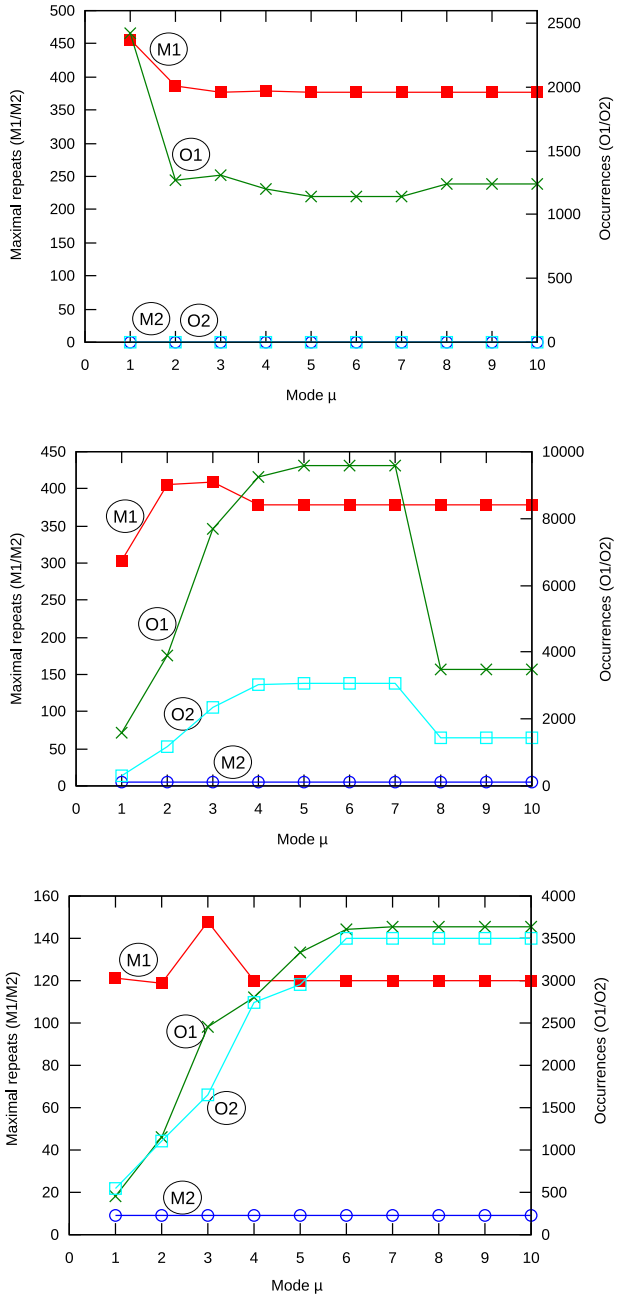

Figure 5: Number of words and of occurrences of maximal repeats with respect to the number of modes for a fixed interval of locality. Study of bacteria NC_006138 (*Desulfotalea psychrophila*, fixed locality from 3Kb to 7Kb), NC_006510 (*Geobacillus kaustophilus* HTA426, fixed locality from 500Kb to 750Kb), NC_009328 (*Geobacillus thermodenitrificans* NG80-2, fixed locality from 300Kb to 500Kb). The number of maximal repeats is represented by curve $M1$ for normal sequences and curve $M2$ for their shuffled counterpart. The number of occurrences of maximal repeats is represented by curve $O1$ for normal sequences and curve $O2$ for their shuffled counterpart.

Note that neighboring depends on two parameters, its maximal size (width, maximal distance) and its minimal support in the set of occurrences. Support is intended to reflect the possibility that some occurrences of the reference words may have disappeared and that consequently neighbors may occur without being associated to references. In practical applications even the default value $\tau = 100$ allows detecting associations.

Computing δ-neighbors

We have proposed an algorithm for computing δ-neighbors with support τ of a language represented as a sorted list L of starting and ending positions in a sequence, in increasing value of starting positions[2]. The notion of δ-neighbor is mainly interesting for non repeated word or maximal repeats. We thus assume that it is sufficient to produce words that are of maximal length with respect to the associated set of starting occurrences. Words that are δ-neighbors with support τ of L may be computed with a function call described in the supplementary material. The idea is first to check for each position of a leaf in the suffix tree the distance to the closest position in L. This establishes easily the status of δ-neighbors for non repeated words. It is achieved by procedure *classOcc* described in algorithm6. Then, it is almost sufficient (details appear in the supplementary material) to maintain from children to parents in the suffix tree two data structures, *mapL* and *mapR*. This is the role of procedure *neighbors* given in algorithm5. Parents correspond to prefixes of children and thus to shorter words. The structure *mapL* manages the distribution with respect to some distance value of occurrences that are in the suitable neighborhood of an element of L at its left but overlapping at the given distance the next element in L. If the size of the word tested in a node becomes sufficiently small, then the word is no more overlapping and may be safely added to δ-neighbors. *mapR* is a similar structure managing occurrences that are to the right and overlapping an element of L. If the size of the word tested in a node becomes sufficiently small, then the word is no more overlapping and moreover may be at the right distance from an element of L. It may be then safely added to δ-neighbors.

The right behavior of the algorithm is supported by an analysis tht is also given as supplementary material.

Complexity: The *classOcc* procedure requires a search in a sorted list of size p, requiring $O(log(p))$ steps, with p the cardinality of L. The procedure *neighbors* requires a loop on elements of associations lists *Synth.mapL* and *Synth.mapR*, which contain at most $O(n)$ elements. If mappings are implemented as hash-tables or arrays of size n, updating these lists is virtually $O(1)$. Computing delta-neighbors of L thus requires $O(n^2 + nlog(p))$ in time and $O(n)$ in space.

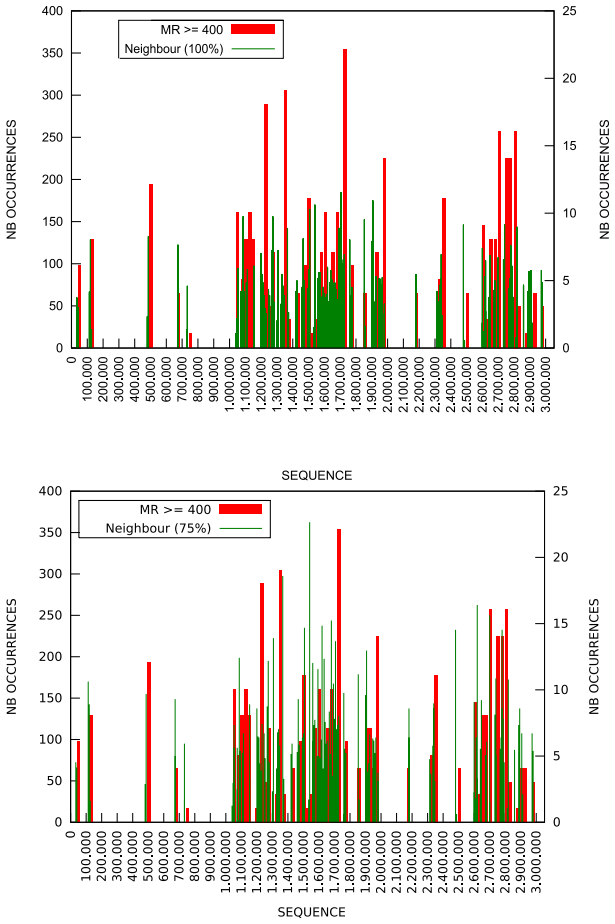

Figure 6: Study of archaeon NC_002754 (*Sulfolobus solfataricus*). Number of occurrences of maximal repeats of size greater than 400bp (red, scale on the right) and number of occurrences of 2Kb-neighbors along the sequence (green, scale on the left) for 100% support and 75% support.

δ-neighbor experiments

The algorithm has been tested on various genomes. Figure 6 shows the results for the archeon *Sulfolobus solfataricus* which has a total size of 3Mb and contains more than 1.5 million of maximal repeats with more than 32 million occurrences. Elements of L have been fixed to most remarkable maximal repeats: those of size greater than 400 nucleotides. The corresponding 135 elements have 381 occurrences that are not randomly distributed in the genome. Almost 3/4 of them appear in regions corresponding to transposons. This fact is well known and points to hot regions of the genome with respect to variations. The rest of elements are mainly non-annotated, non-coding regions and could be traces of other mobile regions of the genome.

The first remarkable point is that even for a 100% quorum τ and a small distance $\delta = 2Kb$, a very demanding threshold requiring all considered words to have a distribution included in a close neighborhood of the selected distribution, numerous neighbor words have been found. The mean number of neighbor occurrences for a given maximal repeat position is $11,235$ for $\tau = 100\%$ and $16,851$ for $\tau = 75\%$. The lower threshold 75% confirms the tendency of hot spots in the distribution, that is, regions with a high density of neighbors: the number of regions with more than 100 occurrences goes from 32 for $\tau = 100\%$ to 70 for $\tau = 75\%$. Moreover, size of words has a wide distribution, ranging from very small word (size 8) to large words (maximal size 387). This type of behavior is not observed on shuffled sequences and the observation of such correlations allows detecting interesting repeats that may have locally a single copy.

3.3 Largest maximal repeats (LMR)

The building bricks of repeats are based on words delimited in terms of their contexts in sequences and formalized in terms of *maximality*.

However, maximal repeats may be included in larger ones if they have more occurrences than this including repeat. This emphasizes the fact that maximal has not to be confused with largest. Super-maximal repeats have been proposed for this purpose. But super-maximal repeats, which filter only maximal repeats that are not substrings of others, are of little use in practice because many positions of repeats are not covered by super-maximal repeats.

In fact, the important characteristics of maximal repeats is that they both represent all other repeats and cover all positions of repeat occurrences. Then the question is: Are maximal repeats optimal with respect to these conditions? The answer appears to be negative. Consider for instance the sequence $ACACGAGAGG$. Maximal repeats are A, AC, G and GAG. But A is not maximal in the sense that AC or GAG are present at every position where A occurs. To the contrary G, which is not a super maximal repeat, cannot be discarded without losing the information of an existing repeat at the end of the sequence. We propose a new definition, largest maximal repeat, that aims at better circumscribing this notion of maximality.

[2] In fact, dummy elements have to be added to L to avoid side effects: If m is the number of positions in L and n is the size of the sequence, $L[0]$ is set to $(-n, -n)$ and $L[m+1]$ to $(2n, 2n)$

Definition 4 (Largest maximal repeat) Let S be a sequence with a set of repeats R. An LMR in S is a repeat $r \in R$ such that at least one occurrence of r is not strictly included in an occurrence of another element of R.

It is worth noticing this definition applies on any kind of repeat. Moreover all these notions of maximality may be ordered: a super maximal repeat is a largest maximal repeat that is itself a maximal repeat.

Example 3 Let $S_1 = GAGAGT$. We consider maximal repeats in S_1 that are GAG occurring positions 1 and 3 and G occurring position 1, 3 and 5. GAG is a largest maximal repeat. However every G occurrence is strictly included in a GAG occurrence, thus G is not a LMR.

Now consider $S_2 = GAGAGG$. In this case GAG is still a MR occurring positions 1 and 3, while G is a MR occurring positions 1, 3, 5 and 6. GAG remains a largest maximal repeat, however, G is a LMR as its occurrence in position 6 is not covered by any other repeat occurrence.

Fortunately, largest maximal repeats keep the nice properties of maximal repeats for their computation: they can be extracted in linear time and space.

However, bounding for a sequence of size n its maximal number of largest maximal repeat (between $\frac{n}{2}$ and n) and the corresponding maximal number of occurrences (between n and n^2) remains an open question.

Computing LMR

The algorithm computing largest maximal repeats is a simple filter on the results of the algorithm for computing maximal repeats. For each maximal repeat that has children leaves in the suffix tree, the starting and ending positions of these children leaves are kept. It is sufficient to scan in linear time the sequence from left to right then, keeping at each position the detected maximal repeats ending after the end of the previous one.

LMR experiments

We have computed LMR on NC_002754, NC_006138 and NC_009051 genomes and their shuffled versions. Results are shown on figure 7. At first glance, one may remark that around half the maximal repeats are not LMR. This is an important reduction since no repeat is lost in choosing a LMR representation. The most remarkable effect is on the influence of the mode parameter. Indeed, while mode has only a limited influence on maximal repeats, a hollow on the number of LMR is observed for mode values between 2 and 7 in all genomes, in particular for μ-locality around 300K. Although we have no biological explanation of this phenomenon, it seems worth further biological investigations and it shows both that the regrouping of copies is not random in biological sequences and that largest maximal repeats allow a finer analysis of the distribution of copies. A last observation is that on shuffled versions, as expected as well for maximal repeats, almost no LMR were found.

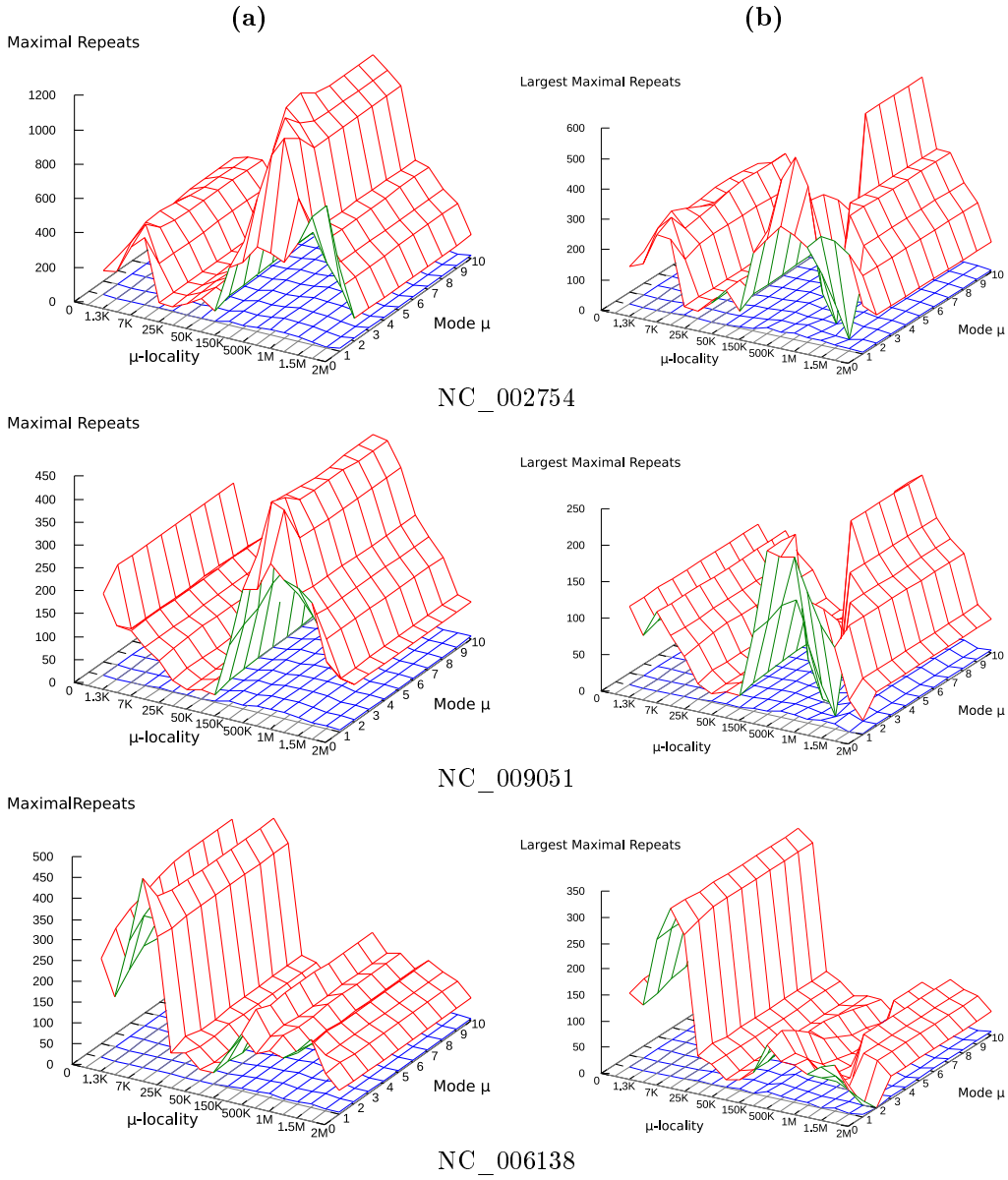

Figure 7: Study on archeon NC_002754 and NC_009051 and on bacteria NC_006138 species. With respect to their mode and their μ-locality, we present largest maximal repeats on column **(b)**. In addition, we recall values obtains with maximal repeats on same parameters on column **(a)**.

3.4 Context of repeats and k-units

The *contexts* of a repeat in a sequence corresponds to flanking letters of occurrences of this repeat in the sequence. Maximality expresses that words are selected on the basis of the existence of at least two occurrences with two different left and right contexts. Our practice on genomic sequences drive us to extend the standard notion of maximality. We propose to look at contexts that are possibly not reduced to a single letter or that may belong to different words. The paper addresses in particular the case of contexts of size two, since they are closely related to the presence of single point mutations (SNP), an important source of individual variations (polymorphism) in genomic sequences. More generally for a given word, one may be interested in the maximal size of left and right contexts for which at each position letters differ among their occurrences. Another characteristic observation in genomic sequences that we propose to model is the presence of sets of overlapping occurrences of maximal repeats. It is interesting in such a case to consider the set as a global unit.

Contexts and discriminant contexts

Definition 5 (context of a word in a sequence) A *context* of a word w in a sequence S is a pair of words (l, r)—the left and right context—such that lwr is a subword of S.

For instance, let $S = ATAT$. The contexts of A are (ϵ, ϵ), (ϵ, T), (ϵ, TA), (ϵ, TAT), (T, ϵ), (T, T) and (AT, T). The contexts of AT are (ϵ, ϵ), (ϵ, A), (ϵ, AT), (T, ϵ), and (AT, ϵ).

Among contexts, those that differ from one occurrence to the other are particularly useful in delimiting repeats. They are called discriminant contexts. Let us denote $S[a, b]$ the substring of sequence S from position a to position b.

Definition 6 (discriminant context) A context $(l[1, s_l], r[1, s_r])$ is a *discriminant context* with respect to a word w in a sequence S if and only if for some context $(l'[1, s_l], r'[1, s_r])$ of w: $\forall i \in [1, s_l], l[i] \neq l'[i]$ and $\forall j \in [1, s_r], r[j] \neq r'[j]$. (s_l, s_r) is the *size* of the context.

The definition is further illustrated in Figure 8. All words have at least (trivially) an empty discriminant context—size (0,0) context. Words may have several copies and only an empty discriminant context, as it is illustrated in the figure on word T. We recall that a usual maximal repeat in a sequence is a word with at least two occurrences in this sequence (maximal pair) such that the extension of these two occurrences by one position either to the left or to the right leads to two different words. With our vocabulary, maximal repeats are words with discriminant context of size (1,1). The size of the maximal discriminant context of a word characterizes in some way how it stands out from the background as a separated meaningful unit. We will show that it allows defining a natural extension of the well-studied notion of maximal repeat that is more robust with respect to repeats with errors.

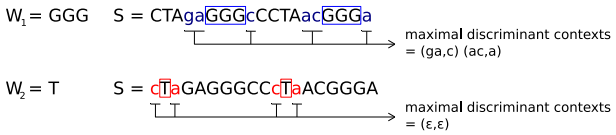

Figure 8: Discriminant contexts of words.

Building and displaying maximal repeats in a sequence

We first recall a nice property of standard maximal repeats with discriminant contexts of size 1:

Proposition 3 *The set of maximal repeats of a sequence is a subtree of the suffix tree of this sequence.*

Indeed, if w is a maximal repeat then it is associated to a node in the suffix tree with two different occurrences l_1w and l_2w in the sequence. For all v that is a proper prefix of w, (l_1, r) and (l_2, r) are two contexts of v in the sequence for some letter r. Now, if V has two occurrences with a different right context, v is a node parent of the node of w in the suffix tree and has at least two contexts (l', r_1) and (l'', r_2), with $r_1 \neq r_2$. At least one of l_1, l_2 differs from l' and l'' and at least one of r_1, r_2 differs from r, proving that v is a maximal repeat. The number of maximal repeats is thus at most $n-1$ in a sequence of size n. The property provides the basis of a linear procedure building them from the suffix tree of the sequence.
Maximal repeats may be computed using the call

$$Attributes(root, empty, addLetter, prevLetter, isDiscriminant),$$

where

- *Synth* represents the set of left contexts (letters before the starting position of each occurrence) of the word at the current node. To be precise, it contains a letter which belongs to $\{\top, \bot\} \cup \Sigma$, where $\top$ stands for any set with cardinality at least 2 and $\bot$ corresponds to the empty set. This one-letter encoding avoids storing the whole context;
- *empty* set *Synth* to the empty set ($\bot$);
- *addLetter* set *Synth* to the *Synth* value of its child if its value is $\bot$ and to $\top$ if its value differs from the value of its child;
- *prevLetter* returns in *Synth* the letter of the sequence at the position preceding the position of the leaf;
- *is_discriminant* returns a boolean indicating if $(Synth = \top)$.

Identifying robust maximal repeats via the size of their contexts

The next step in this work consisted in studying the effect of slight variations on the behavior of maximal repeats. It is indeed important in biological applications to take into account the fact that single point mutations, that is,

substitutions of one character by another one, are frequent in occurrences. Let u and v be two words and a, a', b, b', c and c' be letters such that $a \neq a'$, $b \neq b'$ and $c \neq c'$. Consider a discriminant context for two occurrences of a word ubv, ($aubvc$, $a'ubvc'$). In case of substitution of b by b', it is possible to get instead the pair ($aubvc$, $a'ub'vc'$) and the effect is to split the maximal repeat in two smaller parts u and v. Note that this behavior is also observed in case of insertion-deletion of a nucleotide.

In practice, a train of overlapping maximal repeats are generally observed in genomic sequences instead of clearly separated ones. This may be simply explained by a generalization of the previous reasoning, introducing simultaneously several points of modifications. Given a sequence $uxvyw$, two different points of modification will generate two sequences $ux'vyw$ and $uxvy'w$, and this will result in overlapping maximal repeats uxv and vyw.

Some maximal repeats are robust with respect to these variations and are particularly interesting to delimit borders of repeated regions.

Definition 7 (k-maximal repeats or kMR) If k and k' are strictly positive integers, a *(k,k')-maximal repeat* in a sequence is a word with a discriminant context of size (k, k'). When $k = k'$, it is a k-maximal repeat.

The proposition 3 on the global structure of maximal repeats is not true for (k, k')-maximal-repeat if $k > 1$ or $k' > 1$. For instance, consider sequence $TTCAGCATGCT$. The word CA is a (2,1)-maximal repeat since (TT, G) and (AG, T) are discriminating contexts. However C is a maximal repeat but not a (2,1)-maximal repeat since its contexts are (TT, A), (AG, A) and (TG, T) and cannot be discriminated.

By increasing the value of k or k', one restricts the set of admissible repeats to robust maximal repeats with respect to possible variations of their content. In particular, all maximal repeats subject to single point substitutions as previously described are not k-maximal repeats for $k \geq 2$.
An important practical consequence of this fact is that it is possible to reconstruct maximal repeats that have been split by single point mutations.

Proposition 4 *Maximal repeats that have been split by single point mutations correspond to characteristic occurrences of maximal repeats that are not (2,1)-maximal—or symmetrically not (1,2)-maximal.*

Proof 3 *Indeed, if ubv is a maximal repeat that has been subject to a mutation b/b', then v is a (1,1)-MR but not a (2,1)-MR. Conversely, given v, a (1,1)-MR that is not a (2,1)-MR, there exists an occurrence xubvc and an occurrence $x'ub'vc'$ in the sequence where b and b' and c and c' are different letters (v is a MR), and where u is a non empty word (or else v would be a (2,1)-MR). Moreover, since the occurrences of v are located at different positions, there must exist a word u such that x and x' are words that do not end with the same letter. Thus u is a MR that is not (1,2)-maximal and ubv is the trace of a maximal repeat subject to a single point mutation on letter b.*

Due to the interest of the SNP identification task, we have focused our study on the design of an algorithm for the determination of (2,1)-maximal repeats. It needs almost no extra memory space with respect to the computation of simple maximal repeats. We leave open the issue of finding (k, k')-maximal repeats

for larger values of k and k'. The suffix tree data structure is not well suited for the study of right contexts. A possible extension would be to consider a generalized suffix tree on the sequence and the reverse sequence. For $(k,1)$-maximal repeats, $k > 2$, a simple algorithm would be to keep each left context of size k at each node but this is not tractable in space for very large sequences. In fact, $(k+1,1)$-maximal repeats are a subset of $(k,1)$-maximal repeats and it might be possible to filter them iteratively.

Computing the list of (2,1)-maximal repeats

Taking into account left contexts of size 2 is quite direct with the suffix tree data structure. It is sufficient to manage left contexts that appear on suffixes at the leaf nodes and to compute for each node the union of the contexts of its children. In fact, we use a small refinement for contexts storing. Contexts are indexed on their first letter—e.g., if context AA and AC have to be stored, the set $\{A, C\}$ is stored at index A. Moreover, one character is sufficient to represent the set that has to be stored. The empty set is represented by the special letter $\perp$ and a set of size greater than one is represented by the special letter $\top$. A singleton is simply represented by the letter it contains. This way, storing the contexts only requires a space proportional to the size of the alphabet.

Overall, **(2,1)-Maximal repeats** may be computed in linear time and space, using the call:

$$Attributes(root, empty, cumul, prevLetters, isDiscriminant),$$

where

- *Synth* is made of a boolean Synth.21MR indicating if the word is a (2,1)-MR and a function representing the set of left contexts. Let w denote the word at the current node. For each letter $\sigma \in \Sigma$, $Synth(\sigma)$ represents the set of letters $\{a$ such that σaw belongs to the sequence$\}$. To be precise, it contains a letter which belongs to $\{\top, \perp\} \cup \Sigma$, where $\top$ stands for any set with cardinality at least 2 and $\perp$ corresponds to the empty set;
- *empty* sets $Synth(\sigma)$ to the empty set ($\perp$) for each letter $\sigma \in \Sigma$;
- *cumul* is described in algorithm 7. It updates the set of letters preceding the pointed word in the suffix tree;
- *prevLetters* returns in *Synth* the left context of the suffix w corresponding to the leaf: if abw is a suffix, $Synth(a)$ is set to b and the other values of the function to $\perp$;
- *isDiscriminant* returns the value of Synth.21MR.

The correctness of the algorithm is established in proof 5 in the supplementary material.

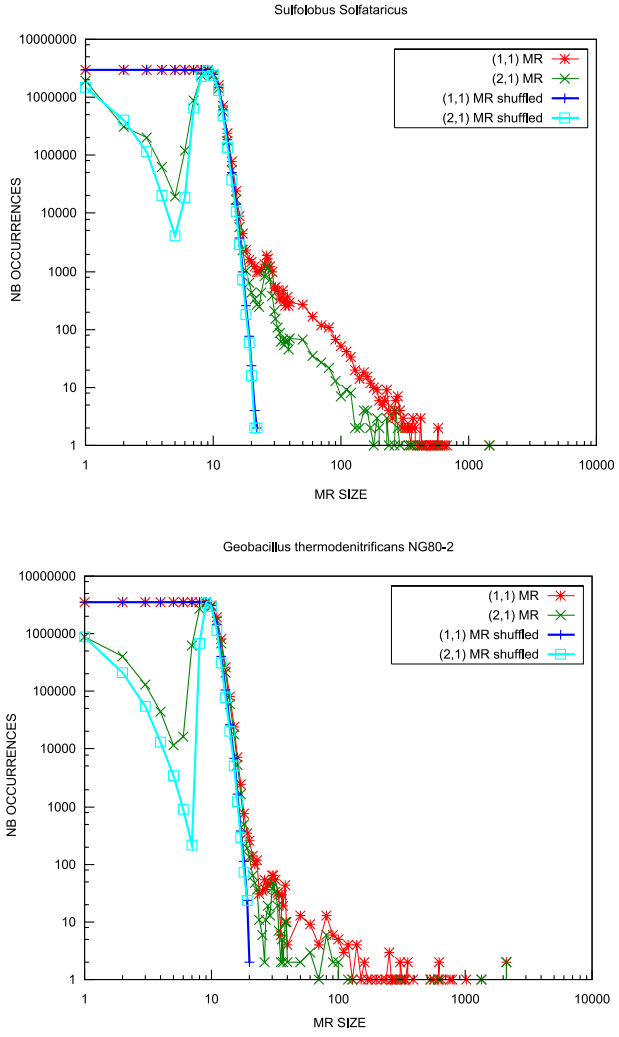

Figure 9: Study of archaeon NC_002754 (*Sulfolobus solfataricus*) and bacteria NC_009328 (*Geobacillus thermodenitrificans* NG80-2). Number of occurrences of (1,1)-maximal repeats and (2,1)-maximal repeats for normal and shuffled genomes. All scales are logarithmic.

Experiments on (1,1) and (2,1)-maximal repeats and SNP

The algorithm computing (1,1) and (2,1)-maximal repeats has been tested on various genomes. Results are illustrated on figure 9 for an archeon and a bacteria. Shuffled genomes have no maximal repeats of size greater than 22. The size of maximal repeats may be far greater for real genomes (maximal observed size is 1,000), but introducing a left context of two letters instead of a single one leads to a two to four fold decrease of this number of occurrences. Enlarging contexts seems also to introduce an unexpected artefactual decrease in the range of small words (less than 10, with a minimum at 6), this bias being observed on both shuffled and real sequences. We have no explanation so far of this fact.

Interesting positions of single nucleotide mutations have also been searched with the following protocol: at each position of mutation p such that at least one maximal repeat v that is not a (2,1)-maximal repeat has an occurrence starting at $p+1$, we have kept its *characteristic context*. This one is defined as the pair of words (u, v) such that

- the size of v is maximum;
- the word u is a maximal repeat with an occurrence ending at $p-1$ and of maximal size with respect to all repeats followed at distance 1 by v.

The SNP positions may then be selected on the basis of the size of uv. An interesting extension would be to directly maximize the size of uv on each position, but the issue seems far more complex.

In our experiments, we have set the threshold on the size of uv to 18, a value consistent with our previous choices and that leads to very few SNP on shuffled versions of the genomes.

In figure 10, the value of uv size is displayed along the genome of *Sulfolobus solfataricus* and of *Geobacillus thermodenitrificans*. Clearly, the observed distributions are not random and differ completely from one species to the other. With a set of strains or individuals of a same species, the concept offers a way to look at fine polymorphisms while avoiding the necessity to run heavy whole genome comparisons.

Units of overlapping repeats

As previously described, more general variations in sequences lead to overlapping maximal repeats. However, overlapping repeats may also be observed in tandem repeats—consecutive copies of a same word—and simply by chance if very small words are taken into account. In the definition of meaningful units, one has thus to precise the repeats to be considered. The previous notion of "not $(2,1)$-MR" and more generally "not $(k,1)$-MR" are useful in the sense that they potentially identifies regions with small local variations that need to be abstracted.

Definition 8 (k-units) A *unit* in a sequence S is a largest series of different overlapping maximal repeat that are not $(2,1)$-MR. A *k-unit* in a sequence S is a largest series of overlapping maximal repeat that are not $(k,1)$-MR.

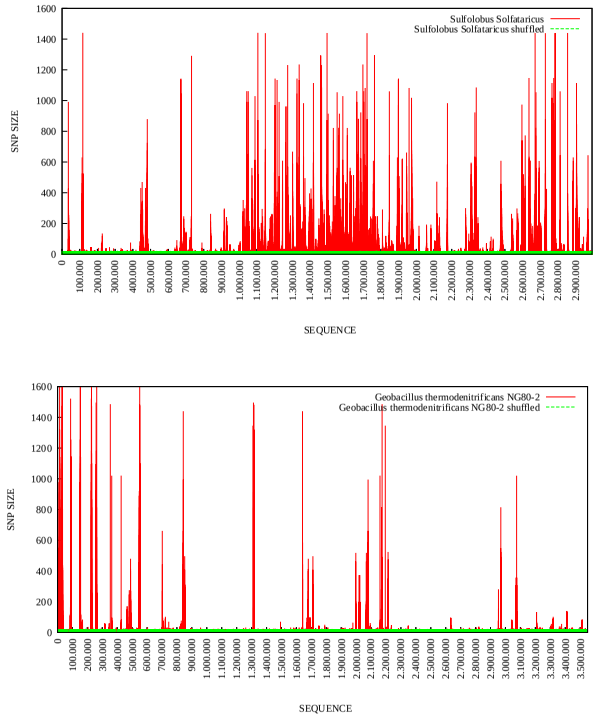

Figure 10: Size of repeats (≥ 18) with a single nucleotide polymorphism (SNP) in archaeon NC_002754 (*Sulfolobus solfataricus*) and in bacteria NC_009328 (*Geobacillus thermodenitrificans* NG80-2). These 2 figures have the same scale and the y-axis has been cut: the biggest repeat with a SNP for *G. thermodenitrificans* NG80-2 is 2338 nucleotides long and thus not visible on the figure.

Example 4 Let S be the following sequence:

$$CGTAGTCACACATGGAGTAACATAGA$$

Some maximal repeats are words TAG, AGT, $CACA$, CA, GA and AG in S. Words TAG, AGT, ACA, GA and C are maximal repeats in S that are not (2,1)-MR. Word $CACACA$ is a tandem repeat covered by 2 occurrences of the same repeat $CACA$ and rejected as a unit. Words $TAGT$, $GAGT$ and $TAGA$, which are slight variations of $TAGT$, are units of S since they are made of a composition of overlapping maximal repeats that are not (2,1)-MR.

Computing units and k-units

Computing units is straightforward. It is sufficient to use previously described MR and (2,1)-MR detection algorithm. Then, reading sequence from left to right, one just has to select the largest set of overlapping MR that does not house (2,1)-MR.

The k-unit detection algorithm is likewise straightforward once disposing from a (k,1)-MR occurrence list. Thus this computation, highly depends on the existence of an efficient (k,1)-MR detection algorithm that remains an open question.

Experiments on units

We have computed units on various genomes and their shuffled version. For small repeats, the number of units is equivalent in normal and random sequences. For repeats of size greater than 18, no unit exists on random sequences and figure 11 displays units in the genome sequence of *Sulfolobus solfataricus*. Numerous units are found this way, corresponding to the recovery of simple copies with natural variations gained during evolution. As expected, the curve decreases rapidly with respect to the size of formed units since for large units there are generally several point of mutations.

4 Discussion

On the definition of locality: The locality of a repeated word has been defined to be the average of scopes of each group of occurrences. We have tried several other possibilities to sum up the contributions of each group, including the sum, the minimum and the maximum.
Overall, the average seems to provide more natural clustering than other measures. First, it is a local measure. The number of modes (clusters) is a secondary parameter that is usually unknown or not relevant and choosing a sum would make it difficult to compare repeats with different numbers of clusters. Second, it is a robust measure with respect to the noise induced by words occurring by chance or by missing words due to degenerative mutations. Maximum and minimum would be too sensitive to these phenomena. Finally, note that the search of optimal groups themselves are based on a cumulative index, not on the average value and this avoids to split the set of occurrences in more clusters than

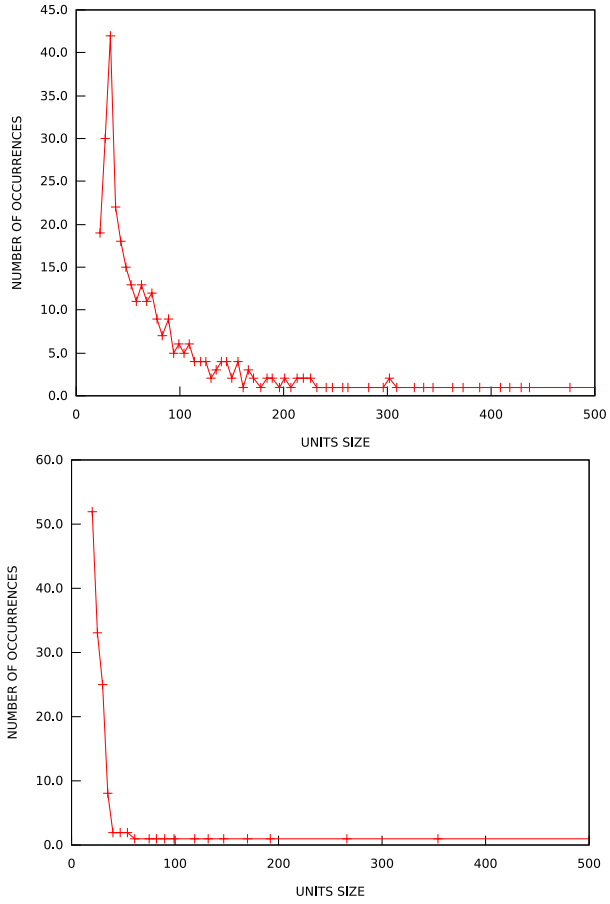

Figure 11: Number of occurrences of units in archaeon NC_002754 (*Sulfolobus solfataricus*, repeats of size ≥ 18) and bacteria NC_009328 (*Geobacillus thermodenitrificans* NG80-2, repeats of size greater than ≥ 15).

necessary. For instance, taking two occurrences of a run of nucleotides (say $AAAAATAAAAA$), the best partition will include the whole run ($AAAAA$) whereas a decision based on the final average index would tend to favor splitting the run in smaller groups.

Basically, a *repeat* as considered in biology refers to an entity made of an *ordered list of common conserved words* occurring at constrained distances. Our proposition is extending the usual setting on repeats in a sequence by constraining distances globally on a set of occurrences rather than locally between two positions. Studying ways to constrain distances opens a new playing field for algorithmics on words in the continuation of works on structured motifs, where the key concept is *flexibility* instead of *approximation*. In approximated pattern matching, the accent is on extending an expression (word or language) by a general error model that is intended to take into account possible variations or errors around the expression to be matched; In flexible matching, the accent is rather on fragmenting the expression in small pieces that have to occur exactly and on its extension by fixing constraints on the possible distances inside and between the occurrences of the expression. Most efficient current algorithms use this notion of exact seeds that form the skeleton of a copied element. Our definitions may help in the search of such seeds. Fundamentally, flexibility expresses the fact that in absence of a relevant model of transformation, variations between two copies can be traced back from the observation of the conjunction of a common prefix and suffix skeleton at a variable distance in each copy. We keep on working on this notion of flexibility to take into account large variations that may occur between occurrences of repeats like transposons.

Number of repeats versus number of occurrences: The number of distinct words (factors) in a sequence and thus the number of repeated factors in a sequence may be quadratic with respect to its length (De Bruijn strings are an example of words with a maximum of factors). In contrast, the maximal number of Maximal Repeats (MR) remains (sub)linear, as it is clearly shown by the fact that a suffix tree of a string of length n has at most $n-1$ internal nodes.

It says however nothing on the number of occurrences of MR although it is of practical importance while looking at their distribution in a sequence. It appears this number to be quadratic itself with respect to n. For instance, the sequence CA^nGA^nT admits exactly n maximal repeats, namely A^k, for $k = 1 \dots n$, and the number of occurrences of all these maximal repeats is $\sum_{i=1}^{n} 2i = n(n+1)$. In fact, since there exists a number of very short maximal repeats, these generate a lot of spurious occurrences of repeats. Supermaximal repeats solve this problem by selecting maximal repeats that are maximal with respect to inclusion. Since there exists at most one supermaximal repeat starting at each position of a sequence, the number of occurrences is bounded by n. In practice, supermaximal repeats exhibit a very low number of occurrences because they generally cover a small set of positions in the sequence.

For instance, the sequence $ACAAGA^k$ has only one supermaximal repeat A^{k-1} with two occurrences (positions 6 and 7), for any value of k. This makes supermaximal repeats unsuitable for the description of genomic sequences. We have proposed two ways to improve the selection of interesting repeats based on their set of occurrences, the notion of largest maximal repeat (LMR) and the notion of locality, and the question naturally arise to study their theoretical impact on the number of occurrences. For instance, on our previous example $ACAAGA^k$,

there are three LMR totalizing $2k+6$ occurrences: A with $k+3$ occurrences, AA with $k+1$ occurrences and A^{k-1} with two occurrences.
Experimentally, we have not been able to exhibit a number of occurrences of LMR in $O(n^2)$. The generality of this conjecture remains to be proved and we leave it as an open problem. However, it is possible to restrict further the definition of LMR in order to guarantee a linear number of occurrences, while maintaining a covering of all repeats. Let us consider pairs (r, O), where r is a repeat in the considered sequence S and where O is the set of occurrences of r in S minus those strictly included in an occurrence of another repeat. The words r with a non empty associated set in such pairs correspond to LMR. The associated occurrences O form a subset of the set of positions and is thus linear, by keeping the nice property of SMR that at most one LMR is starting at each position of a sequence. In our previous example $ACAAGA^k$, the LMR pairs are $(A, \{1\})$, $(AA, \{3\})$ and $(A^{k-1}, \{4, 5\})$.
The locality parameter is another way to filter occurrences while not imposing an arbitrary lower bound on the size of repeats contrary to the common practice. The number of occurrences of each repeat is bounded by definition to be less than the scope minus the size of the repeat times the number of modes.

Possible extensions of the introduced concepts: The definition of scope, locality and largest maximal repeats are not restricted to words and could be extended to patterns since it is only considering positions. Applying these definitions for instance to motifs representing transcription factor sites could help determining interesting regulation sequence in a genome. The concepts could be also applied to sets of sequences and this may be useful in comparative genomics, in the identification of conserved or imported subsequences.

5 Conclusion

The paper has introduced several new important concepts for the analysis of biological sequences, namely μ_locality, largest maximal repeats, δ_neighbors,(k, k')-maximal repeats and units. In most cases, algorithms are given, all using a very generic approach of depth-first search in a suffix tree. The search for (k, k')-maximal repeats has been treated only for $k = 1$ and $k' < 3$, an important practical case.

We have illustrated all introduced concepts on genomic analysis. In fact, all concepts have emerged from real biological observations. The main idea in the chosen developments were to allow the selection of words and their occurrences on the basis of very few parameters and in a way that does not depend on their size. Locality and δ_neighboring allow filtering words on the basis of their distribution in the sequence and need essentially one or two parameters. Maximality has been extended by two parameters allowing a better contextual filtering of words.This way, we have shown how one-letter variations may be taken into account in maximal repeats. We are currently working on a further needed concept of flexibility to take into account larger variations due to inserted or deleted words.

The study is by no means exhaustive on the possible types of constraints that would be meaningful for words selection in the genomic context. Overall,

the main contribution of this study might be to show that genomics remains an endless source of new problems in stringology.

Acknowledgement

The necessary environment for all computations has been provided by the bioinformatics platform from Ouest-Genopole (http://genouest.org).

Funding

This work is supported by a grant from the French *Agence Nationale de la Recherche* (*Modulome* project).

Supplementary material

Procedure **Attributes** recursively modifies an attribute of all nodes encountered during a depth first traversal of a tree. During this traversal, data are synthesized from children and joined into a variable called *Synth* (line 6). Once all children are recursively performed, the *Node*'s attribute is computed with procedure *Attr* with respect to data stored in *Synth* (line 8). The *isLeaf* function takes a node as argument and returns true if and only if this node is a leaf. Depending whether *Node* is considered as terminal or not two data initialization functions are used: initLeaf for leaves (line 2) and initNode for other nodes (line 4).

Algorithm 1 Computes an attribute for each node of a subtree at a given node of a tree, using a depth first traversal of the subtree.

Attributes(*Node*, initNode(), update(), initLeaf(), Attr())

```
1: if isLeaf(Node) then
2:    initLeaf(Node, Synth);
3: else
4:    initNode(Synth);
5:    for each Child of Node do
6:       update( Synth,
                 Attributes(Child, initNode, update, initLeaf, Attr ) );
7:    end for
8:    Attr(Node, Synth);
9: end if
10: Return( Synth);
```

Algorithm 2 Computes *min* and *max* values of positions of a given word

updateMinMax(*Synth*, *SynthChild*)

```
1: Synth.min = min (Synth.min, SynthChild.min);
2: Synth.max = max (Synth.max, SynthChild.max);
```

Algorithm 3 Returns the $\mu\_locality$ of a sorted list of occurrences $Synth$.

muLocality($Synth$)

```
1: { Let n be the size of Synth and μ the maximal number of modes}
2: for j:= 1 to n do
3:    opt[1, j] ← 0;
4: end for
5: for k:= 1 to μ  do
6:    opt[k, 2] ← 0; opt[k, 3] ← 0;
7:    card[k, 2] ← 1; card[k, 3] ← 1;
8: end for
9: for k:= 1 to μ − 1  do
10:   for j:= 2 to n − 2  do
11:      if opt[k + 1, j + 1]>(opt[k, j]+ Synth[j + 1]-Synth[j]) then
12:         opt[k + 1, j + 2] ← opt[k + 1, j + 1];
13:         card[k + 1, j + 2] ← card[k + 1, j + 1];
14:      else if opt[k + 1, j + 1]<(opt[k, j]+ Synth[j + 1]-Synth[j]) then
15:         opt[k + 1, j + 2] ← opt[k, j]+ Synth[j + 1]-Synth[j];
16:         card[k + 1, j + 2] ← card[k, j] +1;
17:      else
18:         opt[k + 1, j + 2] ← opt[k + 1, j + 1];
19:         card[k + 1, j + 2] ← min(card[k + 1, j + 1],card[k, j] +1);
20:      end if
21:   end for
22: end for
23: Return(Synth[n]- Synth[1] - opt[μ, n])/card[μ, n];
```

Algorithm 4 Returns the $asymptotic\_locality$ of a sorted list of occurrences $Synth$.

asymptoticLocality($Synth$)

```
1: { Let n be the size of Synth}
2: opt[2] ← 0; opt[3] ← 0; opt[4] ← Synth[3]-Synth[2];
3: card[2]← 1; card[3] ← 1; card[4] ← 2;
4: for j:= 2 to n − 3  do
5:    if (opt[j + 1]+ Synth[j + 2]-Synth[j + 1])>
      (opt[j]+ Synth[j + 1]-Synth[j]) then
6:       opt[j + 3] ← (opt[j + 1]+ Synth[j + 2]-Synth[j + 1]);
7:       card[j + 3] ← card[j + 1]+1;
8:    else if (opt[j + 1]+ Synth[j + 2]-Synth[j + 1])<
      (opt[j]+ Synth[j + 1]-Synth[j]) then
9:       opt[j + 3] ← (opt[j]+ Synth[j + 1]-Synth[j]);
10:      card[j + 3] ← card[j] +1;
11:   else
12:      opt[j + 3] ← (opt[j + 1]+ Synth[j + 2]-Synth[j + 1]);
13:      card[j + 3] ← min ( card[j + 1]+1, card[j] +1);
14:   end if
15: end for
16: Return (Synth[n]- Synth[1] - opt[n])/ card[n];
```

Words that are δ-neighbors with support τ of L may be computed with function call

$$Attributes(root, zero, neighbors, classOcc, is\_\delta neighbor),$$

where

- *Synth* contains a boolean, two numbers and two association lists that are described in the supplementary material before algorithm 5;
- *zero* sets all values of Synth to 0, $\emptyset$ or $false$, depending on their type;
- procedure *neighbors*, given a sorted list of occurrences L, checks the distance to L of occurrences of the word that are prefixes of the occurrences of the Child. It is described in algorithm 5;
- procedure *classOcc*, given a sorted list of occurrences L, checks the distance to L of the leaf position. It is described in algorithm 6;
- $is\_\delta neighbor$ returns true if and only if *overlap* is false and

$$\frac{Synth.ok + Synth.okR}{Synth.ok + Synth.ko + Synth.l + Synth.r} \geq \tau,$$

  where $Synth.l$ and $Synth.r$ are respectively the total number of occurrences in $Synth.mapL$ and $Synth.mapR$.

The two algorithms algorithm5 and algorithm6) use a structure *Synth* that contains a boolean, two numbers and two association lists:

- $Synth.overlap$ is a boolean indicating that an occurrence of the word overlaps an occurrence of L and thus it cannot be a δ_neighbor;
- $Synth.ok$ is the number of δ_neighbor occurrences of the word (at maximal distance δ from an occurrence of L and not overlapping any occurrence of L). In order to inherit this value in the suffix tree from children to their parent, the algorithm requires moreover that occurrences are counted only if prefixes of the word are also at the right distance from L;
- $Synth.okR$ is the number of δ_neighbor occurrences of the word, at maximal distance δ from an occurrence of L and not overlapping any occurrence of L. Contrary to the previous case ($Synth.ok$), at least one of the prefixes of the word—its first letter—is not at the right distance from L;
- $Synth.ko$ is the number of occurrences of the word that are not neighbors of L, at distance greater than δ from occurrences of L. In order to inherit this value in the suffix tree from children to their parent, the algorithm requires moreover that occurrences are counted only if prefixes of the word are also at distance greater than δ from occurrences of L;
- $Synth.mapL$ is an association list managing the number of overlapping occurrences to the left of an element of L and in the right neighborhood of another element of L. Each element of this list pairs a distance with the number of occurrences of words to the left of an element of L and at the given distance from it. This distance must be smaller than the length

of the word since it overlaps with L. Updating $Synth.mapL$ in the suffix tree from children to their parent consists in checking the elements that keep a distance value smaller than the length of the word;

- $Synth.mapR$ is an association list managing the number of overlapping occurrences to the left of an element of L and not in the right neighborhood of any element of L. Each element of this list pairs a distance with the number of occurrences of words to the right of an element of L and at the given distance from it. This distance must be smaller than the length of the word since it overlaps with L. Updating $Synth.mapR$ in the suffix tree from children to their parent consists in checking the elements that keep a distance value smaller than the length of the word and not greater than the distance δ.

Algorithm 5 Given a sorted list of occurrences L, checks the distance to L of occurrences of the word that are prefixes of the occurrences of the Child.

$neighbors(Synth, SynthChild)$

1: { Let l be the length of the current node's word, δ be the maximal allowed distance to L}
2: **if** $SynthChild.overlap$ or $Synth.overlap$ **then**
3: $Synth.overlap \leftarrow true$;
4: **else**
5: $Synth.ok \leftarrow Synth.ok + SynthChild.ok$;
6: $Synth.ko \leftarrow Synth.ko + SynthChild.ko$;
7: **for** $(Distance, Nbocc) \in SynthChild.mapL$ **do**
8: **if** $Distance \geq l$ **then**
9: $Synth.ok \leftarrow Synth.ok + 1$;
10: **else**
11: $Synth.mapL \leftarrow Synth.mapL \cup (Distance, Nbocc)$;
12: **end if**
13: **end for**
14: **for** $(Distance, Nbocc) \in SynthChild.mapR$ **do**
15: **if** $Distance > l + \delta$ **then**
16: $Synth.ko \leftarrow Synth.ko + 1$;
17: **else**
18: $Synth.mapR \leftarrow Synth.mapR \cup (Distance, Nbocc)$;
19: **if** $Distance \geq l$ **then**
20: $Synth.okR \leftarrow Synth.okR + 1$;
21: **end if**
22: **end if**
23: **end for**
24: **end if**

Proof 4 *of the correctness of computation of δ-neighbors:*
The property of neighboring to be checked contains two conditions: one on the maximal distance δ to some occurrence of L; the other on the non overlapping with all occurrences of L. $L[0]$ start position is set to $-n$ and $L[m+1]$ start position is set to $2n$. Thus, the test in line 1 of Algorithm 6 will always return

Algorithm 6 Given a sorted list of occurrences L, checks the distance to L of the occurrence in the leaf at position p, with a parent w of length l.

classOcc(*Synth*)

1: k denotes the minimal index in $L[1, m+1]$ such that $L.start[k] > p$;
2: **if** $p - L.end[k-1] < 0$ **then**
3: $Synth.overlap \leftarrow true$;
4: **else**
5: **if** $p - L.end[k-1] \leq \delta$ **then**
6: **if** $L.start[k] \geq p + l$ **then**
7: $Synth.ok \leftarrow 1$;
8: **if** $L.start[k] > p + l$, the word at $[p, min(L.start[k]-1, n)]$ is a δ_neighbor.
9: **else**
10: $Synth.mapL \leftarrow Synth.mapL \cup (L.start[k] - p, 1)$;
11: **end if**
12: **else**
13: **if** $L.start[k] \geq n$ **then**
14: $Synth.ko \leftarrow 1$;
15: **else**
16: **if** $L.start[k] > p + l$ **then**
17: the word at $[p, min(L.start[k]-1, n)]$ is a δ_neighbor;
18: **end if**
19: $Synth.mapR \leftarrow Synth.mapR \cup (L.start[k] - p, 1)$;
20: **end if**
21: **end if**
22: **end if**

Algorithm 7 updates the set of letters preceding the pointed word in the suffix tree

cumul(Synth, SynthChild)

1: Synth.21MR $\leftarrow$ False;
2: **for** $a \in \Sigma$ **do**
3: **if** SynthChild[a]$\neq \perp$ **then**
4: **while** $\neg$(Synth.21MR) **and** $(a' \in \Sigma - \{a\})$ **do**
5: **if** (Synth[a']$\neq \perp$) **and** (Synth[a']$\neq$ SynthChild[a]) **then**
6: Synth.21MR$\leftarrow$ True;
7: **end if**
8: **end while**
9: **end if**
10: **end for**
11: **for** $a \in \Sigma$ **do**
12: **if** SynthChild[a]$\neq \perp$ **then**
13: **if** (Synth[a]=SynthChild[a]) **or** (Synth[a]=$\perp$) **then**
14: Synth[a] $\leftarrow$ SynthChild[a];
15: **else**
16: Synth[a] $\leftarrow \top$;
17: **end if**
18: **end if**
19: **end for**

a value $k \in [1, m+1]$ such that $L.start[k-1] \leq p < L.start[k]$.
Let us consider a word w of size m starting at position p, that is, covering positions $[p, p+m-1]$ (for leaves, m will take all values from the size of the parent + 1 to n). The algorithm is based on the identification of six possible cases for occurrences of w. An occurrence of w may be

1. *Overlapping with $L[k-1]$ (line 3 of Algorithm 6 and line 3 of Algorithm 5). This occurrence of w and any of its prefixes do not respect condition 2 of δ_neighbors and Synth.overlap simply propagates this fact.*

2. *At a distance from $L[k-1]$ that is between 0 and δ and non overlapping $L[k]$ (line 7 of Algorithm 6 and lines 5 and 9 of Algorithm 5). This occurrence of w and any of its prefixes do respect both conditions of δ_neighbors and are counted in Synth.ok.*

3. *At a distance from $L[k-1]$ that is between 0 and δ and overlapping $L[k]$ (line 10 of Algorithm 6). This occurrence of w does not respect condition 2 of δ_neighbors but it may become true for smaller prefixes of w. The role of Synth.mapL is to keep track of this possibility. The evolution of condition 2 is checked in line 8 of Algorithm 5.*

4. *At a distance from $L[k-1]$ and $L[k]$ that is greater than δ (line 14 of Algorithm 6 and line 6 and 17 of Algorithm 5). This occurrence of w and any of its prefixes do not respect condition 1 of δ_neighbors and are counted in Synth.ko. Note that if it is a leaf, the case $L[k] > \delta$ occurs only if $k = m+1$.*

5. *At a distance from $L[k-1]$ that is greater than δ and overlapping $L[k]$ (line 19 of Algorithm 6 and line 18 of Algorithm 5). This occurrence of w does not respect condition 2 of δ_neighbors but it may become true for smaller prefixes of w. The role of Synth.mapR is to keep track of this possibility. The evolution of condition 2 is checked in line 19 of Algorithm 5.*

6. *At a distance from $L[k-1]$ that is greater than δ and at a distance from $L[k]$ that is less than δ and not overlapping $L[k]$ (line 17, 19 of Algorithm 6 and line 20 of Algorithm 5). This occurrence of w fulfills all conditions of δ_neighbors and is counted in Synth.okR. Prefixes of w may no more verify condition 1 on maximal distance. Synth.mapR keeps track of this possibility and the condition is checked in line 15 of Algorithm 5.*

Proof 5 *of the correctness of computation of (2,1)-Maximal Repeats:*
Let us fix a repeat w. We assume that the alphabet Σ is ordered. The array Synth tabulates a function describing the left contexts of w:

$$Synth(w)[a] = \{b \in \Sigma,\ \exists c \text{ s.t. } abwc \text{ is a subword of } S\}.$$

Let us fix a right context c. We split up $Synth(w)[a]$ with respect to c:

$$ContextChild(w,c)[a] = \{b \in \Sigma,\ abwc \text{ is a subword of } S\}.$$

$$Cumul(w,c)[a] = \{b \in \Sigma,\ \exists c' < c \text{ s.t. } abwc' \text{ is a subword of } S\}.$$

By definition, w is a (2,1)-Maximal Repeat if there exists $a \neq a'$, $b \neq b'$, $c \neq c'$ such that $abwc$ and $a'b'wc'$ are both subwords of S. In other words, w

is a (2,1)-MR if and only if there exist a, b, c and $a' \neq a$, $b' \neq b$ such that $b \in ContextChild(w,c)[a]$ and $b' \in Cumul(w,c)[a']$.

In order to check this property, thanks to the suffix tree, we reformulate it as follows: w is a (2,1)-MR if and only if there exist a, b, c and $a' \neq a$ such that $b \in ContextChild(w,c)[a]$ and either $Cumul(w,c)[a']$ contains at least two elements—then one of them is different from b and generates a context discrimant from (ab,c)—or it is reduced to a single point different from b.

To build the sets $Synth(w)[a]$, we use the following recursive formulas:

- *for every a, c, we have $Cumul(w,c)[a] = \cup_{c'<c} ContextChild(w,c')[a]$;*
- *for every a, we have $Synth(w)[a] = \cup_{c \in Sigma} ContextChild(w,c)[a]$;*
- *if w corresponds to a leaf of the suffix tree, then w is a suffix of the sequence S so that the sets $ContextChild(w,c)$ cannot be defined. However, w corresponds to a unique position pos in the sequence. If $a = S[pos-2]$ and $b = S[pos-1]$, set $Context(w)[a] = \{b\}$ and $Context(w)[a'] = \emptyset$ for all $a' \neq a$;*
- *if w is not a leaf of the suffix tree, it has several children in the suffix tree. For every child z , $z = wcy$, $Contextchild(w,c)$ is set to $Synth(z)$ and we use relation (2) to compute $Synth(w)$.*

This allows to compute recursively from the leaf to the top of the suffix tree the arrays $Synth(w)$ as a synthesized attribute. For each node, we compute the set $Cumul(w,c)$ from the definition of its children and simultaneously check that it is a 2-MR. Memory is further bounded by computing instead of the set of contexts its projection to $\{\top, \bot\} \cup \Sigma$ where $\top$ stands for any set with cardinality at least 2 and $\bot$ corresponds to the empty set.

References

[1] Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, 2004.

[2] Abouelhoda, Mohamed Ibrahim and Kurtz, Stefan and Ohlebusch, Enno. Enhanced suffix arrays and applications. In Aluru, S., editor, *Handbook on Computational Molecular Biology*, pages 7–1–7–27. Chapman and Hall/CRC, 2006.

[3] A. Apostolico. The myriad virtues of suffix trees. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Words*, volume 12 of *NATO Advanced Science Institutes, Series F*, pages 85–96. Springer-Verlag, Berlin, 1985.

[4] Valentina Boeva, Mireille Regnier, Dmitri Papatsenko, and Vsevolod Makeev. Short fuzzy tandem repeats in genomic sequences, identification, and possible role in regulation of gene expression. *Bioinformatics*, 22(6):676–684, 2006.

[5] G.S. Brodal, R.B. Lyngs, C.N.S. Pedersen, and J. Stoye. Finding maximal pairs with bounded gap. In *CPM*, pages 134–149, 1999.

[6] M. Crochemore, C. S. Iliopoulos, M. Mohamed, and Marie-France Sagot. Longest repeats with a block of don't cares. In *LATIN*, pages 271–278, 2004.

[7] Robert Giegerich and Stefan Kurtz. From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction. *Algorithmica*, 19(3):331–353, 1997.

[8] Dan Gusfield. *Algorithms on strings, trees, and sequences.* Cambridge University Press, 1997.

[9] Costas S. Iliopoulos, James A. M. McHugh, Pierre Peterlongo, Nadia Pisanti, Wojciech Rytter, and Marie-France Sagot. A first approach to finding common motifs with gaps. In *Stringology*, pages 88–97, 2004.

[10] RM. Karp, RE. Miller, and AL. Rosenberg. Rapid identification of repeated patterns in strings, trees and arrays. In *STOC '72: Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 125–136, New York, USA, 1972. ACM Press.

[11] H. H. Kazazian. Mobile elements: drivers of genome evolution. *Science*, 303(5664):1626–1632, 2004.

[12] R. Kolpakov and G. Kucherov. Finding maximal repetitions in a word in linear time. In *Proceedings of the 40th IEEE Annual Symposium on Foundations of Computer Science*, pages 596–604, New York, USA, 1999. IEEE Computer Society Press.

[13] R. Kolpakov and G. Kucherov. On maximal repetitions in words. In G. Ciobanu and Gh. Păun, editors, *Proceedings of the 12th International Symposium on Fundamentals of Computation Theory*, volume 1684, pages 374–385, Iasi, Romania, 1999. Springer-Verlag, Berlin.

[14] R. Kolpakov and G. Kucherov. Finding repeats with fixed gap. In *Proceedings of the 7th International Symposium on String Processing Information Retrieval (SPIRE'00)*, page 162, Washington DC, USA, 2000. IEEE Computer Society.

[15] Stefan Kurtz, J. V. Choudhuri, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich. REPuter: the manifold applications of repeat analysis on a genomic scale. *Nucl. Acids Res.*, 29(22):4633–4642, 2001.

[16] Arnaud Lefebvre, Thierry Lecroq, and Joël Alexandre. An improved algorithm for finding longest repeats with a modified factor oracle. *Journal of Automata, Languages and Combinatorics*, 8(4):647–657, 2003.

[17] L. Marsan and M.-F. Sagot. Extracting structured motifs using a suffix tree – Algorithms and application to consensus identification. In S. Minoru and R. Shamir, editors, *Proceedings of the 4th Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 210–219, Tokyo, Japan, 2000. ACM Press.

[18] F. J. Mojica, C. Díez-Villaseñor, J. García-Martínez, and E. Soria. Intervening sequences of regularly spaced prokaryotic repeats derive from foreign genetic elements. *J. Mol. Evol.*, 60(2):174–182, 2005.

[19] Michele Morgante, Alberto Policriti, Nicola Vitacolonna, and Andrea Zuccolo. Structured motifs search. *Journal of Computational Biology*, 12(8):1065–1082, 2005.

[20] Nadia Pisanti, Alexandra M. Carvalho, Laurent Marsan, and Marie-France Sagot. RISOTTO: Fast Extraction of Motifs with Mismatches. In *Proceedings of the 7th Latin American Symposium on Theoretical Informatics (LATIN'06)*, pages 757–768, Valdivia, Chile, 2006.

[21] C. Pourcel, G. Salvignol, and G. Vergnaud. Crispr elements in yersinia pestis acquire new repeats by preferential uptake of bacteriophage dna, and provide additional tools for evolutionary studies. *Microbiology*, 151(3):653–663, March 2005.

[22] Christine Pourcel, Grégory Salvignol, and Gilles Vergnaud. CRISPR elements in *Yersinia pestis* acquire new repeats by preferential uptake of bacteriophage DNA, and provide additional tools for evolutionary studies. *Microbiology*, 151(3):653–663, 2005.

[23] Mathieu Raffinot. On maximal repeats in strings. *Information Processing Letters*, 80(3):165–169, 2001.

[24] P. Rice, I. Longden, and A. Bleasby. Emboss: the european molecular biology open software suite. *Trends Genet*, 16(6):276–277, June 2000.

[25] Marie-France Sagot and Eugene W. Myers. Identifying satellites and periodic repetitions in biological sequences. *Journal of Computational Biology*, 5(3):539–554, 1998.

[26] Jens Stoye and Dan Gusfield. Simple and flexible detection of contiguous repeats using a suffix tree. *Theor. Comput. Sci.*, 270(1-2):843–856, 2002.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Futurs : Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399