Shuffling Biological Sequences with Motifs Constraints

Dominique Barth

PRiSM, UMR CNRS 8144
Université de Versailles-St-Quentin
Versailles, France.

Johanne Cohen

LORIA, UMR CNRS 7503
Université de Nancy
Nancy, France.

Alain Denise and Romain Rivière [1]

LRI, UMR CNRS 8623
Université Paris-Sud XI
Orsay, France.

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 occurrences of k-lets (i.e. factors of length k) of S. This question involves difficult problems: We notably state that the problem of deciding whether a sequence respects given motifs constraints is NP-complete. Meanwhile, we give an random generation algorithm which turns out to be experimentally efficient.

Key words: biological sequence analysis, random generation, NP-complete problems

[1] To whom correspondence should be adressed. LRI, Bât 490. Université Paris-Sud. 91405 Orsay Cedex. France. Email: Romain.Riviere@lri.fr. Tel: +33 1 69 15 70 99. Fax: +33 1 69 15 65 86.

1 Introduction.

As the amount of data from sequenced genomes increases faster and faster, there is a crucial need for efficient computer-based ways to extract new biological information from sequences. For this purpose, a widely used method consists in comparing biological sequences with random ones. These last ones represent the "background noise", from which relevant biological information is due to stand out. This powerful paradigm is implemented in several fields in sequence analysis. An emblematic example is the search for exceptional motifs, *i.e.* patterns which are over- or underrepresented in a biological sequence, by comparison to the law of their expected number of occurrences in random sequences. The underlying hypothesis is that overrepresented or underrepresented motifs may point out important biological functions (see *e.g.* [19,20]). Another field where random sequences are of great use is sequence comparison. Pairwise sequence comparison algorithms give a score that measures their similarity. A crucial problem is to decide, given the score of an alignment, 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 of comparison of the biological sequences with random ones [4,13].

For the observations to be relevant, random sequences must obey to a model that takes into account some well-chosen characteristics of biological ones. The two widely acknowledged models of random sequences are based on the numbers of occurrences of all k-lets, *i.e.* all motifs of given fixed length k, in one or several biological sequence taken as a reference [9]. In the first model, the random sequences respect *in average* the given numbers of occurrences; they obey to a stationary Markov chain. In the second one, any random sequence contains *exactly* the same numbers of occurrences of k-lets as the reference. The first model is well suitable for long sequences, or large sets of sequences, and is widely used in the context of searching for exceptional motifs. On the other hand, when one has to handle one or a few rather short sequences, the latter fits better, notably because Markov chains may not be irreducible in this case. That is why it is used in the context of comparison of genes, which are rather short sequences. Random sequences are studied both in an analytical and in an algorithmic point of view. Indeed, various analytical methods have been developed for studying the probability distribution of motifs in random sequences, in order to search for exceptional motifs (see *e.g.* [14,16,17].) Meanwhile, in many cases one has to proceed experimentally, by generating sets of random sequences. In particular, this is necessary in the context of sequence comparison, where theoretical results are still scarce. The problem of random generation of sequences according to the Markovian model is straightforward. Regarding the second model (*exact* model), the problem is much more difficult. The first efficient algorithm was given by Kandel, Matias, Unger and Winkler in 1996 [12].

Recent works in biological sequence analysis point out to the necessity to

consider models of random sequences that contain more informations than the classical ones. For example, in [6] a set of sequences where one known motif is strongly overrepresented is given, and the authors aim to find other weaker overrepresented signals. It is shown in [8] that this can be done in a quite general manner by conditioning the occurrence probabilities by the strong signal, in other words by taking it into account in the model of random sequence. In [18], classification of genes is processed according to the counting of occurrences of a set of overrepresented motifs. For practical reasons, all motifs are supposed independant from each other, although some are related to others. The resulting classification might be improved if one could take into account these dependencies in the model. In such works, one has to consider a model of random sequences which takes in account the presence or the overrepresentation of certain motifs in biological sequences.

Thus, in the present paper, 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 the sequences must contain, additionally to the k-lets as above, a given number of occurrences of each motif from the set. In Section 2, we recall the algorithm of Kandel *et al.*, which generates sequences without additional constraints. It constitutes the starting point of our work. Then, in Section 3, we present our approach. Adding motif constraints in the model leads to difficult problems: We notably state that the general problem of deciding whether a sequence respects the given motif constraints is NP-complete. Meanwhile, we give an algorithm which turns out to be experimentally efficient. This is shown in Section 4, where experimental results are given.

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 *shuffled sequence* any sequence which has exactly the same numbers of occurrences of k-lets as S. For example, let S =ACTACTCACG and $k = 3$. 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. 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 means that all shuffled sequences must have the same probability to be generated.

We recall first a correspondence between the set of shuffled sequences and the set of Eulerian trails of a particular graph that we call the *sequence graph of order* k of S.

Definition 2.1 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). We present in Figure 1 an example of sequence graph.

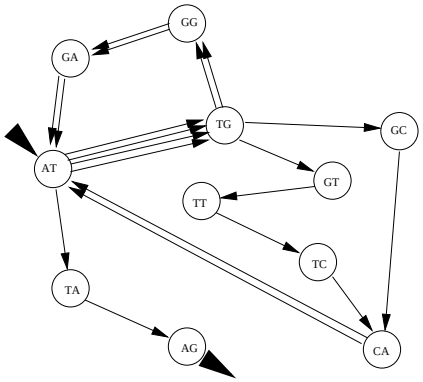

Fig. 1. Sequence Graph of S=ATGTTCATGCATGGATGGATAG with k=3.

In other words, 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: the one given by the sequence of nodes $(s_{[i,i+k-2]})_{i=1}^{i=n-k+2}$. In the following, we note v_b (resp. v_e) the vertex which begins (resp. ends) the Eulerian trail. Note that a sequence graph may be not only path-Eulerian, but Eulerian (*i.e.* cycle-Eulerian). In this case, v_b can be any vertex, and $v_e = v_b$. In any other case, 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 2.2 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 last letter of every other node in the path.

For example, the trace of the path (AT, TG, GG, GA, AT, TG, GT, TT, TC) in Figure 1 is the word ATGGAGTTC. Now the claimed correspondence can be stated:

Proposition 2.3 *Any trace corresponds to exactly one shuffled sequence. And the number of Eulerian trails which correspond to any given trace does not*

depend on the trace: it is equal to $\prod_{v \in V} d^+(v)$*, where* $d^+(v)$ *stands for the outdegree of vertex* v*.*

This correspondence, which was noticed by Fitch [9] in 1983, constitutes the basis of further works by Altschul and Erickson [3] and then by Kandel, Matias, Unger and Winkler [12]. Thus they reduce the problem of generating u.a.r a shuffled sequence to the one of generating u.a.r. an Eulerian trail in a (particular) directed multigraph. The next step needs to make use of the BEST theorem [1], which links Eulerian trails and spanning trees of a graph. In the present context, this theorem states as follows:

Theorem 2.4 (Aardenne-Ehrenfest and de Bruijn, 1951.) *The number of Eulerian trails in* G *which 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 $\mathcal{T}(G)$ *denotes 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 of random generation of Eulerian trails, provided that one is able to generate u.a.r. an arborescence in G: Start at the beginning vertex. At each step, choose u.a.r an arc among all arcs from the current vertex which have not be crossed yet *but the arc which belongs to the arborescence*. This arc can be chosen only if no other arc is available. Then follow the arc and go to the next vertex, which becomes the new current one. The process stops at v_0 when all arcs have been crossed.

Now the problem of generating u.a.r. an Eulerian trail is reduced to the problem of generating u.a.r. an arborescence in G. The algorithm given in [12] is a variant of a very elegant one which was found independently by Aldous [2] and by Broder [7] for undirected graphs. The process states as follows. If G only path-Eulerian, it begins by making it Eulerian by adding a "virtual" arc between v_e and v_b. Then proceed a free random walk in G and, each time an arc is crossed, add it to the arborescence if it is not the virtual one and if no cycle occurs in the resulting arborescence. The expected time complexity of the algorithm is $O(q^2 n)$, where q stands for the number of vertices, *i.e.* the number of distinct k-lets in the sequence S. More recently, Propp and Wilson [15,21] have designed new algorithms, based on similar principles, which improve the time complexity.

3 Shuffling sequences with motif constraints.

3.1 Preliminaries

In the present section, we address the problem of generating shuffled sequences that are subject to additional constraints. As above, we consider a reference

sequence S of length n 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 $\mathcal{M}$ than in S. Overlapping occurrences are not taken into account, *i.e.* if two motifs occurrences overlap in the sequence, then only one is counted. In the following, we call *motifs* the words of $\mathcal{M}$.

The problem consists in generating sequences that have exactly the same k-lets count as S, and containing at least as many occurrences of each motif of $\mathcal{M}$ as its number of occurrences in $\mathcal{M}$. As above, overlapping occurrences are not taken into account. We call *acceptable sequence* any sequence which respects these conditions. As motifs are taken (without overlap) in the reference sequence S, we are ensured that there exists at least one acceptable sequence.

Briefly, our approach consists in two main stages, which are developed respectively in Sections 3.2 and 3.3. In the first stage, we define from $Gr(S, k)$ a new multi-digraph in which each acceptable sequence is the trace of an Eulerian trail. Then we uniformly generate at random an Eulerian trail, and we verify if the corresponding trace corresponds to an acceptable sequence (as we will see, this is not always the case). We show that this last step involves an NP-complete problem. Meanwhile, we propose a simple algorithm which is efficient in practice, as we will see in Section 4. The second stage aims to ensure the uniformity of the random generation. For this purpose, we need to compute the number of Eulerian trails which correspond to any generated trace. Unlike in the original shuffling problem (see Proposition 2.3), here this number strongly depends on the given trace. We give a method to compute it.

Here are two major definitions involving acceptable sequences.

Definition 3.1 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 3.2 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

$$\forall (j, l) \in \{1, \ldots, p\} \ j \neq l, i_j + |M_j| - 1 < i_l \quad or \quad i_l < i_j$$

In other words, there is no overlap between any two occurrences in a perfect configuration. Clearly, a sequence is acceptable if and only if it has a perfect configuration over $\mathcal{M}$.

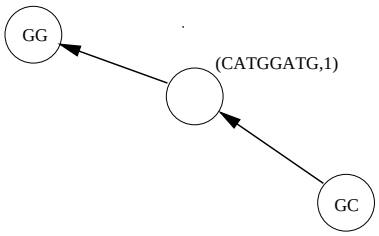

Fig. 2. Sequence Cluster of M_1 =GCATGGATGG, with $k = 3$

3.2 Generating acceptable sequences

3.2.1 Constrained sequence graphs

Definition 3.3 A *sequence cluster* of order k of a word $M_i = m_1 \dots m_{r_i} \in \mathcal{M}$, denoted $Ch_i(M_i, k)$, is a directed multigraph $C = (V, E)$ where

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

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

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

An example of sequence cluster is given in Figure 2.

Definition 3.4 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'), whith

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

and

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

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

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

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

The notion of trace of a sequence graph naturally extends to the constrained sequence graphs, with the following change: when one crosses a cluster node (w, i), one has to concatenate not only the last letter of the word w but its $|w| - k$ last letters. Similarly to Section 2, the following simple result

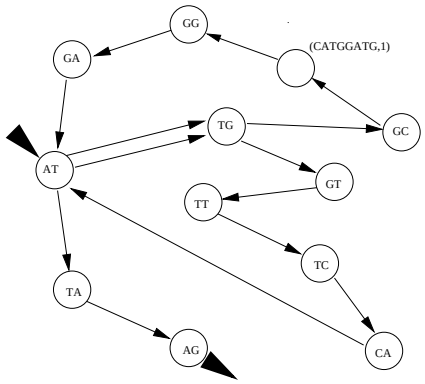

Fig. 3. constrained sequence graph of ATGTTCATGCATGGATGGATAG with $\mathcal{M} = [\text{GCATGGATGG}]$ and $k = 3$.

shows that there is a direct link between acceptable sequences and Eulerian trails in a constrained sequence graph.

Proposition 3.5 *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 word and $J = (j_1, \ldots, j_p)$ 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 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. Thus, calling $C(M_i)$ the cluster node associated with M_i, $T' = (s_{[1,k]}, \ldots, s_{[i_1,i_1+k-1]}, C(M_1), \ldots, s_{[i_p,i_p+k-1]}, C(M_p), \ldots, s_{[n-k+1,n]})$ is an Eulerian trail in $GrC(S, k, \mathcal{M})$ whose trace is S. □

3.2.2 Searching for perfect configurations.

Unfortunately, not all Eulerian trails give raise to an acceptable sequence, because motifs may overlap, as shown in Figure 4. We thus have to verify, when a random sequence S is generated, if it contains a perfect configuration. This problem, that we call PCS for "Perfect Configuration Searching", is defined as follows in its generality:

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 $\mathcal{M}$?

Theorem 3.6 *PCS is NP-complete.*

Proof. With the definition, we can verify in polynomial time whether a given configuration is perfect or not. So PCS is in NP. Now we will reduce PCS to Three dimensional matching (3DM) (see example 3.7). Problem 3DM [10] is defined as follows:

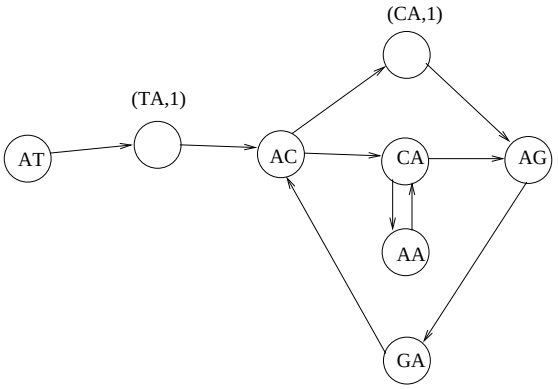

Fig. 4. In this constrained sequence graph with $\mathcal{M}$ =[ATAC,ACAG], the Eulerian trail (AT,TA,AC,CA,AG,GA,AC,CA,AA,AG) gives sequence ATACAGACAAG, which is not acceptable because the only occurrences of ATAC and ACAG are overlapping.

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 length 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 $\mathcal{C}$.

For any finite set D, there exists a bijection ϕ which maps D to the set of integers $\{1, \ldots, |D|\}$, $\forall d \in D$ we note $0^d = 00\ldots0$, where 0 is repeated $\phi(d)$ times.

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 M_x as the multiset built as follows : it contains

(i) one occurrence of the motif $m_x = 0^{x-1}10^{q-x}ab$

(ii) $f_{\mathcal{C}}(x) - 1$ occurences of the motif $m'_x = a0x^{x-1}10^{q-x}a$

Similary, we define $\forall y \in Y$ (resp. $\forall z \in Z$) 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 just need the two following 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 considering independently 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 $c \in \mathcal{C}$.

(i) 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. On the other hand, there is only one occurrence af x, y and z respectively in $\mathcal{C}'$, by definition. Thus, after processing this operation for all $c = (x, y, z) \in \mathcal{C}'$, all the factors w_c of S for $c \in \mathcal{C}'$ are recovered (unless the initial a letter in each of them), and only the motifs m_x, m_y and m_z of M have been used.

(ii) 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 $f_{\mathcal{C}}(x)$ (resp. $f_{\mathcal{C}}(y)$, $f_{\mathcal{C}}(z)$) times in $\mathcal{M}$, finally all the factors w_c of S for $c \notin \mathcal{C}'$ are recovered (unless the terminal 0 in each of them), and all the motifs m'_x, m'_y and m'_z of M have been used.

Finally, all motifs of $\mathcal{M}$ have been used, and no two ones 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 a perfect configuration of $\mathcal{M}$ over S. We shall 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, the part w_x of the factor of S $w_c = w_x w_y w_z 0$ is (partially) recovered by the motif $m_x = 0^{x-1}10^{q-x}ab$ of $\mathcal{M}$.

We have:

- $|\mathcal{C}'| = |\{m_x : x \in X\}| = |X| = q$.
- $\mathcal{C}'$ 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, w_x is recovered by m_x by definition. And 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 $r, s \in X \cup Y \cup Z$) 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 $\mathcal{C}'$.

This concludes the proof. □

Example 3.7 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.

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

At this stage, it is worth noticing that the sequences that we deal with are not general ones. They are particular because they result from an Eulerian trail in a sequence graph. Consequently, we must consider the problem of searching a perfect configuration in such sequences. Definition 3.8 and Proposition 3.9 will allow us to formalise their particularity.

Definition 3.8 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

$$\forall (j,l) \in \{1,\ldots,p\}\ j \neq l, i_j + |M_j| - k < i_l \quad or \quad i_l < i_j$$

This means that all the words pointed by the configuration overlap by length at most $k-1$. We shall omit the parameter k when explicit reference to a constrained sequence graph is given. In this case, k is the order of the graph. Now the following property holds:

Proposition 3.9 *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 node in $GrC(S,k,\mathcal{M}) = (V',E')$, then there exist $[t_{i_1},\ldots,t_{i_p}]$ which are cluster node. By construction, there is no $[t_{i_{l_1}}, t_{i_{l_2}}] \in E'\ \ \forall l_1, l_2$ so there must exist $t_j \in T$ such that $i_{l_1} < j < i_{l_2}$. So, there exists occurrences $m_{i_{l_1}}$ and $m_{i_{l_2}}$ that overlap in S by at most $|t_j| = k-1$ and S contains a pseudo-perfect configuration over $\mathcal{M} = [M_1,\ldots,M_p]$.

On the other hand, 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' out of an Eulerian trail T in $Gr(S,k)$ as in proposition 3.5. □

Thus the actual problem we are adressing, 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 ?

Theorem 3.10 *Problem FPCS is NP-complete.*

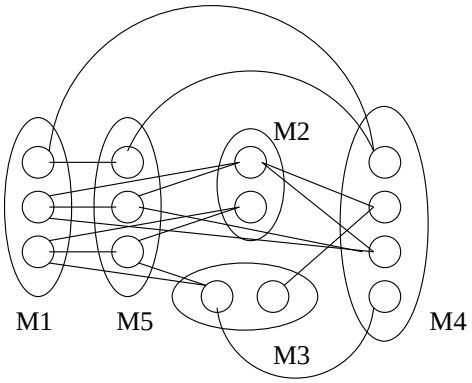

Fig. 5. The overlapping Graph of $\mathcal{M}$=[ATT,TATT,CGAT,TTAT,ATT] over S =ATTATCGATTATATTATCCGACGATTATTC.

Proof. It suffices to prove the result for $k = 2$. It is easy to see that Problem FPCS belongs to NP. Moreover, we transform 3DM to FPCS by the transformation given in 3.6. 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 3.6. 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 Theorem 3.6, 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 3.6, there are 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. □

3.2.3 An algorithm for FPCS

Although we have just stated that FPCS is NP-complete, we present here an algorithm which turns out to be efficient in realistic cases (see Section 4). At first, let us define the *overlapping graph* of $\mathcal{M}$ over S:

Definition 3.11 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 there is an arc between two given nodes if the occurrences they are associated with are overlapping (in the sense of Definition 3.2) in S.

An example of overlapping graph is given in Figure 5. Now, given an overlapping graph G, the algorithm is based on a classical arborescent search

method. We suppose that the motifs of $M = [M_1, \ldots, M_p]$ are ordered, as well as the occurrences $[M_{i,1}, \ldots, M_{i,p_i}]$ of each motif M_i. Then the algorithm proceeds as follows. Take the first occurrence of the first motif and delete all its neighbors. Then proceed 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 last case, backtracking is processed in order to find a suitable sequence of occurrences.

There is also a direct interpretation of a perfect configuration in terms of the graph G: If one considers adding edges making cliques on all the vertex-occurrences of a same motif, 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 u.a.r.

We now focus on the problem of generating random sequences *uniformly*. In the case of constrained sequence graphs, no nice property like Proposition 2.3 holds: The number of Eulerian trails which correspond to a given trace strongly depends on this trace. Consequently the generation process is not uniform *a priori*. Thus we use a classical rejection method for the generation to be uniform: When a trace is generated, we accept it with a probability proportional to its number of corresponding Eulerian trails, or we reject it and start the process again. Hence we need to count the number on Eulerian trails which correspond to a given trace.

Proposition 3.12 *The number of eulerian trails which correspond to any given trace 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 proposition 3.5 and 3.9, we have seen how we could map a k-pseudo-perfect configuration to an eulerian trail in $GrC(S, k, \mathcal{M})$ hence the numerator. However, $\mathcal{M}$ being a multiset, two configurations will be mapped to the same eulerian trail if two occurence number of the same motifs are swapped hence the denominator. □

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

Definition 3.13 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 M is a distinct node, and there exists an edge between two given nodes if the occurrences they are associated with are overlapping by length at least k.

Consider the pseudo-overlapping graph in which all the nodes corresponding to occurrences of any same word are connected together in a clique. Obviously, the number of maximal independent sets (MIS) in this graph equals the number of perfect configurations in the related sequence. The problem of counting MIS is known to be polynomial in intersection graphs (including interval graphs) [5]. Although each pseudo-overlapping graph is clearly an interval graph, it is easy to see that the graphs we consider here are not even perfect graphs (the problem of determining the cardinal of a MIS is polynomial for perfect graphs but NP-complete for general graphs, see [11,10] and refs.). Thus, we conjecture that the problem of knowing if the number of pseudo-perfect configurations is greater or equal to a given integer is NP-complete.

3.4 The random generation algorithm.

We are now able to state the whole 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) *Generate uniformly 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 which correspond to this particular trace T.*
(v) *Return T with probability $1/N$, or goto (ii).*

Remark that if we were able to compute a lower bound m of the minimum over the traces of the number of eulerian trail associated to any traces, we could replace the rejection probability in (v) by m/N. However, this number seems to be very difficult to compute in general.

Proposition 3.14 *Step (iv) of algorithm 1 is called R times in average, where R is the average number of Eulerian trails per trace.*

Proof. Consider the square $[0, 1]^2$. To any given trace of eulerian trail, consider an interval of $[0, 1]$ of the length the probability of choosing this particular trace. Put those intervals, in any given order, one after another. Then, with each interval, construct a rectangle of height the probability of keeping this trace according to algorithm 1. For every rectangle, the area is constant. The

sum of those area is equal to number of different traces over the number of different eulerians. One should easily verify that this number is in fact the inverse of the mean over the traces of the number eulerian trail associated to any trace, which is the expected number of steps needed to hit one of those rectangle and thus stopping the algorithm. □

4 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)).

Thus we will perform simulations on random data in order to get an idea of the average complexity of those routines. Our main goal is to provide some hints about what can and what cannot be done in terms of the size of the parameters. Essentially, routines 2 and 3 above involve performing some arborescent search over an overlapping graph. The first one needs only one "good" search and then stops, but the other one needs in many case to search in all the search tree. As we will see, this difference will make the two algorithms different in terms of what make 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 $t = 4$ as we are interested in DNA sequences. Then, given the cardinality p of $\mathcal{M}$ and the size s of its motifs we generated the multiset $\mathcal{M}$ by choosing p positions in the sequence and taking, for each position, the word of length s which begins at it. Thus, all motifs had the same length.

Our first concern is about the number of times step (ii) is processed. In fact, the experiments showed that, for relatively small k, almost all sequences that were produced contained a perfect configuration and, as a consequence, the algorithm behaves as if there were no rejection.

Intuitively, a difficult case for routine 2 occurs when n is far larger than 4^s, because in this case the motifs tend to have more than one occurrence in S and, as a consequence, the overlapping graph has many nodes. The problem should be even more difficult if those occurrences overlap, and this is the case when the motifs are numerous and large enough. Another condition is that $k >> 1$, because any trace of an Eulerian trail already contains a (k)-pseudo-perfect configuration. In summary, in order to get difficult cases, we need to have $n >> 4^s$ and $s > k >> 1$. So, we need say $k \geq 10$, $s \geq 11$ and as a consequence n should be greater than 10^8. The graph library that we used for our implementation did not allow us to investigate those values efficiently. Thus we restricted our simulations to $k = 5$ and found no case

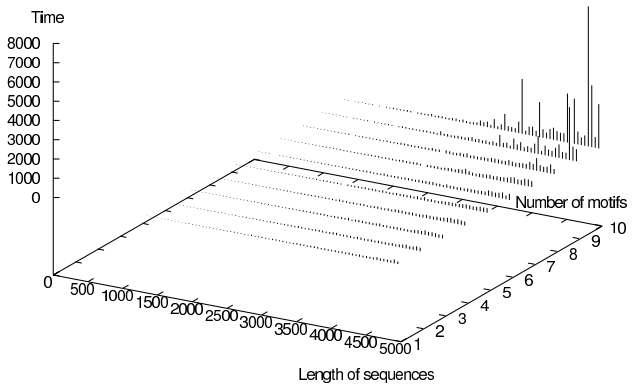

Fig. 6. Experiments on random data, with $k = 3$ and $s = 6$.

when $n < 100000$ $|\mathcal{M}| < 1000$ where the computation time of routine 2 is significant. (Note that $k = 5$ is a rather standard value for shuffling DNA sequences for Bioinformatics purposes.)

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

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

Finally, here is a method that we are experimenting for improving the processing time. One may notice that a number of motifs appear "naturally" in almost any (unconstrained) shuffled sequence, depending on their length and on the nucleotidic composition of the starting sequence. This leads to the following variant of the algorithm: 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" ones. Then produce the constrained sequence graph on $\mathcal{M}_1$ only in step (i) of the algorithm, and consider $\mathcal{M}$ in its entirety in step (iii). Since almost any sequence contains the motifs of $\mathcal{M}_2$, and since step (iv) may be faster, total processing time is indeed much improved.

Acknowledgement

We are grateful to Bodo Lass for his help. This research was partially supported by French IMPG Program.

References

[1] Aardenne-Ehrenfest, T. v. and N. de Bruijn, *Circuits and trees in oriented linear graphs*, Simon Stevin (= Bull. Belgian Math. Soc.) **28** (1951), pp. 203–217.

[2] Aldous, D., *A random walk construction of uniform spanning trees and uniform labelled trees*, SIAM Journal on Discrete Mathematics **3** (1990), pp. 450–465.

[3] Altschul, S. 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. 256–538.

[4] Altschul, S., 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] Balas, E. and C. Yu, *On graphs with polynomially solvable maximum-weigth clique problem*, Networks **19** (1989), pp. 247–253.

[6] Beaudoing, E., 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] Broder, A., *Generating random spanning trees*, in: *30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989, pp. 442–447.

[8] Denise, A., 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] Fitch, W., *Random sequences*, Journal of Molecular Biology **163** (1983), pp. 171–176.

[10] Garey, M. R. and D. S. Johnson, "Computers and intractability : a guide to the theory of NP-completeness." W. H. Freeman and Company, 1979.

[11] Grotschel, M., L. Lovasz and A. Schrijver, *Polynomial algorithms for perfect graphs*, Annals of Discrete Mathematics **21** (1984), pp. 322–356.

[12] Kandel, D., Y. Matias, R. Unger and P. Winkler, *Shuffling biological sequences*, Discrete Applied Mathematics **71** (1996), pp. 171–185.

[13] Lipman, D., W. Wilbur, T. Smith and M. Waterman, *On the statistical significance of nucleic acid similarities*, Nucleic Acids Research **12** (1984), pp. 215–226.

[14] Nicodème, P., B. Salvy and P. Flajolet, *Motif statistics*, Theoretical Computer Science. To appear.

[15] Propp, J. 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] Régnier, M., *A unified approach to word occurrence probabilities*, Discrete Applied Mathematics **104** (2000), pp. 259–280.

[17] Reinert, G., S. Schbath and M. Waterman, *Probabilistic and statistical properties of words: An overview*, Journal of Computational Biology **7** (2000), pp. 1–46.

[18] van Helden, J., *Metrics for comparing regulatory sequences on the basis of pattern counts*, Bioinformatics **20** (2004), pp. 399–406.

[19] van Helden, J., 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] Vanet, A., L. Marsan and M.-F. Sagot, *Promoter sequences and algorithmical methods for identifying them*, Res. Microbiol. **150** (1999), pp. 779–799.

[21] Wilson, D., *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.