Shuffling Biological Sequences with Motif Constraints

Romain Rivière[1], Dominique Barth[2], Johanne Cohen[3], Alain Denise[1⋆]

[1] LRI, UMR CNRS 8623. Université Paris-Sud XI. Orsay, France.
{Alain.Denise,Romain.Riviere}@lri.fr
[2] PRiSM, UMR CNRS 8144. Université de Versailles-St-Quentin. Versailles, France.
Dominique.Barth@prism.uvsq.fr
[3] LORIA, UMR CNRS 7503. Université de Nancy. Nancy, France.
Johanne.Cohen@loria.fr

Abstract. We study the following problem : given a biological sequence S, a multiset $\mathcal{M}$ of motifs and an integer k, generate uniformly random sequences which contain the given motifs and have exactly the same frequencies of occurrence of k-lets (i.e. factors of length k) of S. This is a particularly difficult problem. We notably prove that the problem of deciding whether a sequence respects given motif constraints is NP-complete. We give a random generation algorithm which turns out to be experimentally efficient.

1 Introduction.

The amount of data coming from sequenced genomes is increasing rapidly. Therefore, there is a need for efficient computer-based methods for extracting biological information from sequences. A widely used method for extracting information involves comparing biological sequences with random sequences, which represent the “background noise”,from which any relevant biological information should stand out. This powerful method has been implemented in several areas of sequence analysis [19, 20]. A key example of this method is the search for exceptional motifs in biological sequences. In this approach, an exceptional motif is a pattern that is over- or underrepresented in a biological sequence compared to the expected number of occurrences of that pattern in random sequences. Any overrepresented or underrepresented motifs may indicate important biological functions. Random sequences are also used for sequence comparison. Pairwise sequence comparison algorithms give a score that measures their similarity. After obtaining the score of an alignment, the main problem is to decide whether the two sequences are homologous (*i.e.* derive from a common ancestral sequence) or not. This is done by comparing the given score with scores from the comparison of the biological sequences with random sequences [4, 13].

For the results to be relevant, the random sequences must model some well-chosen characteristics of biological sequences. The two most widely accepted

⋆ Corresponding author.

random sequence models are based on the number of occurrences of all k-lets, *i.e.* all motifs of given fixed length k, in one or several reference biological sequences [9]. In the first model, the random sequences respect *on average* the given numbers of occurrences. In other words, they obey a stationary Markov chain. In the second model, any random sequence contains *exactly* the same number of occurrences of k-lets as the reference sequence. The first model is well suited to long sequences or large sets of sequences, and is widely used for searching for exceptional motifs. For one or several rather short sequences, the second model is better adapted, notably because Markov chains may not be irreducible in this case. Therefore, this model is used for comparing genes, which are rather short sequences [3]. Random sequences can be studied from both an analytical and algorithmic point of view. Indeed, various analytical methods have been developed for studying the probability distribution of motifs in random sequences to search for exceptional motifs (see *e.g.* [14, 16, 17].) However, in many cases an experimental approach is needed, by generating sets of random sequences. This is particularly necessary for sequence comparison, where there are still very few theoretical results. For the Markovian model, it is straightforward to generate random sequences. However, for the second model (*exact* model), the problem is much more difficult. The first efficient algorithm was developed by Kandel, Matias, Unger and Winkler in 1996 [12].

Recent studies in biological sequence analysis have shown that it is necessary to consider models of random sequences that contain more information than previously thought. For example, Beaudoing *et al.* [6], looked for variants of a polyadenylation signal. They gave a set of sequences where one known motif was strongly overrepresented, and aimed to find other weaker overrepresented motifs. This is a typical case in which some motifs that contain the strong one, or that partially overlap it, can appear overrepresented. These "wrong signals" are called *artefacts*. In this study [6], the known strong signal was the motif `AAUAAA`. The motifs `AAAUAA` and `AUAAAA`, among others, were also overrepresented using a classical model of random sequences. Clearly, these too were artefacts. An *ad hoc* method was then applied to remove these artefacts. However, it has been shown [8] that these artefacts can be removed analytically in general manner by conditioning the occurrence probabilities by the strong signal. In other words, the strong signal is taken into account in the model of random sequences. Van Helden *et al.* [18] classified genes according to the number of occurrences of a set of overrepresented motifs. Although some motifs were related to others, for practical reasons, all motifs were considered independent from each other. The resulting classification could be improved if these dependencies could be taken into account. Therefore, a model of random sequences needs to account for the presence or the overrepresentation of a set of motifs in biological sequences. Unfortunately, at the present, an analytical approach to this problem can only be applied in the simplest cases, in which only one strong motif is to be considered.

Here, we address the problem of generating sequences according to the exact model, but with additional motif constraints. A set of motifs of length greater than k is given, and, as well as the k-lets, the sequences must contain a given number of occurrences of each motif from the set.

In Section 2, we reconsider the algorithm of Kandel *et al.*, which generates sequences without additional constraints. We take this as the starting point of our work and then, in Section 3, we develop our approach. The addition of motif constraints in the model results in difficult problems. We notably prove that the general problem of deciding whether a sequence respects the given motif constraints is NP-complete. In Section 5, we give an algorithm which is experimentally efficient, and present experimental results. For readability, the proofs of our principal results are given in Section 4.

2 The shuffling problem.

Let $S = s_1 s_2 \dots s_n$ be a sequence of length n over an alphabet L, and k an integer such that $2 \leq k \leq n$. A *factor* of S is a word $s_{[p,q]}$ such that $s_{[p,q]} = s_p \dots s_q$ for some $1 \leq p \leq q \leq n$. Consider the number of occurrences in S of all possible k-lets, *i.e.* factors of length k. We call a *shuffled sequence* any sequence which has exactly the same numbers of occurrences of k-lets as S. For example, let $S =$ACTACTCACG and $k = 3$. The sequence S contains two occurrences of the 3-let ACT, and one of each of the following 3-lets: CTA, TAC, CTC, TCA, CAC, ACG. The sequence $S' =$ACTCACTACG is a shuffled sequence of S, because it has exactly the same numbers of occurrences of 3-lets as S. The *shuffling problem* is the problem of generating, uniformly at random (u.a.r.), a sequence among all shuffled sequences. Uniformly at random means that all shuffled sequences must have the same probability of being generated.

We first recall a correspondence between the set of shuffled sequences and the set of Eulerian trails of a particular multigraph, which is somewhat similar to the de Bruijn graph. We call this the *sequence graph of order k* of S.

Definition 1. *The* sequence graph of order k *of S, denoted $Gr(S,k)$, is a directed multigraph $G = (V, E)$, with*

$$V = \bigcup_{i=1}^{n-k+2} \{s_{[i,i+k-2]}\}$$

$$E = \bigcup_{i=1}^{n-k+1} [(s_{[i,i+k-2]}, s_{[i+1,i+k-1]})]$$

Note that V is a set, while E is a multiset (hence the brackets in the definition of E. An example of sequence graph is given in Figure 1.

The nodes of the sequence graph are the factors of size $k-1$ of S, and there are as many arcs between two given nodes $v = s_{[1,k-1]}$ and $v' = s_{[2,k-1]} s_k$ as the number of occurrences of the word $s_{[1,k]}$ in S. It follows that any sequence graph is path-Eulerian, *i.e* it contains at least one path that covers all arcs exactly once

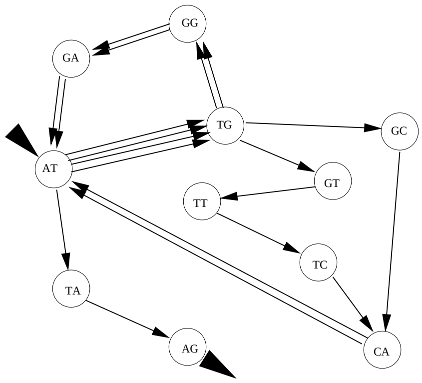

Fig. 1. The Sequence Graph of S=ATGTTCATGCATGGATGGATAG with k=3.

- the sequence of nodes $(s_{[i,i+k-2]})_{i=1}^{i=n-k+2}$. Such a path is called an *Eulerian trail*. In the following, we note v_b (resp. v_e) as the vertex which begins (resp. ends) the Eulerian trail. In some particular cases the sequence graph may be Eulerian (*i.e.* cycle-Eulerian), as well as path-Eulerian. In this case, v_b can be any vertex, and $v_e = v_b$. In all other cases, v_b and v_e are fixed and distinct.

The following definition will help us to formalize the correspondence between shuffled sequences and Eulerian trails.

Definition 2. *The* trace *of a path in a sequence graph is the word produced by concatenation of the $k-1$ letters of the first node and the sequence composed of the last letter of every other node in the path.*

For example, in Figure 1, the word ATGGAGTTC is the trace of the path $(\mathrm{AT}, \mathrm{TG}, \mathrm{GG}, \mathrm{GA}, \mathrm{AT}, \mathrm{TG}, \mathrm{GT}, \mathrm{TT}, \mathrm{TC})$.

Now we can state the claimed correspondence.

Proposition 1. *Any trace corresponds to exactly one shuffled sequence. The number of Eulerian trails which correspond to any given trace does not depend on the trace, and is equal to $\prod_{v \in V} d^+(v)$, where $d^+(v)$ stands for the outdegree of vertex v.*

This correspondence was first noticed by Fitch [9] in 1983, and was the basis of further works by Altschul and Erickson [3] and then Kandel, Matias, Unger and Winkler [12]. Thus, the problem of generating uniformly at random shuffled sequences is reduced to generating uniformly at random Eulerian trails in a (particular) directed multigraph. The next step uses the BEST Theorem [1], which links Eulerian trails and spanning trees of a graph. Here, this theorem can be stated as follows.

Theorem 1 (Aardenne-Ehrenfest and de Bruijn, 1951.). *The number of Eulerian trails in G that begin at v_b and end at v_e is equal to*

$$\mathcal{T}(G)\ (d^+(v_e))! \prod_{v \in V \setminus \{v_e\}} (d^+(v) - 1)!$$

where $\mathcal{T}(G)$ is the number of inbound spanning trees, or arborescences*, whose root is v_e, and $d^+(v)$ stands for the outdegree of vertex v.*

The proof is constructive and leads to a straightforward algorithm for the random generation of Eulerian trails, but only if an arborescence in G can be generated uniformly at random. Starting at the beginning vertex we choose uniformly at random, at each step, an arc from among all the arcs from the current vertex which have not yet been crossed *except the arc which belongs to the arborescence.* This arc can be chosen only if no other arc is available. Then follow the arc to the next vertex, which becomes the new current vertex. The process stops at v_0 when all arcs have been crossed.

The problem of generating uniformly at random Eulerian trails is now reduced to the problem of generating uniformly at random arborescences. G. Kandel *et al.* [12] give an algorithm which is a variant of a very elegant algorithm found independently by Aldous [2] and by Broder [7] for undirected graphs. The algorithm is as follows: if G is only path-Eulerian, then it is first made cycle-Eulerian by adding a virtual arc between v_e and v_b. Then proceed by a free random walk in G, and each time an arc is crossed add it to the arborescence only if it is not the virtual one and no cycle occurs in the resulting arborescence. The expected time complexity of this algorithm is $O(q^2n)$, where q is the number of vertices, *i.e.* the number of distinct k-lets in the sequence S. Recently, Propp and Wilson [15, 21] have developed new algorithms, based on similar principles, which improve the time complexity.

3 Shuffling sequences with motif constraints.

3.1 Preliminaries

In this section, we address the problem of generating shuffled sequences that are subject to additional constraints. SWe consider a reference sequence S of length n on an alphabet L and an integer k such that $2 \leq k \leq n$. Now, let $\mathcal{M} = [M_1, \ldots, M_p]$ be a multiset of words over L with $|M_i| > k\ \forall i \in [1, p]$, such that each M_i is a factor of S, and there are, at most, as many occurrences of M_i in $\mathcal{M}$ as in S. Overlapping occurrences are not taken into account, *i.e.* if the occurrence of two motifs overlap in the sequence, in which case, only one is counted. In the following, we call the words of $\mathcal{M}$ *motifs*.

The problem consists of generating sequences that have exactly the same k-lets count as S, and contain at least as many occurrences of each motif of $\mathcal{M}$ as its number of occurrences in $\mathcal{M}$. verlapping occurrences are again not taken into account. *Acceptable sequences* are any sequence that respects these conditions. As motifs are taken (without overlap) for the reference sequence S, we are guaranteed at least one acceptable sequence.

Our approach consists of two principal steps. These are developed in Sections 3.2 and 3.3. In the first step, we define a new multi-digraph from $Gr(S, k)$ in which each acceptable sequence is the trace of an Eulerian trail. We then generate uniformly at random an Eulerian trail, and verify that the corresponding trace gives rise to an acceptable sequence - this is not always the case. This step involves an NP-complete problem. However, we propose a simple efficient heuristic algorithm for solving this problem (Section 5). The second step aims to ensure the uniformity of the random generation. For this, we need to compute the number of Eulerian trails that correspond to any generated trace. Unlike the original shuffling problem (see Proposition 1), this number strongly depends on the given trace. This counting problem is #P-complete, but we propose a method to solve it in practice.

We present three major definitions involving acceptable sequences.

Definition 3. *A* configuration *of a sequence S according to a multiset of words $\mathcal{M} = [M_1, \ldots, M_p]$ is a p-uplet $(i_1, \ldots, i_p)$ of integers, where i_l is the position of one occurrence of the word M_l in S.*

Definition 4. *Let $C = (i_1, \ldots, i_p)$ and $C' = (i'_1, \ldots, i'_p)$ be two configurations of a sequence S according to the multiset $\mathcal{M} = [M_1, \ldots, M_p]$. For any word w in $\mathcal{M} = [M_1, \ldots, M_p]$, let J_w be the set of integers such that $J_w = \{j : M_j = w\}$. Configurations C and C' are said to be* equivalent *if, and only if, for any word w in $\mathcal{M} = [M_1, \ldots, M_p]$, the two sets $\{i_j : j \in J_m\}$ and $\{i'_j : j \in J_m\}$ are equal.*

Definition 5. *A configuration C of a sequence S according to a multiset of words $\mathcal{M} = [M_1, \ldots, M_p]$ is* perfect *if, and only if, for any i and j, there is no overlap between any two occurrences of M_i and M_j.*

Clearly, a sequence is acceptable if, and only if, it has a perfect configuration over $\mathcal{M}$.

3.2 Generating acceptable sequences

Constrained sequence graphs

Definition 6. *The* sequence cluster *of order k of a word $M_i = m_1 \ldots m_{r_i} \in \mathcal{M}$, denoted $Ch_i(M_i, k)$, is a directed multigraph $C = (CV, CE)$ composed of three nodes:*

$$CV = \{m_{[1,k-1]}, (m_{[2,r_i-1]}, i), m_{[r_i-k+2,r_i]}\}$$

and two arcs:

$$CE = [(m_{[1,k-1]}, (m_{[2,r_i-1]}, i)), ((m_{[2,r_i-1]}, i), m_{[r_i-k+2,r_i]})]$$

The special node $(m_{[2,r_i-1]}, i)$ is called a cluster node.

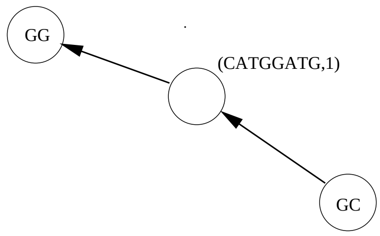

Fig. 2. Sequence Cluster of M_1 =GCATGGATGG, with $k = 3$

An example of sequence cluster is given in Figure 2.

Definition 7. *Let S be an acceptable sequence. Let $G = Gr(S, k) = (V, E)$, the sequence graph associated with S and k. For all $i \in [1, p]$, let $G_i = Gr(M_i, k) = (V_i, E_i)$ and $C_i = Ch_i(M_i, k) = (CV_i, CE_i)$ be the sequence graphs and the sequence clusters associated with each M_i. The* constrained sequence graph G', *denoted $GrC(S, k, \mathcal{M}) = (V', E')$, is defined by $G' = (V', E')$, with*

$$E' = E \cup \bigcup_{i=1}^{p} CE_i - \bigcup_{i=1}^{p} E_i$$

and

$$V' = \{v \in V'' | deg_{G''}(v) \neq 0\}$$

where $G'' = (V'', E')$ with

$$V'' = V \cup \bigcup_{i=1}^{p} CV_i \ .$$

We have replaced the subgraphs representing each M_i in $Gr(S, k)$ by the sequence cluster of M_i. There are as many cluster nodes in $GrC(S, k, \mathcal{M})$ as there are motifs in $\mathcal{M}$. An example of a constrained sequence graph is given in Figure 3.

The notion of a trace of a sequence graph can be extended to the constrained sequence graphs, by making the following change: on crossing a cluster node (w, i), its $|w| - k$ last letters have to be concatenated. As in Section 2, the following simple result shows that there is a direct link between acceptable sequences and Eulerian trails in a constrained sequence graph.

Proposition 2. *The set of acceptable sequences is included in the set of traces of Eulerian trails in $GrC(S, k, \mathcal{M})$.*

Proof. Let S be an acceptable sequence over $\mathcal{M} = [M_1, \ldots, M_p]$, a multiset of words, and $J = (j_1, \ldots, j_p)$ be a perfect configuration of S according to $\mathcal{M}$. Let $(i_1, \ldots, i_p)$ be the positions in S of the occurrences of words pointed to

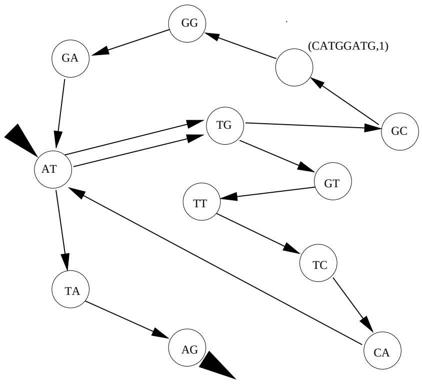

Fig. 3. The constrained sequence graph of ATGTTCATGCATGGATGGATAG with $\mathcal{M}$ = [GCATGGATGG] and $k = 3$.

by the perfect configuration J. Let us consider $T = (s_{[1,k]}, \ldots, s_{[i_1,i_1+k-1]}, \ldots, s_{[i_1+|M_1|-1,i_1+|M_1|+k-2]}, \ldots, s_{[i_p,i_p+k-1]}, \ldots, s_{[i_p+|M_1|-1,i_p+|M_1|+k-2]}, \ldots, s_{[n-k+1,n]})$, an Eulerian trail in $Gr(S,k)$ whose trace is S. Let $C(M_i)$ be the cluster node associated with M_i. Then, $T' = (s_{[1,k]}, \ldots, s_{[i_1,i_1+k-1]}, C(M_1), \ldots, s_{[i_p,i_p+k-1]}, C(M_p), \ldots, s_{[n-k+1,n]})$ is an Eulerian trail in $GrC(S,k,\mathcal{M})$ whose trace is S.

Searching for perfect configurations.
Unfortunately, not all Eulerian trails give rise to an acceptable sequence, because motifs may overlap, as shown in Figure 4. Therefore, once a random sequence S has been generated, we have to verify whether it contains a perfect configuration. We call this problem PCS for "Perfect Configuration Searching", and it is defined as follows.

Instance: An alphabet A, a sequence S over A, a multiset $\mathcal{M} = [M_1, \ldots, M_p]$ of p words.
Question: Does there exist a perfect configuration of S according to $\mathcal{M}$?

At this stage, the sequences that we are dealing with are not general sequences because they result from an Eulerian trail in a sequence graph. Therefore, we need to considerv the problem of searching for a perfect configuration in such sequences. Definition 8 and Proposition 3 will allow us to do this.

Definition 8. *Let k be a positive integer. A configuration C of a sequence S according to a multiset of words $\mathcal{M}$ is* (k)-pseudo-perfect *if, and only if, for any i and j, there is no overlap of as much as k letters between any two occurrences of M_i and M_j.*

This means that all the words pointed to by the configuration overlap by at most $k-1$ letters. We shall omit the parameter k when explicit reference to a

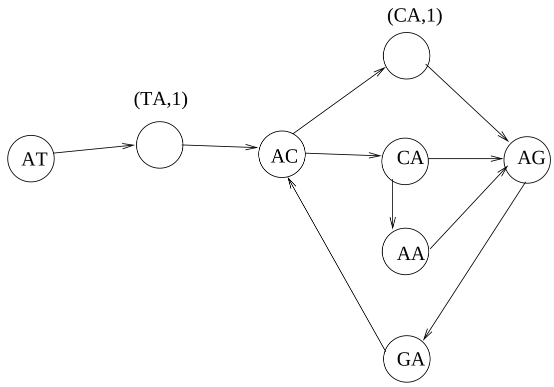

Fig. 4. In this constrained sequence graph with $\mathcal{M}$ =[ATAC,ACAG], the Eulerian trail (AT,TA,AC,CA,AG,GA,AC,CA,AA,AG) gives a sequence ATACAGACAAG, which is not acceptable because the only occurrences of ATAC and ACAG are overlapping.

constrained sequence graph is given. In this case, k is the order of the graph. Now, the following property holds.

Proposition 3. *A sequence S has a k-pseudo-perfect configuration according to $\mathcal{M}$ if, and only if, S is the trace of an Eulerian trail in the constrained sequence graph $GrC(S, k, \mathcal{M})$.*

Proof. Let S be the trace of an Eulerian trail $T = (t_1, \ldots, t_{n-k+1})$ given as a sequence of nodes in $GrC(S, k, \mathcal{M}) = (V', E')$. Some of these nodes, say $t_{i_1}, \ldots, t_{i_p}$, are cluster nodes. Therefore, for any l_1, l_2, there is no arc $(t_{i_{l_1}}, t_{i_{l_2}})$ in E'. Thus, there exists $t_j \in T$ such that $i_{l_1} < j < i_{l_2}$. This implies that occurrences $m_{i_{l_1}}$ and $m_{i_{l_2}}$ overlap in S by at most $|t_j| = k-1$ letters, and S contains a pseudo-perfect configuration over $\mathcal{M} = [M_1, \ldots, M_p]$.

Conversely, if S has a k-pseudo-perfect configuration $(j_1, \ldots, j_p)$ over $\mathcal{M}$, then we can construct the same Eulerian trail T' from an Eulerian trail T in $Gr(S, k)$.

Thus, the actual problem we are addressing, FPCS for "Further Perfect Configuration Searching", is defined as follows.

Instance: An alphabet A, a multiset $\mathcal{M} = [M_1, \ldots, M_p]$, an integer k and S a word over A such that S has a (k)-pseudo-perfect configuration over $\mathcal{M}$.
Question: Does there exist a perfect configuration M over S ?

Unfortunately, we have:

Theorem 2. *Problem FPCS is NP-complete.*

And we deduce:

Corollary 1. *PCS is NP-complete.*

For readability, the proofs of Theorem 2 and Corollary 1 are given in Section 4.

An algorithm for FPCS

Despite having just stated that FPCS is NP-complete, we present here an algorithm that is efficient in realistic cases (see Section 5). First, we define the *overlapping graph* of $\mathcal{M}$ over S.

Definition 9. *The* overlapping graph *of $\mathcal{M}$ over S is the undirected graph $G = (V, E)$ such that every occurrence in S of each word in $\mathcal{M}$ is a distinct node and such that there is an arc between two given nodes if the occurrences they are associated with are overlapping in S.*

An example of an overlapping graph is given in Figure 5. Given an overlapping

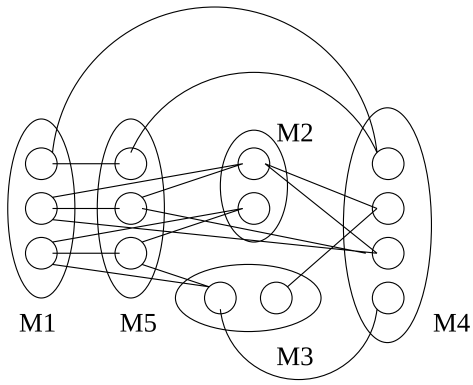

Fig. 5. The overlapping Graph of $\mathcal{M}$=[ATT,TATT,CGAT,TTAT,ATT] over S =ATTATCGATTATATTATCCGACGATTATTC.

graph G, the algorithm is essentially a classical arborescent search. We suppose that the motifs of $M = [M_1, \ldots, M_p]$ and the occurrences $[M_{i,1}, \ldots, M_{i,p_i}]$ of each motif M_i are ordered. The algorithm then proceeds as follows: Take the first occurrence of the first motif and delete all of its neighbours. Then continue in the same manner with the first (remaining) occurrence of the second motif and so on, until either the last motif is taken, or the process stops before reaching all motifs. In the first case, the set of occurrences that were chosen constitutes a perfect configuration. In the second case, we backtrack until we find a suitable sequence of occurrences.

There is also a direct interpretation of a perfect configuration in terms of graph G. If we add edges to make cliques on all the vertex-occurrences of a same motif, then there is a one-to-one correspondence between the set of perfect configurations and the set of maximum independent sets in this new graph.

3.3 Generating sequences uniformly at random.

We now focus on the problem of generating random sequences *uniformly.* For constrained sequence graphs, no property such as Proposition 1 holds. The number of Eulerian trails corresponding to a given trace strongly depends on this trace. Consequently, the generation process is not necessarily uniform. Therefore, we use a classical rejection method to make the generation uniform. When a trace is generated, we either accept it with a probability proportional to the number of its corresponding Eulerian trails, or we reject it and start the process again. Hence, we need to count the number of Eulerian trails corresponding to a given trace.

Proposition 4. *The number of Eulerian trails corresponding to any given trace S is*

$$N(k, S, \mathcal{M}) / \prod_{m \in \mathcal{M}} |\mathcal{M}|_m!$$

where $N(k, S, \mathcal{M})$ is the number of (k)-pseudo-perfect configurations of S according to $\mathcal{M}$, and $|\mathcal{M}|_m$ is the number of occurrences of m in the multiset $\mathcal{M}$.

Proof. In Propositions 2 and 3, we have seen how to map a k-pseudo-perfect configuration to an Eulerian trail in $GrC(S, k, \mathcal{M})$. This gives the numerator. However, if two configurations are equivalent (see Definition 4), they will be mapped to the same Eulerian trail, giving the denominator.

Now, counting the number of Eulerian trails corresponding to a given trace reduces to counting the number of equivalence classes of pseudo-perfect configurations. Our counting algorithm is based on the *pseudo-overlapping graph* of $\mathcal{M}$ over S, similar to the previously defined overlapping graph.

Definition 10. *The* pseudo-overlapping graph *of $\mathcal{M}$ over S is the undirected graph $G = (V, E)$ such that each occurrence in S of every word in $\mathcal{M}$ is a distinct node, and there exists an edge between two given nodes if the occurrences with which they are associated overlap by at least k letters.*

If we consider the pseudo-overlapping graph in which all the nodes corresponding to occurrences of any same word are connected together in a clique, the number of maximal independent sets (MIS) in this graph is obviously equal to the number of equivalence classes of perfect configurations in the related sequence. The problem of counting MISs is known to be polynomial in intersection graphs (including interval graphs) [5]. Although each pseudo-overlapping graph is clearly an interval graph, the graphs we consider here are not even perfect graphs (the problem of determining the cardinal of an MIS is polynomial for perfect graphs but NP-complete for general graphs, see [11, 10] and refs.). Unfortunately, we have

Theorem 3. *The problem of counting the equivalence classes of perfect configurations of S according to $\mathcal{M}$ is #P-complete.*

The proof of this Theorem is given in Section 4.

3.4 The random generation algorithm.

We are now able to state the complete algorithm for generating constrained sequences uniformly at random.

Algorithm 1 *Random generation*
Input: a sequence S, an integer k, a multiset $\mathcal{M}$
Output: a sequence T

(i) Produce the constrained sequence graph $G = GrC(S, k, \mathcal{M})$.
(ii) Uniformly generate a random Eulerian trail in G, and take its trace T.
(iii) If T has no perfect configuration then goto (ii).
(iv) Compute the number N of Eulerian trails corresponding to this particular trace T.
(v) Return T with probability $1/N$, or goto (ii).

If we could compute a lower bound m of the minimum over the traces of the number of Eulerian trails associated with any traces, we could replace the rejection probability in (v) by m/N. However, in general, it is very difficult to compute this lower bound.

Proposition 5. *Step (iv) of Algorithm 1 is called at most R times on average, where R is the average number of Eulerian trails per trace.*

Proof. Consider the square $[0, 1]^2$. For any given trace of an Eulerian trail, consider an interval of $[0, 1]$ whose length is the probability of choosing this particular trace. Place those intervals one after each other in any given order. Then, above each interval, construct a rectangle whose height is the probability of keeping this trace according to Algorithm 1. The sum of the areas of these rectangles equals the number of distinct traces divided by the number of distinct Eulerian trails. It is easy to verify that this number is, in fact, the inverse of the average number of Eulerian trails associated to a trace. This is the expected number of steps needed to hit one of these rectangles, and to stop the algorithm.

4 Proofs of Theorem 2, Corollary 1 and Theorem 3

Clearly, PCS is a special case of FPCS. For the sake of clarity, we first prove the NP-completeness of PCS (Corollary 1) and then generalise it for FPCS(Theorem 2).

Corollary 1 PCS is NP-complete.

Proof. Clearly, we can verify in polynomial time whether or not a given configuration is perfect. So PCS is in NP. We now reduce PCS to 3 Dimensional Matching (3DM) (see example 1). The 3DM problem [10] is defined as follows:
Instance: A set $\mathcal{C} \subseteq X \times Y \times Z$ where X, Y, Z are disjoint sets having the same number q of elements.

Question: Does $\mathcal{C}$ contain a matching, that is, a subset $\mathcal{C}' \subseteq \mathcal{C}$ such that $|\mathcal{C}'| = q$ and no two elements of $\mathcal{C}'$ agree in any coordinate? Let us consider an instance of 3DM, that is 3 sets X, Y, Z of the same cardinality q and $\mathcal{C} \subset X \times Y \times Z$. For any $r \in X \cup Y \cup Z$, we define $f_{\mathcal{C}}(r)$ as the number of occurrences of r in $\mathcal{C}$.

Let $\mathcal{C} = \{c_1, \ldots, c_s\}$ and define $S = w_{c_1} \ldots w_{c_s}$ where, $\forall c = (x, y, z) \in \mathcal{C}$, $w_c = w_x w_y w_z 0$, with $w_x = a0^{x-1}10^{q-x}a$, $w_y = b0^{y-1}10^{q-y}b$, and $w_z = ba0^{z-1}10^{q-z}ba0$.

For any $x \in X$, we define a multiset M_x as follows: it contains

1. one occurrence of the motif $m_x = 0^{x-1}10^{q-x}ab$
2. $f_{\mathcal{C}}(x) - 1$ occurences of the motif $m'_x = a0x^{x-1}10^{q-x}a$

Similary, for any $y \in Y$ (resp. $z \in Z$) we define M_y (resp. M_z) as the multiset containing one occurrence of $m_y = 0^{y-1}10^{q-y}bba$ (resp. $m_z = 0^{z-1}10^{q-z}ba0$) and $f_{\mathcal{C}}(y) - 1$ occurrences of $m'_y = b0y^{y-1}10^{q-y}b$ (resp. $f_{\mathcal{C}}(z) - 1$ occurrences of $m'_z = ba0x^{z-1}10^{q-z}ba$). Finally, we set $\mathcal{M} = \bigcup_{e \in X \cup Y \cup Z} M_e$ where $\bigcup$ denotes the union of multisets.

Obviously, this transformation is polynomial with respect to the instance of 3DM. So, we only need the following two claims to conclude.

Claim 1 If there exists a perfect matching in $\mathcal{C}$, then there exists a perfect configuration of $\mathcal{M}$ over S.

Let $\mathcal{C}'$ be a perfect matching for $\mathcal{C}$. We construct a perfect configuration of S over $\mathcal{M}$ by independently considering the factors $w_c = a0^{x-1}10^{q-x}a\, b0^{y-1}10^{q-y}b\, ba0^{z-1}10^{q-z}ba0$ of S, for all $c \in \mathcal{C}$.

1. Each $c = (x, y, z) \in \mathcal{C}'$ is recovered by the following three motifs of $\mathcal{M}$: $m_x = 0^{x-1}10^{q-x}ab$, $m_y = 0^{y-1}10^{q-y}bba$, and $m_z = 0^{z-1}10^{q-z}ba0$. Each of these motifs occurs only once in M_x, M_y and M_z respectively. Since there is only one occurrence of x, y and z respectively in $\mathcal{C}'$ by definition of a matching, only the motifs in $\{m_x, m_y, m_z : x \in X, y \in Y, z \in Z\}$ of $\mathcal{M}$ have been used to cover all the factors w_c of S for any $c \in \mathcal{C}'$.
2. Each $c = (x, y, z) \notin \mathcal{C}'$ is recovered by the following three motifs of $\mathcal{M}$: $m'_x = a0^{x-1}10^{q-x}a$, $m'_y = b0^{y-1}10^{q-y}b$, and $m'_z = ba0^{z-1}10^{q-z}ba$. Since motif m'_x (resp. m'_y, m'_z) occurs $f_{\mathcal{C}}(x) - 1$ (resp. $f_{\mathcal{C}}(y) - 1$, $f_{\mathcal{C}}(z) - 1$) times in $\mathcal{M}$, all the factors w_c of S for any $c \notin \mathcal{C}'$ are covered (unless the terminal 0 in each of them), and all the motifs m'_x, m'_y and m'_z of $\mathcal{M}$ have been used.

Finally, all motifs of $\mathcal{M}$ have been used, and no two overlap. We have thus defined a perfect configuration of S according to $\mathcal{M}$.

Claim 2 If there exists a perfect configuration of $\mathcal{M}$ over S, then there exists a perfect matching in $\mathcal{C}$.

Let P be a perfect configuration of $\mathcal{M}$ over S. We construct a perfect matching $\mathcal{C}'$ in $\mathcal{C}$. We define $\mathcal{C}'$ as follows: $c = (x, y, z) \in \mathcal{C}'$ if, and only if, in P, w_x of factor $w_c = w_x w_y w_z 0$ of S is partially recovered by the motif $m_x = 0^{x-1}10^{q-x}ab$ of $\mathcal{M}$.

Since $|\{m_x : x \in X\}| = |X| = q$, by construction we get $|\mathcal{C}'| = |\{m_x : x \in X\}| = q$. It remains to prove that $\mathcal{C}'$ is a perfect matching of $\mathcal{C}$. Indeed, let $c =$

$(x, y, z) \in \mathcal{C}'$. In the corresponding factor $w_c = w_x w_y w_z 0$ of S, by definition w_x is recovered by m_x. The $f_{\mathcal{C}}(x) - 1$ remaining occurrences of w_x in S are recovered by the $f_{\mathcal{C}}(x) - 1$ motifs m'_x. Thus, w_c is necessarily recovered by $m_x m_y m_z$, because, by construction, no two motifs m_r and m'_s (for any $r, s \in X \cup Y \cup Z$) can recover a factor w_c without overlapping. As there is exactly one motif m_x (resp. m_y, m_z) per element of X (resp. Y, Z), each element of X (resp. Y, Z) occurs exactly once in $\mathcal{C}'$. Finally, $\mathcal{C}'$ is a perfect matching of $\mathcal{C}$.

This concludes the proof.

Example 1. We consider an instance $\mathcal{I}$ of 3DM such that $X = \{x, x'\}$, $Y = \{y, y'\}$, $Z = \{z, z'\}$, $\mathcal{C} = \{c_1 = (x, y', z), c_2 = (x', y, z'), c_3 = (x, y, z')\}$. The instance $T(\mathcal{I})$ of PCS is defined as follows:

- $w_x = a10a$, $w_{x'} = a01a$,$w_y = b10b$, $w_{y'} = b01b$,$w_z = ba10ba$, $w_{z'} = ba01ba$
- $w_{c_1} = a10ab01bba10ba0$, $w_{c_2} = a01ab10bba01ba0$,$w_{c_3} = a10ab10bba01ba0$
- $\mathcal{M} = [10ab, a10a, 01ab, 10bba, b10b, 01bba, 10ba0, 01ba0, ba01ba]$.
- $S = a10ab01bba10ba0a01ab10bba01ba0a10aba10bba01ba0$.

The instance $\mathcal{I}$ has a matching composed of c_1 and c_2. For the instance $T(\mathcal{I})$ has a perfect configuration of $\mathcal{M}$ over S.

$$S = \underbrace{a\,\overbrace{10ab}\,\overbrace{01bba}\,\overbrace{10ba0}}_{w_{c_1}}\underbrace{a\,\overbrace{01ab}\,\overbrace{10bba}\,\overbrace{01ba0}}_{w_{c_2}}\underbrace{\overbrace{a10a}\,\overbrace{b10b}\,\overbrace{ba01ba}\,0}_{w_{c_3}}$$

Theorem 2 Problem FPCS is NP-complete.

Proof. We only need to prove the result for the particular case where $k = 2$. It is easy to see that Problem FPCS belongs to NP. Moreover, we transform 3DM to FPCS by the transformation given in Corollary 1. Let us consider an instance $\mathcal{I}$ of 3DM, that is, three sets X, Y, Z of the same length q and $\mathcal{C} \subset X \times Y \times Z$. Let $\mathcal{I}'$ be an instance of FPCS obtained by the transformation in Corollary 1. Instance $\mathcal{I}'$ is composed of an alphabet $A = \{0, 1, a, b\}$, a multiset $\mathcal{M}$ of p words, and a word S over A. By the proof of corollary 1, there exists a perfect matching in $\mathcal{C}$ if, and only if, there exists a perfect configuration of $\mathcal{M}$ over S.

It remains to prove that S has a (2)-pseudo-perfect configuration C over $\mathcal{M}$. Let $x \in X$. Recall that $f_{\mathcal{C}}(r)$ is the number of occurrences of r in $\mathcal{C}$. By the transformation from 3DM in corollary 1, there is one motif $0^{x-1}10^{q-x}ab$ and $f_{\mathcal{C}}(x) - 1$ motifs $a0^{x-1}10^{q-x}a$ in $\mathcal{M}$. We now construct a pseudo-perfect configuration C over $\mathcal{M}$. The $f_{\mathcal{C}}(x)$ patterns $a0^{x-1}10^{q-x}ab$ contained in S can be covered by the corresponding $f_{\mathcal{C}}(x)$ motifs in $\mathcal{M}$. We apply the same construction for all elements of Y and Z. Now, for each $c = (x, y, z)$ in $\mathcal{C}$, the word w_c in S is covered by three motifs of $\mathcal{M}$, and, by construction, two consecutive motifs overlap by at most one letter. So, C is a (2)-pseudo-perfect configuration over M.

Theorem 3 The counting problem of the equivalence classes of perfect configurations of S according to $\mathcal{M}$ is #P-complete.

Proof. Problem #PCS belongs to the class #P because there is a polynomial-time algorithm to determine, given an instance x of #PCS and a configuration y of S according to $\mathcal{M}$, if y is a perfect configuration of S according to $\mathcal{M}$. We demonstrate that #PCS is #P-hard, by showing a parsimonious reduction from the #P-complete problem #Perfect Matching defined as follows:
Instance: A bipartite graph G.
Question: How many perfect matchings does G have?

Suppose that we are given an instance I of the #Perfect Matching problem with bipartite graph $G = (V_1 \cup V_2, E)$ such that no two vertices within V_1 (resp. V_2) are adjacent. The reduction can be splitted into two parts.

First, instance I is tranformed into an instance I' of 3DM such that:

- $X = V_1$, $Y = V_2$ and $Z = V_2$.
- $\mathcal{C} = \{(x_1, x_2, x_2) : (x_1, x_2) \in E \wedge x_1 \in V_1 \wedge x_2 \in V_2\}$

Therefore, the number of perfect matchings in G is equal to the number of matchings in $\mathcal{C}$. Indeed, there is a one-to-one correspondence between the set of perfect matchings in G and the matching in $\mathcal{C}$.

Also, the instance I' of 3DM is transformed into an instance I'' of #PCS using the same transformation as in the proof of Corollary 1. The proofs of Claims 1 and 2 show that there exists a one-to-one correspondence between the set of matchings in $\mathcal{C}$ and the set of equivalence classes of perfect configurations of S. So, there exits a one-to-one correspondence between the set of perfect matchings in G and the set of equivalence classes of perfect configurations of S.

Thus, this reduction from #Perfect Matching to #PCS is parsimonious.

5 Experimental results.

We know the theoretical complexity of every routine of our algorithm except for

1. the number of times step (ii) of the algorithm is processed
2. searching if T contains a perfect configuration (step (iii));
3. counting the number of Eulerian trails which correspond to T (step (iv)).

Therefore, we carried out simulations on random data to determine the average complexity of these routines. We aimed to determine what can and cannot be done in terms of the size of the parameters. Routines 2 and 3 involve essentially an arborescent search over an overlapping graph. The first routine requires only one "good" search and then stops. However, in many case for routine 3, a search of the whole search tree is needed. This makes the two algorithms different in terms of what makes them difficult.

We generated random instances of the problem as follows. Sequences of size n were generated according to uniform Bernoulli probabilities over an alphabet of size t. Generally, we took $t = 4$ because we are interested in DNA sequences.

Given the cardinality p of $\mathcal{M}$ and the size s of its motifs, we then generated the multiset $\mathcal{M}$ by choosing p positions in the sequence and taking, for each position, the word of length s beginning at that position. Thus, all motifs had the same length.

We first looked at the number of times step (ii) was processed. We found that for small k, say $k < 10$, almost all the sequences produced contained a perfect configuration. Therefore, the algorithm behaves as if there were no rejection.

For routine 2, a difficult case would occur when n is far larger than 4^s. In this case the motifs would tend to have more than one occurrence in S and therefore, the overlapping graph would have many nodes. The problem is even more difficult if these occurrences overlap, which is the case when the motifs are numerous and large enough. If $k >> 1$, the instance also becomes difficultbecause any trace of an Eulerian trail already contains a (k)-pseudo-perfect configuration. Therefore, to generate difficult cases, we need $n >> 4^s$ and $s > k >> 1$. If we choose $k \geq 10$ and $s \geq 11$, then n should be greater than 10^8. The graph library we used for our implementation did not allow us to investigate this many values efficiently. Therefore, we restricted our simulations to $k = 5$ and found no case when $n < 100000$ and $|\mathcal{M}| < 1000$ in which the computation time of routine 2 was significant. (For bioinformatics purposes, $k = 5$ is a standard value for shuffling DNA sequences.) We are currently working on implementing a graph library that should allow us further investigations.

Routine 3 is the bottleneck of our algorithm. As it enumerates all pseudo-perfect configurations, its complexity strongly depends on the number of nodes of the pseudo-overlapping graph. As for routine 2, this number is very dependent on the number of occurrences of the motifs in the sequence, which is itself related to the ratio of $n/4^s$. If this ratio is high, we expect a high number of occurrences of motifs and, consequently, a high computation time. This is what we observed with random data, as illustrated in Figure 6. We show here the case for $s = 6$, but the results are similar for other values of s, with the time scale increasing exponentially when s decreases.

In practice, the program can generate sequences up to a length of 100000 with $|\mathcal{M}|$ up to several dozens of motifs in a few minutes on a standard PC.

We are also trying to improve the processing time. We have found that a number of motifs appear "naturally" in almost any (unconstrained) shuffled sequence, depending on their length and on the nucleotide composition of the starting sequence. Therefore, we can use a variant of the algorithm. We divide $\mathcal{M}$ into two multisets $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $\mathcal{M}_1$ contains the "more likely" motifs and $\mathcal{M}_2$ contains the "less likely" motifs. We then produce the constrained sequence graph on $\mathcal{M}_1$ using only step (i) of the algorithm, and consider $\mathcal{M}$ in its entirety in step (iii). As almost any sequence contains the motifs of $\mathcal{M}_2$, and as step (iv) may be faster, the total processing time is much improved.

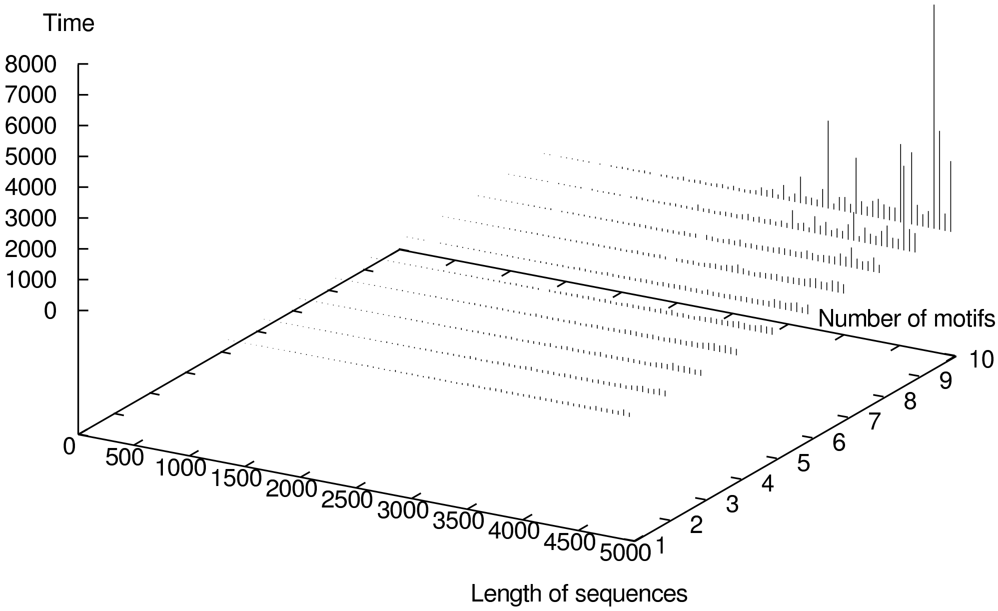

Fig. 6. Experiments on random data, with $k = 3$ and $s = 6$.

6 Acknowledgement

We are grateful to Bodo Lass for his help. This research was partially supported by the French IMPG Program ad the project "π-vert" of the ACI "New Interfaces of Mathematics".

References

1. T. van Aardenne-Ehrenfest and N.G. de Bruijn, *Circuits and trees in oriented linear graphs*, Simon Stevin (= Bull. Belgian Math. Soc.) **28** (1951), pp. 203–217.
2. D. Aldous, *A random walk construction of uniform spanning trees and uniform labelled trees*, SIAM Journal on Discrete Mathematics **3** (1990), pp. 450–465.
3. S. Altschul and B. Erickson, *Significance of nucleotide sequence alignments: a methode for random sequence permutation that preserves dinucleotide and codon usage*, Mol. Biol. Evol. **2** (1985), pp. 526–538.
4. S. Altschul, T. Madden, A. Schäffer, J. Zhang, Z. Zhang, W. Miller and D. Lipman, *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs*, Nucleic Acids Research **25** (1997), pp. 3389–3402.
5. E. Balas and C. Yu, *On graphs with polynomially solvable maximum-weigth clique problem*, Networks **19** (1989), pp. 247–253.
6. E. Beaudoing, S. Freier, J. Wyatt, J. Claverie and D. Gautheret, *Patterns of variant polyadenylation signal usage in human genes*, Genome Research **10** (2000), pp. 1001–1010.

7. A. Broder, *Generating random spanning trees*, in: *proc. of 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989, pp. 442–447.
8. A. Denise, M. Régnier and M. Vandenbogaert, *Assessing statistical significance on overrepresented oligonucleotides*, in: O. Gascuel and B. Moret, editors, *Proceedings of WABI'01*, Lecture Notes in Computer Science **2149** (2001), pp. 85–97.
9. W. Fitch, *Random sequences*, Journal of Molecular Biology **163** (1983), pp. 171–176.
10. M. R. Garey and D. S. Johnson, "Computers and intractability : a guide to the theory of NP-completeness." W. H. Freeman and Company, 1979.
11. M. Grotschel, L. Lovasz and A. Schrijver, *Polynomial algorithms for perfect graphs*, Annals of Discrete Mathematics **21** (1984), pp. 322–356.
12. D. Kandel, Y. Matias, R. Unger and P. Winkler, *Shuffling biological sequences*, Discrete Applied Mathematics **71** (1996), pp. 171–185.
13. D. Lipman, W. Wilbur, T. Smith and M. Waterman, *On the statistical significance of nucleic acid similarities*, Nucleic Acids Research **12** (1984), pp. 215–226.
14. P. Nicodème, B. Salvy and P. Flajolet, *Motif statistics*, Theoretical Computer Science **287** (2), 2002, pp. 593–618.
15. J. Propp and D. Wilson, *How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph*, Journal of Algorithms **27** (1998), pp. 170–217.
16. M. Régnier, *A unified approach to word occurrence probabilities*, Discrete Applied Mathematics **104** (2000), pp. 259–280.
17. G. Reinert, S. Schbath and M. Waterman, *Probabilistic and statistical properties of words: An overview*, Journal of Computational Biology **7** (2000), pp. 1–46.
18. J. van Helden, *Metrics for comparing regulatory sequences on the basis of pattern counts*, Bioinformatics **20** (2004), pp. 399–406.
19. J. van Helden, B. André and J. Collado-Vides, *Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies*, Journal of Molecular Biology **281** (1998), pp. 827–842.
20. A. Vanet, L. Marsan and M.-F. Sagot, *Promoter sequences and algorithmical methods for identifying them*, Res. Microbiol. **150** (1999), pp. 779–799.
21. D. Wilson, *Generating random spanning trees more quickly than the cover time*, in: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, 1996, pp. 296–303.