Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification

Laurent Marsan [1]

Marie-France Sagot [1,2,*]

[1]Institut Gaspard Monge
Université de Marne la Vallée
5, bd Descartes, Champs-sur-Marne
77454 - Marne-la-Vallée Cedex 2
marsan@univ-mlv.fr

[2]Institut Pasteur
Service d'Informatique Scientifique
28, rue du Dr. Roux
75724 - Paris Cedex 15
sagot@pasteur.fr
* corresponding author

Abstract

This paper introduces two exact algorithms for extracting conserved structured motifs from a set of DNA sequences. Structured motifs may be described as an ordered collection of $p \geq 1$ "boxes" (each box corresponding to one part of the structured motif), p substitution rates (one for each box) and $p-1$ intervals of distance (one for each pair of successive boxes in the collection). The contents of the boxes – that is, the motifs themselves – are unknown at the start of the algorithm. This is precisely what the algorithms are meant to find. A suffix tree is used for finding such motifs. The algorithms are efficient enough to be able to infer site consensi, such as, for instance, promoter sequences or regulatory sites, from a set of unaligned sequences corresponding to the non coding regions upstream from all genes of a genome. In particular, both algorithms time complexity scales linearly with N^2n where n is the average length of the sequences and N their number. An application to the identification of promoter and regulatory consensus sequences in bacterial genomes is shown.

key words: structured motif extraction, promoter and regulatory site, consensus, model, suffix tree

1 Introduction

DNA binding-site identification is an important problem in molecular biology, especially as we enter the era of large-scale genome sequencing. Besides being a major biological issue in itself, accurately identifying ribosome binding sites, promoter and other regulatory sequences, may thus greatly enhance our capacity to correctly predict genes and, in some cases, gene function.

There are two problems related to this identification. One is binding-site location prediction, the other binding-site consensus extraction. Location prediction algorithms often use the results produced by consensus extraction methods to establish precise site position. This paper addresses the second kind of problem: extracting consensus motifs for DNA binding sites.

The two problems are difficult. Consensus extraction in particular is hard both biologically and computationally. Current algorithms are limited in either the biological models upon which they are built, or the quantity and type of data they may treat (see [Brazma *et al.*, 1998a] for a general survey and [Vanet *et al.*, 1999] for the more specific case of promoter consensus identification). Up to recently, consensi were therefore often derived from relatively small, well-established and clean datasets [Galas *et al.*, 1985] [Lawrence and Reilly, 1990] [Stormo and Hartzell, 1989] [Queen *et al.*, 1982]. Very few approaches (all of which either deal with exact, possibly degenerate motifs or are heuristics) have confronted the problem of extracting site consensi *ab initio* from a whole genome [Brazma *et al.*, 1998b] [Tompa, 1999] [van Helden *et al.*, 1998] or considered the fact that binding sites often come together in a well-ordered and regularly spaced manner [Fraenkel *et al.*, 1995] [Cardon and Stormo, 1992] [Klingenhoff *et al.*, 1999] [van Helden *et al.*, 2000] [Wolfertstetter *et al.*, 1996]. This latter characteristic may be because two sites are recognized by the same protein (as is, for instance, frequently the case of the RNA polymerase) [Lewin, 1997] or because the sites are recognized by different macromolecular complexes that then make contact with one another [Werner, 1999].

This paper presents two exact algorithms that are efficient enough to tackle site consensus extraction from big datasets (possibly concerning a whole genome). Furthermore, the algorithms take into consideration the fact that binding sites may be multiple and may present a constrained spatial structure along the DNA chain. Such algorithms should therefore enable to identify in genomic sequences what we shall call *structured motifs*. A structured motif may be described as an ordered collection of $p \geq 1$ "boxes" (each box corresponding to one part of the structured motif), p maximum error rates (one for each box) and $p-1$ intervals of distance (one for each pair of successive boxes in the collection). The contents of the boxes – that is, the motifs themselves – are unknown at the start of the algorithm. This is precisely what the algorithms are meant to find. A suffix tree is used for finding such motifs. The second algorithm has a better time complexity than the first but needs more space. The first is easier to understand and implement. Both have a time complexity whose dominating terms are $N^2 n$ and $k_{total}^{e_{global}}$, where n is the average length of the sequences, N their number, k_{total} the total maximum length of the motifs (spacers between parts excluded) and e_{global} a global maximum number of errors allowed.

Section 2 presents definitions and formally states the addressed problem. Section 3 is a reminder of the use of a suffix tree for single motif extraction [Sagot, 1998]. Section 4 introduces the new algorithms for structured motif extraction. To simplify exposition of the main ideas, the algorithms are described for the case of motifs composed of two parts of equal length, each having the same maximum rate of errors and separated by a distance. We then suggest how to extend this to the extraction of general structured motifs composed of $p > 2$ parts. In a first version, the value of the distance between parts belongs to a known interval. In a second version, it is just known to vary inside a restricted interval whose limits are unknown. We end by showing an application to promoter consensus identification from whole bacterial genomes.

2 Definitions and Statement of the Problem

2.1 Motifs as Models – A Reminder

Let us start by establishing some of the terms we are going to use. The term "motif" in particular will be replaced by the pair (*model*, *occurrence*). A motif has often been employed in the literature to denote both something that is in a sequence, *i.e.* a word, and the representative or representation of a set of such words (that possibly satisfy a certain property). We wish to keep both concepts separate. An occurrence shall be "something" in the sequence and a model something that "represents" ("models") a set of occurrences. Models will thus serve to locate, as well as describe DNA binding sites in a set of sequences. They may never be present in the sequences.

More formally, let Σ be the alphabet {A, C, G, T} of nucleotides. An element $u \in \Sigma^+$ is said to be a word in a sequence $s \in \Sigma^+$ if $s = xuy$ for $x, y \in \Sigma^*$. An element $m \in \Sigma^+$, called a *model*, is said to have an e-occurrence (or simply an *occurrence*) in s for e a non negative integer if there is at least one word u in s such that the Hamming distance (*i.e.* minimum number of substitutions) between u and m is no more than e. Given N sequences $s_1, \ldots, s_N \in \Sigma^*$ and an integer $1 \leq q \leq N$, an element $m \in \Sigma^+$ is said to be a *valid model* if it has at least an occurrence in at least q distinct sequences of the set (q is called the *quorum*). More complex definitions of models (e.g. as elements of $\mathcal{P}^+$ where $\mathcal{P}$ is the set of all subsets of Σ) and of distances between models and their occurrences may be used (e.g. a Levenshtein distance that represents the minimum number of substitutions, insertions and deletions between two objects of same type), but we shall not consider them here (see [Sagot and Viari, 1996] [Sagot *et al.*, 1995] and [Sagot *et al.*, 1997]).

2.2 From Single to Structured Models

Although the objects defined in the previous section can be reasonable, algorithmically tractable models for DNA binding sites, they do not incorporate any information concerning the relative positions of such sites when more than one participates in a biological process. It is common knowledge that these relative positions are often not random. For instance, the most frequently observed prokaryotic promoter sequences are in general composed of two parts, or "boxes", that come approximately 10 and 35 bases respectively upstream from the start of transcription. The two boxes, whose core sequences are six bases long, are therefore frequently situated 15 to 19 bases apart. The reason for this strict distance is that the boxes are recognized by the same protein, the RNA polymerase, in fact, a factor of the polymerase, the σ^{70}. RNA polymerases may have attached to them other σ factors. Prokaryotic promoters recognized by these factors are, for the same reason, commonly composed of two boxes although the distance between them may be different [Gross *et al.*, 1992]. There may also be other regulatory sites, whose distance from the promoter sequence is more variable but often not random. Activator binding sites are thus frequently located upstream from the promoter at a distance allowing for the activator to interact with the RNA polymerase. Repressor binding sites on the other hand may overlap the promoter sequences or come just downstream from it.

The case of eukaryotic transcription regulation is more complicated [Werner, 1999]. Promoter sequences contain, as for prokaryotes, one or two boxes recognized by a same protein, but there may be more regulatory sites, appearing sometimes repeated, that are recognized by distinct proteins which interact with one another. The relative positions of these sites along a DNA sequence remains thus not always indifferent [Werner, 1999]. Finally, enhancer sequences are an important feature of eukaryotic gene regulation. Such sequences may be located at very long distances from the gene start.

There is therefore a need for defining promoter or regulatory site models as objects that take such characteristics into account. This has the biological motivation just mentioned but presents also interesting algorithmical aspects: exploiting such characteristics could lead to algorithms that are both more sensitive and more efficient. Models that incorporate such characteristics are called *structured models*.

Formally, a structured model is a pair (m, d) where:

- m is a p-tuple of single models $(m_1, \ldots, m_p)$ (the p boxes);
- d is a $(p-1)$-tuple of triplets $((d_{min_1}, d_{max_1}, \delta_1), \ldots, (d_{min_{p-1}}, d_{max_{p-1}}, \delta_{p-1}))$ (the $p-1$ intervals of distance);

with p a positive integer, $m_i \in \Sigma^+$ and d_{min_i}, d_{max_i} $(d_{max_i} \geq d_{min_i})$, δ_i non negative integers. Given a set of N sequences $s_1, \ldots, s_N$ and an integer $1 \leq q \leq N$, a model (m, d) is said to be *valid* if: for all $1 \leq i \leq (p-1)$ and for all occurrences u_i of m_i, there exist occurrences $u_1, \ldots,$ u_{i-1}, $u_{i+1}, \ldots, u_p$ of $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_p$ such that:

- $u_1, \ldots, u_{i-1}, u_i, u_{i+1}, \ldots, u_p$ belong to the same sequence of the set;
- there exists d_i, with $d_{min_i} + \delta_i \leq d_i \leq d_{max_i} - \delta_i$, such that the distance between the end position of u_i and the start position of u_{i+1} in the sequence is equal to $d_i \pm \delta_i$;
- d_i is the same for p-tuples of occurrences present in at least q distinct sequences.

The term d_i represents a distance between the parts and $\pm\delta_i$ an allowed interval around that distance. When $\delta_i = (d_{max_i} - d_{min_i} + 1)/2$, δ_i is omitted and d in a structured model (m, d) is denoted by a pair (d_{min_i}, d_{max_i}). An example of a model with p equal to two is given in Figure 1.

2.3 Statement of the Problem

This paper proposes solutions to variants of increasing generality of a same basic problem. These variants may be stated as follows; given a set of N sequences $s_1, \ldots, s_N$, a non negative integer e and a positive integer $q \leq N$:

Problem 1 finds all models $((m_1, m_2), (d_{min_1}, d_{max_1}))$ that are valid;

Problem 2 finds all models $((m_1, \ldots, m_p), ((d_{min_1}, d_{max_1}), \ldots, (d_{min_{p-1}}, d_{max_{p-1}})))$ that are valid where $p \geq 2$;

Problem 3 finds all models $((m_1, m_2), (d_{min_1}, d_{max_1}, \delta_1))$ that are valid.

Problem 4 finds all models $((m_1, \ldots, m_p), ((d_{min_1}, d_{max_1}, \delta_1), \ldots, (d_{min_{p-1}}, d_{max_{p-1}}, \delta_{p-1})))$ that are valid where $p \geq 2$.

The last two problems represent situations where the exact intervals of distances separating the parts of a structured site are unknown, the only known fact being that these intervals cover a restricted range of values. How restricted is indicated by the δ_i parameters.

To simplify matters, we shall consider that, for $1 \leq i \leq p$, $m_i \in \Sigma^k$ where k is a positive integer, *i.e.* each single model m_i of a structured model (m, d) is of fixed, unique length k. In a likewise manner, we shall assume that each part m_i has the same substitution rate e and, when dealing with models composed of more than two boxes, that the d_{min_i}, d_{max_i} and, possibly, δ_i for $1 \leq i \leq (p-1)$ have identical values. We shall note d_{min}, d_{max} and δ these values. Problem 2 is then formulated as finding all models $((m_1, \ldots, m_p), (d_{min}, d_{max}))$ that are valid and Problem 4 as finding all valid models $((m_1, \ldots, m_p), (d_{min}, d_{max}, \delta))$.

Besides fixing a maximum substitution rate for each part in a structured model, one can also establish a maximum substitution rate for the whole model. Such a global error rate, noted e_{global}, allows to consider in a limited way possible correlations between boxes in a model.

A solution to Problem 1 is described in section 4.1, solutions to Problems 2 and 3 are given in sections 4.3 and 4.4 respectively. Solving Problem 4 implies simply putting together the solutions proposed for Problems 2 and 3. The complexity of such an operation is indicated in section 4.5. In section 4.6, we discuss how handling variable box lengths modify the previously described algorithms. Finally, section 4.7 introduces other kinds of constraints, either local or global such as, for instance, maximum substitution rate, which may be useful for some applications.

3 Single Model Extraction Using Suffix Trees – A Reminder

The algorithms for solving either of the problems stated in section 2.3 make use of a generalized suffix tree $\mathcal{T}$ of the set of sequences $s_1, \ldots, s_N$ given as input. This is the classical suffix tree introduced by McCreight [McCreight, 1976] and modified to consider $N \geq 1$ sequences [Bieganski *et al.*, 1994] [Gusfield, 1997]. The modification consists in:

- placing at the end of each sequence in the set a symbol not in the alphabet and specific to that sequence;
- storing at each node v in the tree a boolean-array of size N indicating the sequences in the set to which belong the strings labeling the path to v from the root of $\mathcal{T}$. Following a same notation as in [Sagot, 1998], we denote such an array $colors_v$.

The suffix tree construction we adopt is that of Ukkonen [Ukkonen, 1995]. To facilitate the exposition of the main ideas, a suffix trie is considered instead of a tree, *i.e.* arcs are labelled by a single letter. We shall refer to it as a suffix tree since adapting the algorithm to deal with a compact tree is straightforward. The level of a node v in the suffix trie will therefore denote the number of nodes in the paths from the root to v. In either trie or tree, this corresponds also to the length of the word the path spells.

Given a maximum number e of substitutions allowed, it has been shown in [Sagot, 1998] that extracting all valid single models, that is, all models $m \in \Sigma^{k \geq 1}$ verifying a quorum q, can be done by simultaneously and recursively traversing (in a depth-first way) the (virtual) lexicographic trie $\mathcal{M}$ of all possible models of length k (stopping the descent down an arc and pruning the subtree rooted at the end node of that arc as soon as the quorum is no longer verified) and the (actually built) suffix tree $\mathcal{T}$ of the sequences. While traversing $\mathcal{T}$, up to e mispellings are allowed of the labels of the arcs in $\mathcal{M}$ corresponding to potentially valid models. One may also see this operation as an application of a sparse dynamic programming technique to two trees instead of two sequences. Tree $\mathcal{T}$ is another way of representing a sequence while tree $\mathcal{M}$ explores all possible (valid) models.

Occurrences in the sequences of a model m that are identical are grouped (by their end-positions) into those whose spelling leads to a same node in $\mathcal{T}$. These occurrences correspond therefore to nodes in the tree. A node-occurrence of m is represented by a pair (v, e_v) where v is a node in $\mathcal{T}$ and e_v is the number of errors accumulated between a model m and the label of the path from the root to v (at all times, $e_v \leq e$). The main recurrence upon which the algorithm is based is given by the following Lemma where parent(v) denotes the parent of a node v in $\mathcal{T}$.

Lemma 3.1 *[Sagot, 1998] A pair (v, e_v) is a node-occurrence of $m' = m\alpha$ with $m \in \Sigma^l$ for $1 \leq l < k$ and $\alpha \in \Sigma$ if, and only if, one of the following two conditions is verified:*

(match) *A pair $(parent(v), e_v)$ is a node-occurrence of m and the label of the arc from $parent(v)$ to v is α;*

(subst.) *A pair $(parent(v), e_v - 1)$ is a node-occurrence of m and the label of the arc from $parent(v)$ to v is $\beta \neq \alpha$.*

Further details may be found in [Sagot, 1998]; an illustration is also given in Figure 2.

Since suffix tree $\mathcal{T}$ is used as just another way of representing the sequences, observe that it is simply "read" (*i.e.* traversed) over and over again to extract all valid single models. Once it is built, no use is made of the suffix links in $\mathcal{T}$, tree arcs only are followed. This will not be the case of the suffix tree for inferring structured models in one of the algorithms given below.

In [Sagot, 1998], it was shown that this algorithm has an $O(N^2 n \mathcal{V}(e,k))$ time complexity for finding models of length k. The term $\mathcal{V}(e,k)$ is called the e-neighbourhood of a k-long word m: it is the number of distinct words u that are at a Hamming distance at most e from m. We have:

$$\mathcal{V}(e,k) = \sum_{i=0}^{e} \left(\begin{array}{c} k \\ i \end{array} \right) (|\Sigma| - 1)^i \leq k^e |\Sigma|^e$$

Observe that the second N comes from the union operations that we need to do to the boolean arrays at each node to check whether the quorum is still satisfied at each step in the construction of models.

The space complexity is $O(N^2 n)$. The term Nn comes from the suffix tree, the second N from the boolean arrays.

4 Structured Models Extraction

4.1 Algorithms for a Known Interval of Distance (Problem 1)

4.1.1 Naive Approach

A naive way of solving Problem 1 consists in extracting and storing all valid single models of length k (given q and e), and then, once this is finished, in verifying which pairs of such models could represent valid structured models (given an interval of distance $[d_{min}, d_{max}]$).

One way of doing this verification profits from the simple observation that two single models m_1 and m_2 may form a structured one only if at least one occurrence of m_1 is at the right distance of at least one occurrence of m_2. Building an array of size nN where cell i contains the list of models having an occurrence starting at that position in $s = s_1 \dots s_N$ allows us to compare models in cell i to models in cells $i + d_{min}, \dots, i + d_{max}$ only. If the sets of occurrences of models are ordered, this comparison may be done in an efficient way (in time proportional to the size of the sets of node-occurrences, which is upper-bounded by nN).

4.1.2 Algorithm 1: Jumping in the Suffix Tree

A first non naive approach to solving the problem starts by extracting single models of length k. Since we are traversing the trie of models in depth-first fashion (also in lexicographic order), models are recursively extracted one by one. At any time, a single model m (and its prefixes) is being considered. Once a valid model m_1 of length k is obtained together with its set of $\mathcal{T}$-node-occurrences V_1 (which are nodes located at level k in $\mathcal{T}$), the extraction of all single models m_2 with which m_1 could form a structured model $((m_1, m_2), (d_{min}, d_{max}))$ starts. This is done with m_2 representing the empty word and having as node-occurrences the set V_2 given by:

$$V_2 = \{(w, e_w = e_v) \mid \exists v \in V_1 \ with \ d_{min} \leq \ level(w) - \ level(v) \leq d_{max}\}$$

where level(v) indicates the level of node v in $\mathcal{T}$. From a node-occurrence v in V_1, a jump is therefore made in $\mathcal{T}$ to **all potential start node-occurrences** w of m_2. These nodes are the d_{min}- to d_{max}-generation descendants of v in $\mathcal{T}$. Exactly the same recurrence formula given in Lemma 3.1 may be applied to the nodes w in V_2 to extract all single models m_2 that, together with m_1 could form a structured model verifying the conditions of the problem, for all valid m_1. An illustration is given in Figure 3 and a pseudo code is presented below. The procedure ExtractModels is called with m the empty word having as sole node-occurrence the root of $\mathcal{T}$ and i equal to 1.

```
procedure ExtractModels(Model m, Block i)
1.  for each node-occurrence v of m do
2.      if i = 2 then
3.          put in PotentialStarts the children w of v at levels k + d_min to k + d_max
4.      else
5.          put v (i.e the root)in PotentialStarts
6.  for each model m_i (and its occurrences) obtained by doing a recursive depth-first
          traversal from the root of the virtual model tree M while simultaneously
          traversing T from the node-occurrences in PotentialStarts (Lemma 3.1 and
          quorum constraint) do
7.      if i = 1 then
8.          ExtractModels(m = m_1, i + 1)
9.      else
10.         report the complete model m = ((m_1, m_2), (d_min, d_max)) as a valid one
```

Since the minimum and maximum length of a structured model (m, d) that may be considered are, respectively, $2k + d_{min}$ and $2k + d_{max}$, we need only to build the tree of suffixes of length $2k + d_{min}$ or more, and for each such suffix to consider at most the first $2k + d_{max}$ symbols. To do this is not difficult. Eliminating suffixes of length less than d_{min} is immediate: during the construction of $\mathcal{T}$ (as described in Gusfield [Gusfield, 1997]), we just do not put such suffixes in $\mathcal{T}$. Placing in $\mathcal{T}$ the prefixes of length at most d_{max} of the suffixes of $\{s_1, \ldots, s_N\}$ means that at each step j during the construction of the tree, nodes whose paths to the root spell a word of greater length need not be considered for extension. The extension process starts therefore with the node v whose path at the end of step $j-1$ spelled a word of length d_{max}. The other nodes to be treated at step j are then obtained by following $\mathcal{T}$'s suffix links from v until the root is reached.

The observation made in the previous paragraph applies also to Algorithm 2 (section 4.1.3). Note that, in both cases, this implies $n_i \leq n_{i+1} \leq Nn$ for all $i \geq 1$ where n_i is the number of nodes at depth i in $\mathcal{T}$.

4.1.3 Algorithm 2: Modifying the Suffix Tree

Algorithm 2 initially proceeds like Algorithm 1: it starts by building single models of length k, one at a time. For each node-occurrence v of a first part m_1 considered in turn, a jump is made in $\mathcal{T}$ down to the descendants of v situated at lower levels. This time however, the algorithm just passes through the nodes at these lower levels, grabs some information the nodes contain and jumps back up to level k again (in a way that will be explained in a short while). The information grabbed in passing is used to temporarily and partially modify $\mathcal{T}$ and start, **from the root of $\mathcal{T}$**, the extraction of the second part m_2 of a potentially valid structured model $((m_1, m_2), (d_{min}, d_{max}))$. Once the operation of extracting all possible companions m_2 for m_1 has ended, that part of $\mathcal{T}$ that was modified is restored to its previous state. The construction of another single model m_1 of a structured model $((m_1, m_2), (d_{min}, d_{max}))$ then follows, and the whole process unwinds in a recursive way until all structured models satisfying the initial conditions are extracted.

More precisely, the operation between the spelling of models m_1 and m_2 locally alterates

$\mathcal{T}$ up to level k to a tree $\mathcal{T}'$ that contains only the k-long prefixes of suffixes of $\{s_1, \ldots, s_N\}$ starting at a position between d_{min} and d_{max} from the end position in s_i of an occurrence of m_1. Tree $\mathcal{T}'$ is, in a sense, the union of all the subtrees t of depth at most k rooted at nodes that represent start occurrences of a potential companion m_2 for m_1.

For each model m_1 obtained, before spelling all possible companions m_2 for m_1, the content of $colors_z$ for all nodes z at level k in $\mathcal{T}$ are stored in an array L of dimension n_k (this is for later restoration of $\mathcal{T}$). Tree $\mathcal{T}'$ is then obtained from $\mathcal{T}$ by considering all nodes w in $\mathcal{T}$ that may be reached on a descent of, this time, $k + d_{min}$ to $k + d_{max}$ arcs down from the node-occurrences (v, e_v) of m_1. These correspond to **all end node-occurrences** (instead of start as in Algorithm 1) of potentially valid models having m_1 as first part. The boolean arrays $colors_w$ for all w indicate to which input strings these occurrences belong. This is the information we grab in passing and take along the only path of suffix-links in $\mathcal{T}$ that leads back to a node z at level k in $\mathcal{T}$. If it is the first time z is reached, $colors_z$ is set equal to $colors_w$, otherwise $colors_w$ is added (boolean "or" operation) to $colors_z$. Once all nodes v and w have been treated, the information contained in the nodes z that were reached during this operation are propagated up the tree from level k to the root (using normal tree arcs) in the following way: if $\bar{z}$ and $\hat{z}$ have same parent z, then $colors_z = colors_{\bar{z}} \cup colors_{\hat{z}}$. Any arc from the root that is not visited at least once in such a traversal up the tree is not part of $\mathcal{T}'$, nor are the subtrees rooted at its end node.

The extraction of all second parts m_2 of a structured model (m, d) follows as for single models in the initial algorithm (Lemma 3.1 in section 3).

Restoring the tree $\mathcal{T}$ as it was before the operations described above requires restoring the value of $colors_z$ preserved in L for all nodes z at level k and propagating the information (state of boolean arrays) from z up to the root.

Since nodes w at level between $2k + d_{min}$ to $2k + d_{max}$ will be solicited for the same operation over and over again, which consists in following the unique suffix-link path from w to a node z at level k in $\mathcal{T}$, $\mathcal{T}$ is pre-treated so that one single link has to be followed from z. Going from w to z takes then constant time.

An illustration is given in Figure 4. A pseudo code for the algorithm is as follows. The procedure ExtractModels is called, as for Algorithm 1, with m the empty word having as sole node-occurrence the root of $\mathcal{T}$ and with i equal to 1.

procedure ExtractModels(Model m, Block i)
1. **for** each node-occurrence v of m **do**
2.   **if** $i = 2$ **then**
3.   put in $PotentialEnds$ the children w at levels $2k + d_{min}$ to $2k + d_{max}$
4.   **for** each node-occurrence w in $PotentialEnds$ **do**
5.   follow fast suffix-link to node z at level k
6.   put z in L
7.   **if** first time z is reached **then**
8.   initialize $colors_z$ with zero
9.   put z in $NextEnds$
10.   add $colors_w$ to $colors_z$
11.   do a depth-first traversal of $\mathcal{T}$ to update the boolean arrays from the root to all z in $NextEnds$ (let $\mathcal{T}'$ be the k-deep tree obtained by such an operation)
12.   **if** $i = 1$ **then**
13.   *Tree* $= \mathcal{T}$
14.   **else**
15.   *Tree* $= \mathcal{T}'$
16. **for** each model m_i (and its occurrences) obtained by doing a recursive depth-first traversal from the root of the virtual model tree $\mathcal{M}$ while simultaneously traversing *Tree* from the root (Lemma 3.1 and quorum constraint) **do**
17.   **if** $i = 1$ **then**
18.   ExtractModels($m = m_1$, $i + 1$)
19.   **else**
20.   report the complete model $m = ((m_1, m_2), (d_{min}, d_{max}))$ as a valid one
21.   restore tree $\mathcal{T}$ to its original state using L

Proposition 4.1 *The following two facts are true:*

- *$\mathcal{T}'$ contains only the k-long prefixes of suffixes of $\{s_1, \ldots, s_N\}$ that start at a position between d_{min} and d_{max} of the end position in $\{s_1, \ldots, s_N\}$ of an occurrence of m_1;*

- *the above algorithm solves Problem 1.*

Proof
The proof is straightforward. We start by observing that, at the end of line 3, *PotentialEnds* indeed contains all end node-occurrences for a potential companion second box to the first one just found. From each one of the nodes w in *PotentialEnds*, it is possible to go up the suffix tree $\mathcal{T}$ and, by the definition of suffix links, this leads us to a node z such that the path from the root to z spells a potential k-long occurrence of a second box. This path may represent words in the sequences that are not occurrences of a second box (because they are not at the right distance). However, these are "eliminated", *i.e.* not counted for verifying the quorum, once boolean arrays along the path are, first initialized, then modified with the boolean array in w. Finally, no potential k-long occurrence of a second box is lost since all end node-occurrences w for a potential companion second box were considered. □

4.2 Complexity

The naive approach to solving Problem 1 requires $nN^2\mathcal{V}(e,k)$ time to find single models that could correspond to either part of a structured model (and $nN\mathcal{V}(e,k)$ space to store all potential parts). If we denote by Δ the value $d_{max} - d_{min} + 1$, finding which pair of single models may be put together to produce a structured model could then be done in time proportional to:

$$\underbrace{\mathcal{V}(e,k)}_{1}\underbrace{\Delta\mathcal{V}(e,k)}_{2}\underbrace{nN}_{3}\underbrace{nN}_{4}$$

where 1. is the maximum number of single models to which a position may belong, 2. is the maximum number of models to which a position at a distance between $k + d_{min}$ and $k + d_{max}$ from the first may belong, 3. is the maximum number of comparisons that must be done to check whether two single models may form a structured one and, finally, 4. is the number of starting positions to consider.

To obtain the complexity of Algorithm 1, we have to calculate the total number of visits we may do to nodes between level $2k + d_{max}$ (the deeper level we ever reach) and the root. To count this, we need to consider, for each node between levels $2k+d_{min}$ and $2k+d_{max}$ in $\mathcal{T}$, how many times it could represent the node-occurrence of a model composed of two boxes, each one having length k and separated by a spacer of length d_{min} to d_{max}. This number is at most:

$$\sum_{i=d_{min}}^{d_{max}} n_{2k+i}\mathcal{V}^2(e,k) \leq \min\{2,\Delta\}n_{2k+d_{max}}\mathcal{V}^2(e,k) \leq \min\{2,\Delta\}n_{2k+d_{max}}k^{2e}|\Sigma|^{2e}$$

where Δ denotes the value $d_{max} - d_{min} + 1$ and $n_{2k+d_{max}}$ is the number of tree nodes at depth $2k + d_{max}$. This last number is never more than nN. The $\min\{2,\Delta\}$ in the bound comes from the fact that the degree of any internal node of $\mathcal{T}$ is at least 2.

Since each visit to a node requires at most $O(N)$ operations, the time complexity of Algorithm 1 is $O(\min\{2,\Delta\}Nn_{2k+d_{max}}\mathcal{V}^2(e,k))$, that is, $O(Nn_{2k+d_{max}}\mathcal{V}^2(e,k))$. Space complexity is $O(N^2n)$ as for the extraction of single models.

In the case of Algorithm 2, we need to consider the number of operations necessary for building the two parts of each model using $\mathcal{T}$ or $\mathcal{T}'$, as well as the number of operations needed to obtain $\mathcal{T}'$ from $\mathcal{T}$ and then to restore back $\mathcal{T}$.

The single models composing either two parts of a structured model may be built in at most $Nn_k\mathcal{V}^2(e,k)$ operations. The reason for this is that, when spelling either part of a model, we are working with nodes between the root and level k only (there are at most $2n_k$ such nodes), and there are $\mathcal{V}^2(e,k)$ ways of spelling two paths from a node at level k to the root (each path corresponding to one part of a structured model) allowing for up to e substitutions in each.

The total number of operations needed to modify the first k levels of the suffix tree $\mathcal{T}$ to obtain $\mathcal{T}'$ before the identification of a second part at a right distance of the first is upper-bounded by:

$$\underbrace{\left(\sum_{i=d_{min}}^{d_{max}} Nn_{2k+i}\mathcal{V}(e,k)\right)}_{\text{visits to nodes } z \text{ coming from } w \text{ for all } m_1} + \underbrace{(Nn_k\mathcal{V}(e,k))}_{\text{propagations from } z \text{ to root for all } m_1} \leq \min\{2,\Delta\}Nn_{2k+d_{max}}\mathcal{V}(e,k).$$

Restoring $\mathcal{T}$ to start the extraction of another structured model from a different first part takes $O(Nn_k\mathcal{V}(e,k))$ operations for all m_2 using $O(Nn_k)$ additional space (size of array L, each cell possibly pointing to a node at level k in $\mathcal{T}$ or to nil). The total time complexity of Algorithm 2 is therefore $O(Nn_k\mathcal{V}^2(e,k) + \min\{2,\Delta\}Nn_{2k+d_{max}}\mathcal{V}(e,k) + Nn_k\mathcal{V}(e,k))$. This results in an $O(Nn_k\mathcal{V}^2(e,k) + Nn_{2k+d_{max}}\mathcal{V}(e,k))$ time complexity. Space complexity is slightly higher than for Algorithm 1: $O(N^2n + Nn_k)$ where $n_k \leq Nn$. The second term is for array L.

In either case, the complexity obtained is better both in terms of time and space than the one given by a naive approach to Problem 1 (see above).

4.3 Extending the Algorithms to Extract Structured Models with $p > 2$ Parts (Problem 2)

4.3.1 Algorithm 1: Jumping in the Suffix Tree

Extending Algorithm 1 to extract structured models composed of $p > 2$ parts, that is solving Problem 2, is immediate. After extracting the first i parts of a structured model $((m_1, \ldots, m_p),(d_{min},d_{max}))$ for $1 \leq i < (p-1)$, one jumps down in the tree $\mathcal{T}$ (following normal tree arcs) to get to the d_{min}- to d_{max}-descendants of every node-occurrence of $((m_1,\ldots,m_i),(d_{min},d_{max}))$ then continues the extraction from there using Lemma 3.1.

A pseudo-code is given below.

procedure ExtractModels(Model m, Block i)
- **for** each node-occurrence v of m **do**
  - **if** $i > 1$ **then**
    - put in *PotentialStarts* the children w of v at levels $(i-1)k + (i-1)d_{min}$ to $(i-1)k + (i-1)d_{max}$
  - **else**
    - put v (the root) in *PotentialStarts*
- **for** each model m_i (and its occurrences) obtained by doing a recursive depth-first traversal from the root of the virtual model tree $\mathcal{M}$ while simultaneously traversing $\mathcal{T}$ from the node-occurrences in *PotentialStarts* (Lemma 3.1 and quorum constraint) **do**
  - **if** $i < p$ **then**
    - ExtractModels($m = m_1 \cdots m_i$, $i+1$)
  - **else**
    - report the complete model $m = ((m_1,\cdots,m_p),(d_{min},d_{max}))$ as a valid one

4.3.2 Algorithm 2: Modifying the Suffix Tree

Extending Algorithm 2 to solve Problem 2 is slightly more complex and thus calls for a few remarks. The operations done to modify the tree between building $m_{i\geq 1}$ and m_{i+1} are almost the same as those described in section 4.1.3 except for two facts. One is that up to $(p-1)$ arrays L are now needed to restore the tree after each modification it undergoes. The second, more important, difference is that we need to keep for each node v_k at level k reached from an ascent up $\mathcal{T}$'s suffix links a list, noted $Lptr_{v_k}$, of pointers to the nodes at lower levels that affected the contents of v_k. The reason for this is that tree $\mathcal{T}$ is modified up to level k only (resulting in tree $\mathcal{T}'$) as these are the only levels concerned by the search for occurrences of each box of a structured model. Lower levels of $\mathcal{T}$ remain unchanged, in particular the boolean arrays at each node below level k. To obtain the correct information concerning the potential end node-occurrences of boxes i for $i > 2$ (*i.e.* to which strings such occurrences belong), we therefore cannot descend $\mathcal{T}$ from the ends of node-occurrences in $\mathcal{T}'$ of box $(i-1)$. If we did, we would not miss any occurrence but we could get more, *e.g.* ones that did not have an occurrence of a previous box in the model. We might thus overcount some strings and consider as valid a model which, in fact, no longer satisfied the quorum. We have to go down $\mathcal{T}$ from the ends of node-occurrences **in** $\mathcal{T}$, that is from the original ends of node-occurrences in $\mathcal{T}$ of the boxes built so far. These are reached from the list of pointers $Lptr_{v_k}$ for the nodes v_k that are identified as occurrences of the box currently just treated. For models composed of p boxes, we need at most $(p-1)$ lists $Lptr_{v_k}$ for each node v_k at level k.

A pseudo-code for the algorithm is as follows:

```
procedure ExtractModels(Model m, Block i)
  for each node-occurrence v of m do
     if i > 2 then
        put in PotentialEnds the children w at levels ik + (i − 1)d_min to ik + (i − 1)d_max
        for each node-occurrence w in PotentialEnds do
           follow fast suffix-link to node z at level k
           put z in L(i)
           if first time z is reached then
              initialize colors_z with zero
              put z in NextEnds
           add colors_w to colors_z
        do a depth-first traversal of T to update the boolean arrays from the root to all
           z in NextEnds (let T′ be the k-deep tree obtained by such an operation)
        if i = 1 then
           Tree = T
        else
           Tree = T′
  for each model m_i (and its occurrences) obtained by doing a recursive depth-first
        traversal from the root of the virtual model tree M while simultaneously
        traversing Tree from the root (Lemma 3.1 and quorum constraint) do
     if i < p then
        ExtractModels(m = m_1 · · · m_i, i + 1)
     else
        report the complete model m = ((m_1, · · · , m_p), (d_min, d_max)) as a valid one
  if i > 1 then
     restore tree T to its original state using L(i)
```

4.3.3 Complexity

Using a same reasoning as before, it is not difficult to see that Algorithm 1 requires $O(N v_{pk+(p-1)d_{max}}$ $\mathcal{V}^p(e,k))$ time, where $\mathcal{V}^p(e,k)) \leq k^{pe}|\Sigma|^{pe}$. The space complexity remains the same as for solving

Problem 1, that is $O(N^2n)$.

In the case of Algorithm 2, the p single models composing a structured model may be built in a number of operations upper bounded by $O(Nn_k\mathcal{V}^p(e,k))$.

The total number of operations needed to modify the first k levels of the suffix tree $\mathcal{T}$ to obtain $\mathcal{T}'$ before the identification of a box $(i+1)$ for $i > 2$ at a right distance of box i is upper-bounded by:

$$(\sum_{j=d_{min}}^{d_{max}} Nn_{ik+(i-1)j}\mathcal{V}^{i-1}(e,k)) + (Nn_k\mathcal{V}(e,k)) \leq \min\{2,\Delta\}Nn_{ik+(i-1)d_{max}}\mathcal{V}^{i-1}(e,k)$$

Restoring $\mathcal{T}'$ as we back off to the preceding box takes, as before, $O(Nn_k\mathcal{V}(e,k))$ operations using $O(N(p-1)n_k)$ additional space (size of arrays $L(1)$ to $L(p)$).

The total time complexity of Algorithm 2 is therefore of $O(Nn_k\mathcal{V}^p(e,k) + Nn_{pk+(p-1)d_{max}}$ $\mathcal{V}^{p-1}(e,k))$. The space complexity is $O(N^2n + N(p-1)n_k)$.

4.4 Extending the Algorithms to Handle Restricted Intervals of Unknown Limits (Problem 3)

4.4.1 Algorithm 1

In the case where the distances between the two parts m_1 and m_2 of a single model vary inside a restricted interval whose limits are unknown, Algorithm 1 is extended in the following way. Once a first part m_1 of a structured model $((m_1, m_2), (d_{min}, d_{max}, \delta))$ has been extracted, we jump as before to nodes w in V_2 given in section 4.1.2. To verify, as we now must, that:

- there exists d, with $d_{min} + \delta \leq d \leq d_{max} - \delta$, such that level($w$) - level($v$) is equal to $d \pm \delta$;
- (more particularly) d is **the same** for pairs of occurrences (one occurrence for each part of the structured model) present in at least q **distinct sequences**;

we just need to keep at each node its distance from level k and to count the number of distinct sequences for each restricted interval $d \pm \delta$ separately.

4.4.2 Algorithm 2

In order to verify the same two points mentioned above in the case of Algorithm 2, we need this time to keep an additional information at the nodes z situated at level k that are reached from w by jumping back up the tree (following suffix links). This information is required because a node at level k may be reached from nodes w corresponding to different distances from occurrences of the previous box. We therefore need to have at each node z an array of dimension not N but $((d_{max} - d_{min} - (2*\delta)) \times N)$. The node-occurrences at each extension step of the second part of a model are added for each cell $i \in (d_{max} - d_{min} - (2*\delta))$ in turn. If for any i, this number is at least q, the model is valid and the second part may be further extended (if its length is still smaller than k).

We denote this boolean array $Colors_z$ with a capital C to stress that it is now multi-dimensional. If it is the first time a node z is reached from w, the l cells of $Colors_z$ for $l \in [\max\{d_{min} + \delta, \text{level}(w){-}\text{level}(z) - \delta\}, \min\{d_{max} - \delta, \text{level}(w){-}\text{level}(z) + \delta\}]$ are set equal to

$colors_w$ and all the other cells are initialized to zero, otherwise $colors_w$ is added (boolean "or") to the l cells of $Colors_z$. Once all nodes v and w have been treated, the information contained in the nodes z that were reached during this operation are propagated up the tree from level k to the root (using normal tree arcs) in the following way: if $\bar{z}$ and $\hat{z}$ have same parent z, then, for all l such that $d_{min} + \delta \leq l \leq d_{max} - \delta$, $Colors_z[l] = Colors_{\bar{z}}[l] \cup Colors_{\hat{z}}[l]$.

4.4.3 Complexity

The time complexity of Algorithm 1 for solving Problem 3 remains $O(Nn_{2k+d_{max}}\mathcal{V}^2(e,k))$ and the space complexity $O(nN^2)$.

The time complexity of Algorithm 2 for solving the same problem becomes $O(N\Delta' n_k \mathcal{V}^2(e,k) + N\Delta' n_{2k+d_{max}}\ \mathcal{V}(e,k))$ where $\Delta' = d_{max} - d_{min} - (2*\delta)$. The space complexity is $O(N^2 n + N\Delta' n_k)$.

4.5 Extending the Algorithms to Extract Structured Models with $p > 2$ Parts and to Handle Restricted Intervals of Unknown Limits (Problem 4)

Few changes are required when one wishes to consider structured models that are composed of more than two boxes separated by intervals of distances of the type $d \pm \delta$ for some d and a fixed δ. The main one concerns Algorithm 2: the boolean arrays at each node in the suffix tree have now to be of dimension $N(p-1)\Delta'$. The Δ' comes from having to handle restricted intervals of unknown limits as we saw in section 4.4. The $(p-1)$ comes from the fact that d may be different for each pair of successive boxes in the structured model. The time and space complexity will therefore be further multiplied by a term of $(p-1)$.

The time complexity of Algorithm 2 (the only one for which there is a change) for solving Problem 4 is $O(N\Delta'(p-1)n_k\mathcal{V}^p(e,k) + N\Delta'(p-1)n_{pk+(p-1)d_{max}}\ \mathcal{V}^{p-1}(e,k))$. The space complexity is $O(N^2 n + N\Delta'(p-1)n_k)$.

4.6 Handling Boxes of Variable Lengths

A straightforward way of handling boxes of variable length requires relatively few modifications. The main one comes from the fact that the operations of jumping (Algorithm 1) or modifying the tree (Algorithm 2) to find a box i once boxes 1 to $i-1$ have been identified is done for all possible allowed lengths (between a minimum and a maximum) for the previous boxes.

4.7 Adding Local and Global Constraints

4.7.1 Global Substitution Rate

One important constraint one may wish to add to the model concerns introducing a global maximum substitution rate e_{global} (this is in addition to the maximum substitution rate e allowed for each box in the model).

No substantial changes need to be done to Algorithm 1 to consider such a global constraint for solving Problem 1. The only difference is that a model has now two substitution counters, one for the current box being built and the other for the global rate.

Algorithm 2 on the other hand requires more important changes for solving the same problem. The main one is that each node v' in $\mathcal{T}'$ will have attached to it a boolean-array that needs this time to be of dimension $(e_{global}+1) \times N$ instead of N. The reason is that in $\mathcal{T}'$ we are grouping nodes from $\mathcal{T}$ that may have accumulated a different number of substitutions against m_1. We denote this boolean array $Colors_{v'}$ with a capital C as before.

If it is the first time z is reached, the e_v cell of $Colors_z$ is set equal to $colors_w$ and the other cells are initialized to zero, otherwise $colors_w$ is added (boolean addition) to the e_v cell of $Colors_z$. Once all nodes v and w have been treated, the information contained in the nodes z that were reached during this operation are propagated up the tree from level k to the root (using normal tree arcs) in the following way: if $\bar{z}$ and $\hat{z}$ have same parent z, then, for $1 \leq j \leq e_{global}$, $Colors_z[j] = Colors_{\bar{z}}[j] \cup Colors_{\hat{z}}[j]$.

Observe that the substitution information concerning a node v' in $\mathcal{T}'$ is obtained from both $Colors_{v'}$ (global error) and $e_{v'}$ (this latter corresponds to the error for one box and is therefore initialized to zero). Apart from having to take into account the fact that $\mathcal{T}'$ groups at each node occurrences in $\{s_1, \ldots, s_N\}$ that may present a different number of substitutions against m_1, the extraction of all second parts m_2 of a structured model (m, d) follows as for single models in the initial algorithm (Lemma 3.1 in section 3).

The time complexity of Algorithm 1 for solving Problem 1 becomes $O(Nn_{2k+d_{max}}\mathcal{V}(e_{global}, 2k))$ if $e_{global} \leq e$. The space complexity does not change.

In Algorithm 2, the building of models takes now time $O(Nn_k\mathcal{V}(e_{global}, 2k))$, again if $e_{global} \leq e$. Obtaining $\mathcal{T}'$ from $\mathcal{T}$ requires $O(Nn_{2k+d_{max}}\mathcal{V}(e, k))$ time. This does not count initialization of the boolean arrays $Colors_z$ in $\mathcal{T}'$ which are done just once for each model m_1 identified and costs $O((e_{global} + 1)n_k\mathcal{V}(e, k))$ operations overall.

Restoring $\mathcal{T}$ to start the extraction of another structured model from a different first part takes $O(Nn_k\mathcal{V}(e, k))$ operations using $O(Nn_k)$ additional space (size of array L, each cell possibly pointing to a node at level k in $\mathcal{T}$ or to nil). The total time complexity of Algorithm 2 for solving Problem 1 is therefore $O(Nn_k\mathcal{V}(e_{global}, 2k) + Nn_{2k+d_{max}}\mathcal{V}(e, k) + (e_{global} + 1)n_k\mathcal{V}(e, k) + Nn_k\mathcal{V}(e, k))$. If we assume N is bigger than e, this results in an $O(Nn_k\mathcal{V}(e_{global}, 2k) + Nn_{2k+d_{max}} \mathcal{V}(e, k))$ time complexity. Space complexity is higher than for Algorithm 1: $O(N^2n + N(e_{global} + 1)n_k + Nn_k)$ where $n_k \leq Nn$. The second term comes from the fact that, in $\mathcal{T}'$, we group together nodes from $\mathcal{T}$ that, as occurrences of models, have accumulated a different number of errors. The third term is for array L.

Observe that, in either case, the complexity obtained improves even more in relation to the one given by a naive approach to Problem 1 which would be in $O(\mathcal{V}(e_{global}, k)\Delta\mathcal{V}(e_{global}, k)nNnN)$ for time and $O(nN\mathcal{V}(e_{global}, k))$ for space.

4.7.2 Other Possible Local and Global Constraints

Another possible local and/or global constraint one may wish to consider for some applications concerns the composition of the boxes.

One may, for instance, determine that the frequency of one or more nucleotide in a box (or among all boxes) be below or above a certain threshold. For structured models composed of more than p boxes, one may also establish that a box i is palindromic in relation to a box j for $1 \leq i < j \leq p$.

In algorithmical terms, the two types of constraints just mentioned are not equivalent. The first type, box composition whether local or global, can in general be verified only *a posteriori* while the second type (palindromic boxes) will result in a, sometimes substantial, pruning of the virtual trie of models.

5 Application to Promoter Consensus Identification from Whole Bacterial Genomes

The aim of the present section is not to show a fully developed computer analysis of promoter sequences with the algorithms described in this paper (see [Vanet *et al.*, 2000] for that) but to illustrate their use for inferring promoter models from a set of bacterial sequences extracted from a whole genome. We address here Problem 1. The algorithms are applied to the identification in *Bacillus subtilis*, *Helicobacter pylori* and *Escherichia coli* of promoter sequences recognized by the RNA polymerase σ^{70} factor (σ^{80} for *H. pylori*).

5.1 Data

The data consisted in three sets of non coding regions located between two divergent genes, that is, between genes transcribed in divergent directions (one on each strand). Each non coding region appears therefore twice in the set, once as a sequence read from the genomic one as publicly released, the other as the same sequence reversed and complemented. This data was extracted from the whole genomes of *B. subtilis* (ftp://ncbi.nlm.nih.gov/genbank/ genomes/bacteria/ Bsub/), *H. pylori* (ftp://ftp.tigr.org/ pub/data/h_pylori) and *E. coli* (http:// mol.genes.nig.ac. jp/ecoli/). The sequences in the three sets (called Gs, Gp and Gc respectively, G denoting "Genomic") are therefore non coding on both strands. Sequences having less than 40 bases were eliminated and only up to 330 nucleotides before the start of translation (as annotated) were initially kept. The first and last 15 bases were then discarded from the sequences in both sets. This eliminated the Shine-Dalgarno sequence as a potential motif. Gs contains 1,062 sequences for a total of 196,736 nucleotides, Gc 1,148 sequences and 226,928 nucleotides while Gp contains 308 sequences and 52,100 nucleotides. The choice of data for this illustration was dictated by the desire to show an application to a whole genome while reducing the amount of noise (non coding sequences containing no promoters). More extensive genomic studies are discussed in [Vanet *et al.*, 2000] and [Vanet *et al.*, 1999].

5.2 Measuring Statistical Significance

Once all structured models satisfying the initial constraints (number of boxes, maximum error rates, distances between boxes and quorum) have been extracted, they may be classified according to their statistical significance. No *a priori* method of evaluating such significance is completely satisfactory for our purposes. This has two main reasons. One is that our models are composed of two parts. The second reason is that, as we allow errors between models and their occurrences, we need methods that are able to either deal with such errors, or handle multiple exact motifs statistics. Those currently available [Régnier and Szpankowski, 1997] [Reinert and Schbath, 1998] [Tompa, 1999] appear too computationally intensive for our purposes. A further complication comes from the fact that we are interested in assessing the statistical significance of the number of occurrences of a model per sequence, *i.e.* of distinct sequences having at least one occurrence, and not of the total number of occurrences.

For these reasons, it seemed more appropriate for now to evaluate the significance of a structured model by using instead a data shuffling approach [Karlin *et al.*, 1989].

Statistical pertinence of the models found was thus evaluated by performing a χ^2 (with one degree of freedom) on two contingency tables, one corresponding to what is observed, the other to what was expected under the null hypothesis [Press *et al.*, 1993], and then determining the probability of getting the models observed given the null hypothesis. A hundred random shufflings preserving both the mono and di-nucleotide frequency distributions of the original sequences was performed to derive the values in the contingency table for the null hypothesis.

Another type of statistics, based on a Z-score, was also tried. It produced a classification of the models that could be considered equivalent: very few permutations in the order obtained were observed.

5.3 Results

The results are presented in the graphics of Figure 5. These graphics plot the intervals of distance allowed between the two parts of a structured model (from 5±1 to 25±1 by increments of one) against the statistical value obtained by the most significant model found in Sets Gs (Fig. 5a), Gp (Fig. 5b) and Gc (Fig. 5c) with up to 1 substitution permitted and a quorum of 4% for *B. subtilis*, 6% for *H. pylori* and 2% for *E. coli* (these correspond to the highest, or close to the highest quorum at which significant models, or any models at all were found).

The most significant model identified for each interval is shown above the curves. This is done only for the intervals located at or near a peak.

Figure 5a shows that the algorithm is able to detect the consensus given in the literature [Helmann, 1995] for the σ^{70} promoter sequence in *B. subtilis*, TTGACAx(17±1)TATAAT, with the highest peaks at the known distance (17±1 bases) between the site at -35 and the TATA-box. The model corresponding to the remaining peak in the curve, with a distance of 7±1 between the two parts, is of unknown function. It is interesting to note that this model is palindromic and that the distance between the two boxes is small. Palindromic motifs separated by such short distances have a chance of corresponding to the binding sites of a dimeric protein. An eample of such a protein is the CRP (Cyclic-AMP Receptor Protein) of which we shall speak below when discussing the models found with the *E. coli* dataset.

This result increases our confidence that the models found for *H. pylori* are also biologically pertinent besides being statistically significant (Figure 5b). The models corresponding to the second peak in particular have been suggested in more detail in [Vanet *et al.*, 2000] to represent the σ^{80} promoter sequence in the bacterium. The models related to the first peak are of unknown function.

Surprisingly, in Set Gc, the consensus given in the literature for the σ^{70} promoter sequence [Record *et al.*, 1996] in *E. coli* (the same as for *B. subtilis*), TTGACAx(17±1)TATAAT, is not identified (Figure 5c). This does not seem to be due to a failure of the algorithms as one structured model is found significant (at an apparently optimal distance of 6±1 between the two parts of the model) that corresponds to a well-kwown palindromic motif, that of the CRP binding site (see above and [Berg and von Hippel, 1988] [Combrugghe *et al.*, 1984] [Lawrence and Reilly, 1990] [Schneider *et al.*, 1986]).

Our feeling that the algorithms are not at fault concerning *E. coli* is reinforced by the fact that the classical consensus, TTGACAx(17±1)TATAAT, is found, although with a weaker conservation than in *B. subtilis*, when the algorithms are run on a set of well-established *E. coli* sequences containing an experimentally determined transcription start or, sometimes, promoter (Figure 5d). We call this set Ec, the E denoting "Experimental". These sequences were obtained from Ozoline [Ozoline *et al.*, 1998]. They are aligned on the predicted transcription start. Although this information is not used, nor needed, by our algorithms, it allows us to verify that the model identified as the most statistically significant at the highest peak (for interval 17±1) corresponds indeed to the promoter sequence although the model itself is slightly different (TTGACTx(17±1)TAAAAT) when compared to the consensus given in the literature. The CRP binding site is not found in Set Ec simply because the sequences in Ec are much shorter than in Gc (their length varies between 60 and 80).

The results concerning the *E. coli* set of non coding sequences between divergent genes are interesting. It has been observed [Mulder *et al.*, 1997] that promoters of gram-positive organisms such as *B. subtilis* exhibit higher consensus requirements than those of *E. coli*. This may explain why it seems much harder to extract a promoter consensus sequence for *E. coli*, indeed impossible with high enough confidence from the set of non coding sequences of the organism (considering all the non coding sequences instead of just those located between divergent genes as in this paper does not change the results), although *E. coli* is believed to have less promoter sequence families than *B. subtilis* (8 as opposed to 18 for *B. subtilis*). This may suggest not only that the σ^{70} promoter family is more degenerate in *E. coli*, but also that it may contain more elements.

A deeper analysis of the algorithms biological interest (including current limitations) may be found in [Vanet *et al.*, 2000] and [Vanet *et al.*, 1999].

Acknowledgements

The authors were partly supported by a CAPES/COFECUB project (of type II, number 272/99) between the universities of Marne-la-Vallée and Rouen in France and of São Paulo and Campinas in Brazil, as well as by the REMAG project with the INRIA, France. They would like to thank O. N. Ozoline for having kindly made available to them Set Ec in ascii format, Anne Vanet from the Institut de Biologie Physico-Chimique, Paris, France, for having suggested and worked on the promoter problem with them, a work that showed the necessity of developing new algorithms, as well as Maxime Crochemore from the Institut Gaspard-Monge, University of Marne-la-Vallée, France, for a very careful reading of the manuscript. Last but not least, they wish to thank all referees of the RECOMB 2000 version of this paper for detailed and very helpful remarks that helped improve the work.

References

[Berg and von Hippel, 1988] O. G. Berg and P. H. von Hippel. Selection of DNA binding sites by regulatory proteins. II. The binding specificity of cyclic AMP receptor protein to recognition sites. *J. Mol. Biol.*, 200:709–723, 1988.

[Bieganski *et al.*, 1994] P. Bieganski, J. Riedl, J. V. Carlis, and E.M. Retzel. Generalized suffix trees for biological sequence data: applications and implementations. In *Proc. of the 27th Hawai Int. Conf. on Systems Sci.*, pages 35–44. IEEE Computer Society Press, 1994.

[Brazma *et al.*, 1998a] A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *J. Comp. Biol.*, 5:279–305, 1998.

[Brazma *et al.*, 1998b] A. Brazma, I. Jonassen, J. Vilo, and E. Ukkonen. Predicting gene regulatory elements *in silico* on a genomic scale. *Genome Research*, 8:1202–1215, 1998.

[Cardon and Stormo, 1992] L. R. Cardon and G. D. Stormo. Expectation Maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments. *J. Mol. Biol.*, 223:139–170, 1992.

[Combrugghe *et al.*, 1984] B. Combrugghe, S. Busby, and H. Buc. Cyclic AMP receptor protein: role in transcription activation. *Science*, 224:831–838, 1984.

[Fraenkel *et al.*, 1995] Y. M. Fraenkel, Y. Mandel, D. Friedberg, and H. Margalit. Identification of common motifs in unaligned DNA sequences: application to *escherichia coli lrp* regulon. *Comput. Appl. Biosci.*, 11:379–387, 1995.

[Galas *et al.*, 1985] D. J. Galas, M. Eggert, and M. S. Waterman. Rigorous pattern-recognition methods for DNA sequences. Analysis of promoter sequences from E*scherichia coli*. *J. Mol. Biol.*, 186:117–128, 1985.

[Gross *et al.*, 1992] C. A. Gross, M. Lonetto, and R. Losick. Bacterial sigma factors. In S. L. Knight and K. R. Yamamoto, editors, *Transcriptional Regulation*, volume 1, pages 129–176. Cold Spring Harbor Laboratory Press, 1992.

[Gusfield, 1997] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[Helmann, 1995] J. D. Helmann. Compilation and analysis of *Bacillus subtilis* α-dependent promoter sequences: evidence for extended contact between RNA polymerase and upstream promoter DNA. *Nucleic Acids Res.*, 23:2351–2360, 1995.

[Karlin *et al.*, 1989] S. Karlin, F. Ost, and B. E. Blaisdell. Patterns in DNA and amino acid sequences and their statistical significance. In M. S. Waterman, editor, *Mathematical Methods for DNA Sequences*, pages 133–158. CRC Press, 1989.

[Klingenhoff *et al.*, 1999] A. Klingenhoff, K. Frech, K. Quandt, and T. Werner. Functional promoter modules can be detected by formal models independent of overall nucleotide sequence similarity. *Bioinformatics 1*, 15:180–186, 1999.

[Lawrence and Reilly, 1990] C. E. Lawrence and A. A. Reilly. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins: struct., funct., and genetics*, 7:41–51, 1990.

[Lewin, 1997] B. Lewin. *Genes VI.* Oxford University Press, 1997.

[McCreight, 1976] E. M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, 23:262–272, 1976.

[Mulder *et al.*, 1997] M. A. Mulder, H. Zappe, and L. M. Steyn. Mycobacterial promoters. *Tuber. Lung Dis.*, 78:211–223, 1997.

[Ozoline *et al.*, 1998] O. N. Ozoline, A. A. Deev, and M. V. Arkhipova. Non-canonical sequence elements in the promoter structure. cluster analysis of promoters recognized by E*scherichia coli* RNA polymerase. *Nucleic Acids Res.*, 25:4703–4709, 1998.

[Press *et al.*, 1993] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing.* Cambridge Univ. Press, 1993.

[Queen *et al.*, 1982] C. Queen, M. N. Wegman, and L. J. Korn. Improvements to a program for DNA analysis: a procedure to find homologies among many sequences. *Nucleic Acids Res.*, 10:449–456, 1982.

[Record *et al.*, 1996] M. T. Record, W. S. Reznikoff, M. L. Craig, K. L. McQuade, and P. J. Schlax. *Escherichia coli* RNA polymerase σ^{70} promoters, and the kinetics of the steps of transcription initiation. In F. C. Neidhardt, editor, *Escherichia coli and Salmonella*, volume 1, pages 792–820. ASM Press, 1996.

[Régnier and Szpankowski, 1997] M. Régnier and W. Szpankowski. On the approximate pattern occurrences in a text. 1997. manuscript.

[Reinert and Schbath, 1998] G. Reinert and S. Schbath. Compound poisson and poisson process approximations for occurrences of multiple words in markov chains. *J. Comp. Biol.*, 5:223–253, 1998.

[Sagot and Viari, 1996] M.-F. Sagot and A. Viari. A double combinatorial approach to discovering patterns in biological sequences. In D. Hirschberg and G. Myers, editors, *Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 186–208. Springer-Verlag, 1996.

[Sagot *et al.*, 1995] M.-F. Sagot, A. Viari, and H. Soldano. A distance-based block searching algorithm. In C. Rawlings, D. Clark, R. Altman, L. Hunter, T. Lengauer, and S. Wodak, editors, *Third International Symposium on Intelligent Systems for Molecular Biology*, pages 322–331, Cambridge, England, 1995. AAAI Press.

[Sagot *et al.*, 1997] M.-F. Sagot, A. Viari, and H. Soldano. Multiple comparison: a peptide matching approach. *Theoret. Comput. Sci.*, 180:115–137, 1997. presented at *Combinatorial Pattern Matching 1995.*

[Sagot, 1998] M.-F. Sagot. Spelling approximate repeated or common motifs using a suffix tree. In C. L. Lucchesi and A. V. Moura, editors, *LATIN'98: Theoretical Informatics*, Lecture Notes in Computer Science, pages 111–127. Springer-Verlag, 1998.

[Schneider *et al.*, 1986] T. D. Schneider, G. D. Stormo, L. Gold, and A. Ehrenfeucht. Information content of binding sites on nucleotide sequences. *J. Mol. Biol.*, 188:415–431, 1986.

[Stormo and Hartzell, 1989] G. D. Stormo and G. W. Hartzell. Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl. Acad. Sci. USA*, 86:1183–1187, 1989.

[Tompa, 1999] M. Tompa. An exact method for finding short motifs in sequences, with application to the ribosome binding site problem. In *Seventh International Symposium on Intelligent Systems for Molecular Biology*, pages 262–271, Heidelberg, Germany, 1999. AAAI Press.

[Ukkonen, 1995] E. Ukkonen. On-line construction of suffix-trees. *Algorithmica*, 14:249–260, 1995.

[van Helden *et al.*, 1998] J. van Helden, B. Andre, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J. Mol. Biol.*, 281:827–842, 1998.

[van Helden *et al.*, 2000] J. van Helden, A. F. Rios, and J. Collado-Vides. Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Res.*, 28:1808–1818, 2000.

[Vanet *et al.*, 1999] A. Vanet, L. Marsan, and M.-F. Sagot. Promoter sequences and algorithmical methods for identifying them. *Research in Microbiology*, 150:1–21, 1999. in press.

[Vanet *et al.*, 2000] A. Vanet, L. Marsan, A. Labigne, and M.-F. Sagot. Inferring regulatory elements from a whole genome. An analysis of the σ^{80} family of promoter signals. *J. Mol. Biol.*, 297:335–353, 2000.

[Werner, 1999] T. Werner. Models for prediction and recognition of eukaryotic promoters. *Mamm. Genome*, 10:168–175, 1999.

[Wolfertstetter *et al.*, 1996] F. Wolfertstetter, K. Frech, G. Herrmann, and T. Werner. Identification of functional elements in unaligned nucleic acid sequences by a novel tuple search algorithms. *Comput. Appl. Biosci.*, 12:71–80, 1996.

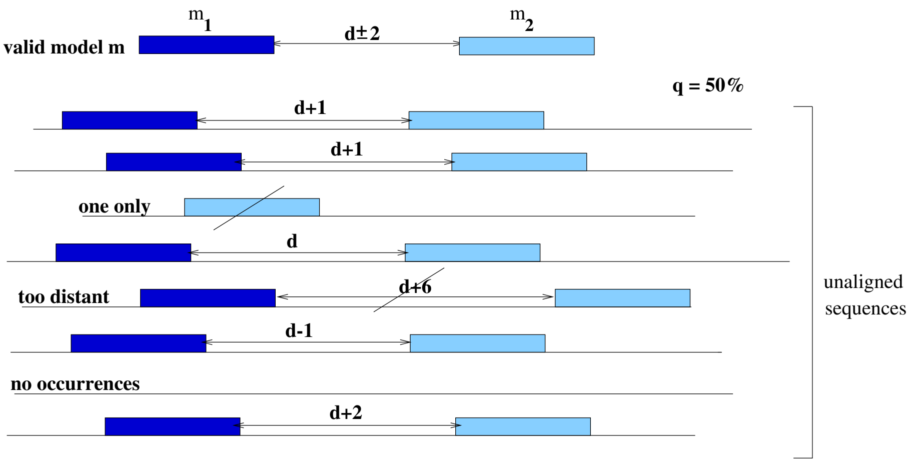

Figure 1: Example of a valid model with two boxes ($p = 2$).

L. Marsan

Legend Figure 1
No legend

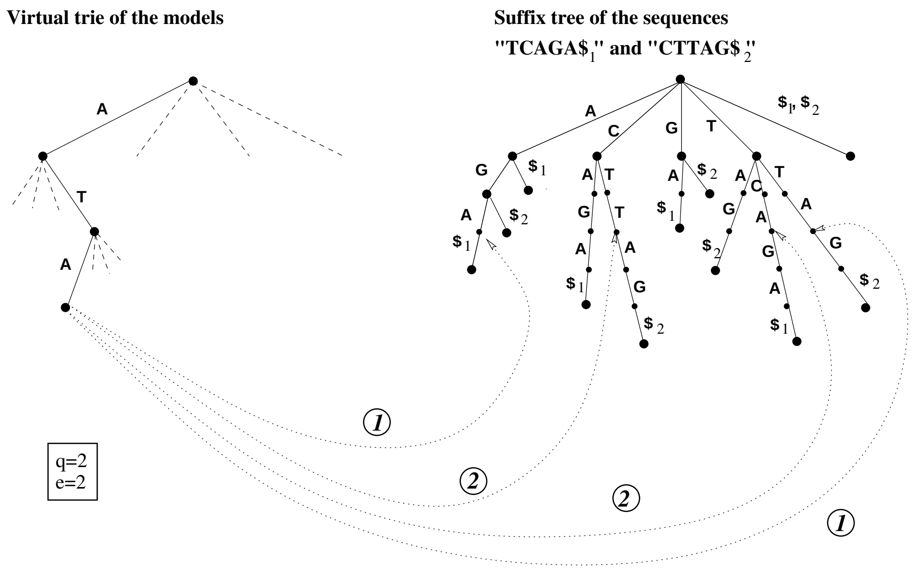

Figure 2: Extracting single models with a suffix tree – An example.

Legend Figure 2

Single models are extracted by simultaneously and recursively traversing (in a depth-first way) the (virtual) lexicographic trie $\mathcal{M}$ of all possible models of length k (stopping the descent down an arc and pruning the subtree rooted at the end node of that arc as soon as the quorum is no longer verified) and the (actually built) suffix tree $\mathcal{T}$ of the sequences. While traversing $\mathcal{T}$, up to e mispellings are allowed of the labels of the arcs in $\mathcal{M}$ corresponding to potentially valid models. In the Figure, model ATA has been spelled. It has suffixes AGA, CTT, TCA and TTA as occurrences with up to two substitutions. It is valid since 2 occurrences belong to string 1 and 2 to the other string (the quorum is 2). To avoid clutering the figure, boolean arrays at each node have not been drawn.

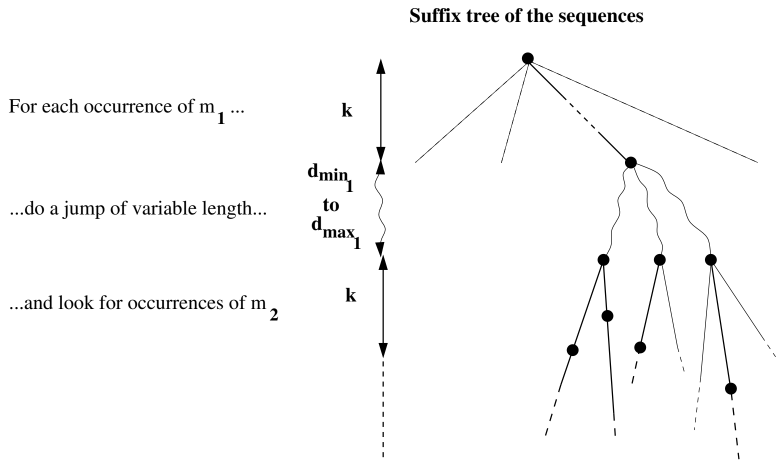

Figure 3: Extracting structured models (in the context of Problem 1) with a suffix tree – An illustration of Algorithm 1.

L. Marsan

Legend Figure 3
No legend

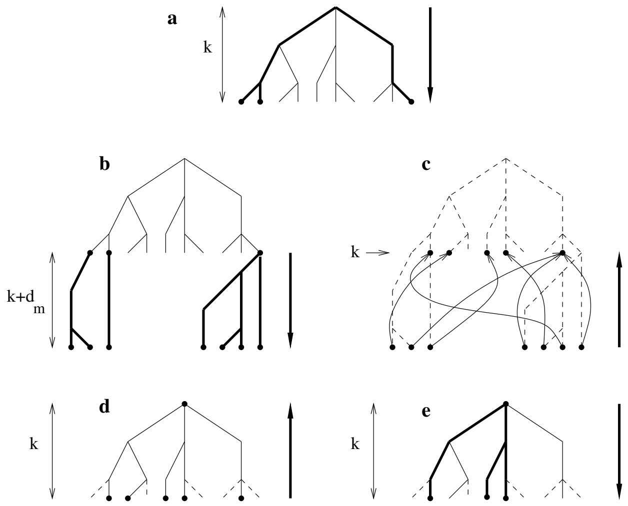

Figure 4: Extracting structured models (in the context of Problem 1) with a suffix tree – An illustration of Algorithm 2.

L. Marsan

Legend Figure 4

Fig. 4a corresponds to the extraction of the first single models m_1 of structure models (m, d); Fig. 4b to the jump of $k+d_{min}$ to $k+d_{max}$ down normal tree arcs to grab some information from the ends of potential node-occurrences for a second box (to lighten the figure, we made here $d_{min} = d_{max} = d_m$); Fig. 4c shows the jump back up to level k following suffix links with the information grabbed in passing; Fig. 4d represents the propagation of the information received at level k up to the root; finally Fig. 4e illustrates the search for second single models m_2 of structure models (m, d) in tree $\mathcal{T}'$.

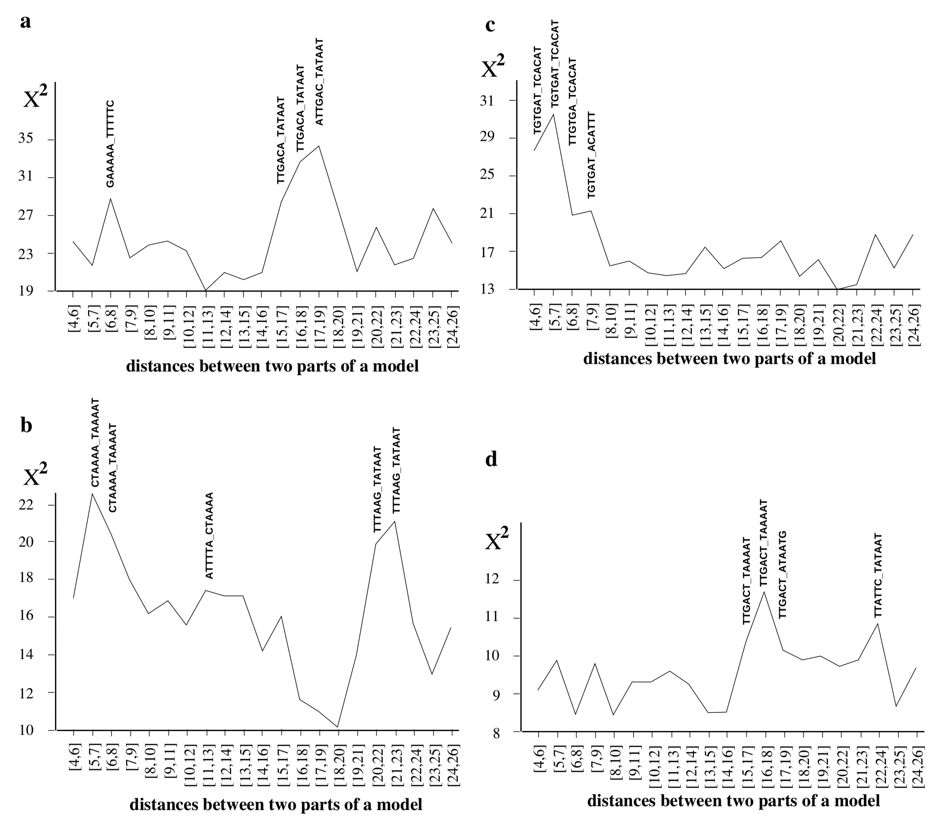

Figure 5: An application of the algorithms (in the context of Problem 1) to the identification of promoter sequences in *B. subtilis*, *H. pylori* and *E. coli*.

Legend Figure 5
The intervals of distance allowed between the two parts of a structured model are plotted against the statistical value obtained by the most significant model found in Sets Gs (Fig. 5a), Gp (Fig. 5b), Gc (Fig. 5c) and Ec (Fig. 5d) with up to 1 substitution permitted and a quorum of 4% for *B. subtilis*, 6% for *H. pylori* and 2% for the *E. coli* sets of sequences. For this, a χ^2 was calculated between the number of occurrences observed (allowing for the same maximum number of errors) in all sets against that observed on average in shuffled versions of each (100 simulations were performed). The most pertinent models identified for each interval at or near a peak is shown above the curves.