Extracting structured motifs using a suffix tree – Algorithms and application to promoter consensus identification

Laurent Marsan [1] Marie-France Sagot [1,2]

[1]Institut Gaspard Monge
Université de Marne la Vallée
2, rue de la Butte Verte
93160 - Noisy le Grand

[2]Institut Pasteur
Service d'Informatique Scientifique
28, rue du Dr. Roux
75324 - Paris Cedex 15

Abstract

This paper introduces two exact algorithms for extracting conserved structured motifs from a set of DNA sequences. Structured motifs are composed of $p \geq 2$ parts separated by constrained spacers. These algorithms use a suffix tree for fulfilling this task. They are efficient enough to be able to extract site consensus, such as promoter sequences, from a whole collection of non coding sequences extracted from a genome. In particular, their time complexity scales linearly with N^2n where n is the average length of the sequences and N their number. An application with interesting results to the identification of promoter consensus sequences in bacterial genomes is shown.

key words: motif extraction, structured motif, promoter, 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, or 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. Up to recently, consensi were therefore often derived from relatively small, well-established and clean datasets [7] [12] [21]. 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 [3] [22] [24] or considered the fact that binding sites come often together in a well-ordered and regularly spaced manner [4] [6] [11] [28]. 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) [13] or because the sites are recognized by proteins that make contact with one another (this is more typical of regulatory proteins in eukaryotes) [27].

This paper presents two exact algorithms that are efficient enough to tackle whole genome site consensus extraction and take into consideration the possible structure of binding sites. They should therefore enable to identify what we shall call *structured motifs* in genomic sequences, that is, motifs composed of $p \geq 2$ parts separated from one another by constrained spacers. 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 factors are N^2n and k^e, where n is the average length of the sequences, N their number, k the maximum length of the motifs (spacers between parts excluded) and e a maximum number of errors allowed.

Section 2 presents definitions and formally states the addressed problem. Section 3 is a quick reminder of the use of a suffix tree for single motif extraction [19]. 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 that are separated by a distance. We then suggest how to extent 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 verify a certain property between themselves). 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.

2.2 From Single to Structured Models

Although the objects defined in the previous section can be reasonable, algorithmically efficient 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 also, for the same reason, commonly composed of two boxes although the distance between them may be different [8].

The case of eukaryotic transcription regulation is more complicated. Promoter sequences contain, as for prokaryotes, two boxes recognized by a same protein, but there may also be other regulatory sites recognized by distinct proteins which interact with one another. The relative positions of these sites along a DNA sequence is thus not always indifferent [27].

There is therefore a need for defining 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 flexible 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 $(m_1, \ldots, m_p)$;
- d is a $(p-1)$-tuple of triplets $((d_{min_1}, d_{max_1}, \delta_1), \ldots, (d_{min_{p-1}}, d_{max_{p-1}}, \delta_{p-1}))$;

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:
  - all these occurrences 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;
- (this is a consequence of the last point) for all $1 \leq i \leq p$, model m_i has at least one occurrence in at least q distinct sequences of the set.

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}).

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

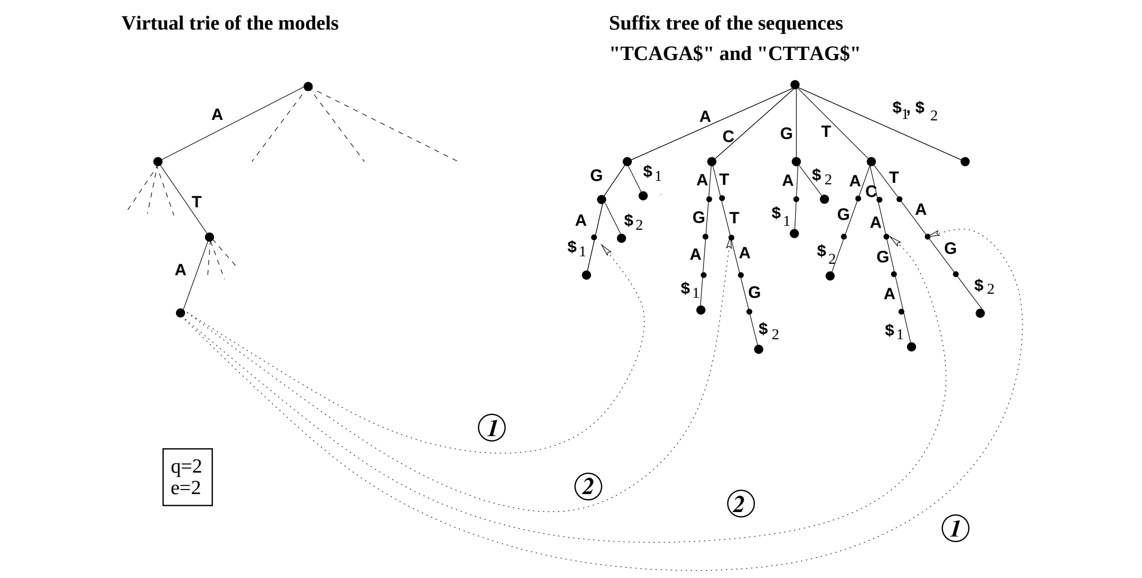

Figure 1: Extracting single models with a suffix tree – An example. Single models are extracted by simultaneously and recursively traversing (in a depth-first way) the (virtual) lexicographic trie $\mathcal{M}$ of all possible models of length k (stopping the descent down an arc and pruning the subtree 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).

single model m_i of the structured model (m, d) is of fixed length k. Generalization to models of a length varying between a minimal and a maximal value, possibly different for each m_i, is straighforward and is not discussed in this extended abstract.

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:

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.

A solution to Problem 1 is described in section 4.1, solutions to Problems 2 and 3 are sketched in sections 4.3 and 4.4 respectively. Solving Problem 4 implies simply putting together the solutions proposed for Problems 2 and 3 and is not discussed here.

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 $s_1, \ldots, s_N$. This is the classical suffix tree introduced by McCreight [14] and modified to consider $N \geq 1$ sequences [2] [9]. 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 b_v of size N indicating the sequences in the set to

which belong the prefixes of suffixes leading to v from the root of $\mathcal{T}$.

The suffix tree construction we adopt is that of Ukkonen [23]. 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 straighforward.

Given a maximum number e of substitutions allowed, it has been shown in [19] 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.

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 on which the algorithm is built is given by the following Lemma where parent(v) denotes the parent of a node v in $\mathcal{T}$.

Lemma 3.1 *[19] A pair (v, e_v) is a node-occurrence of $m' = m\alpha$ with $m \in \Sigma^k$ 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 [19]. See also Figure 1.

Observe that the suffix tree $\mathcal{T}$ is used as another way of representing the sequences and is thus 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 [19], 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:

$$\mathcal{V}(e,k) = \sum_{i=0}^{e} \binom{k}{i} (|\Sigma| - 1)^i \leq k^e |\Sigma|^e$$

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

4 Structured Models Extraction

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

4.1.1 Algorithm 1: Jumping in the Suffix Tree

A first approach to solving the problem starts by extracting single models of length k. Each time a single valid model m_1 is obtained (in lexicographic order) 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 $((m_1, m_2), (d_{min_1}, d_{max_1}))$ starts. This is done with m_2 representing the empty word and having as node-occurrences the set V_2 given by:

$$V_2 = \{(w, e_w = e_v) \mid \exists v \in V_1 \; with \; d_{min_1} \leq \; level(w) - level(v) + 1 \leq d_{max_1}\} \quad (1)$$

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_1}- to d_{max_1}-generation of descendants of v in $\mathcal{T}$. The same recurrence formula given in Lemma 3 is applied to the nodes w 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 2.

Since the minimum and maximum length of a structured model (m, d) that may be considered are, respectively, $2k + d_{min_1}$ and $2k + d_{max_1}$, we need to build only the tree of suffixes of length $2k + d_{min_1}$ or more, and for each such suffix consider at most the first $2k + d_{max_1}$ symbols. To do this is not difficult. This observation applies also to Algorithm 2 below. Note that this implies $v_i \leq v_{i+1} \leq Nn$ for all $i \geq 1$ where v_i is the number of nodes at depth i in $\mathcal{T}$.

4.1.2 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. As each single model m_1 is obtained in turn, a jump is made in $\mathcal{T}$ down to the levels $k + d_{min_1}, \ldots, k + d_{max_1}$ as before. This time however, the algorithm just passes through the nodes at the lower levels, grabs some information the nodes contain and jumps back up to level k again (in a way that will be explained

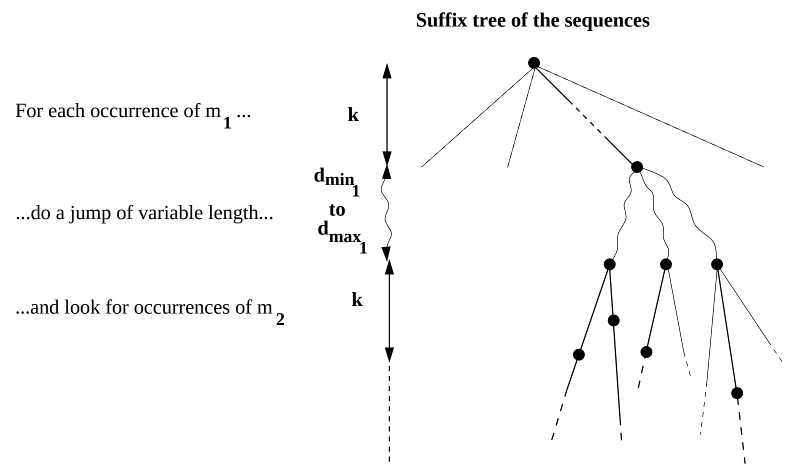

Figure 2: Extracting structured models (in the context of Problem 1) with a suffix tree – An illustration of Algorithm 1.

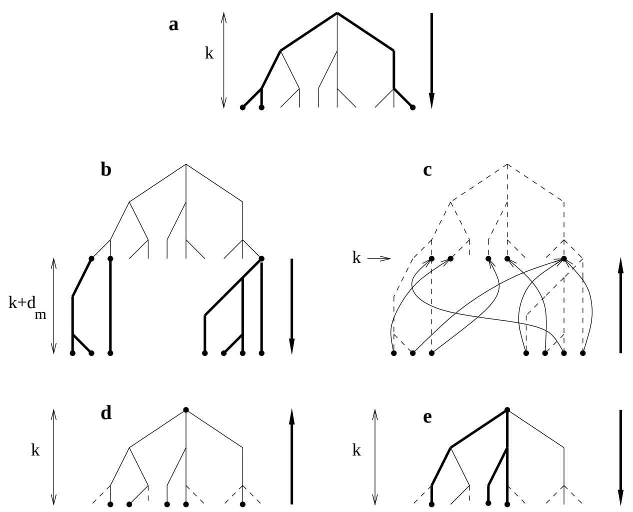

Figure 3: Extracting structured models (in the context of Problem 1) with a suffix tree – An illustration of Algorithm 2. Fig. 3a corresponds to the extraction of the first single models m_1 of structure models (m, d); Fig. 3b to the jump of $k + d_{min}$ to $k + d_{max}$ down normal tree arcs to grab some information (to lighten the figure, we made here $d_{min} = d_{max} = d_m$); Fig. 3c shows the jump back up to level k following suffix links with the information grabbed in passing; Fig. 3d represents the propagation of the information received at level k up to the root; finally Fig. 3e illustrates the search for second single models m_2 of structure models (m, d) in tree $\mathcal{T}'$.

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 $((m_1, m_2), (d_{min_1}, d_{max_1}))$. 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 $((m_1, m_2), (d_{min_1}, d_{max_1}))$ then follows, and the whole process unwinds in a recursive way until all structured models verifying the initial conditions are extracted.

More precisely, the operation between the spelling of models m_1 and m_2 locally alterates $\mathcal{T}$ to a tree $\mathcal{T}'$ that contains only those suffixes of $\{s_1, \ldots, s_N\}$ starting at a position between d_{min_1} and d_{max_1} from the end position in s_i of an occurrence of m_1. $\mathcal{T}'$ is, in a sense, the union of all the subtrees t rooted at nodes w that represent start occurrences of a potential companion m_2 for m_1. The difference is that each node v' in $\mathcal{T}'$ has a boolean-array $b_{v'}$ that will now be of dimension $(e+1) \times N$ instead of N. The reason is that in $\mathcal{T}'$ we are putting together nodes from $\mathcal{T}$ that may have accumulated a different number of substitutions against m_1.

$\mathcal{T}'$ is obtained by considering all nodes w in $\mathcal{T}$ that may be reached on a descent of $k + d_{min_1}$ to $k + d_{max_1}$ arcs down from the node-occurrences (v, e_v) of m_1. From each w, the only path of suffix-links in $\mathcal{T}$ that leads back to a node z at level k in $\mathcal{T}$ is then followed. The address of z and content of b_z are stored in an array L of dimension v_k (this is for later restoration of $\mathcal{T}$). If it is the first time z is reached, the e_v cell of b_z is set equal to b_w and the other cells are initialized to zero, otherwise b_w is added to the e_v cell of b_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 $1 \leq j \leq e$, $b_z[j] = b_{\bar{z}}[j] \cup b_{\hat{z}}[j]$. 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.

An illustration is given in Figure 3.

Observe that the error information concerning a node v' in $\mathcal{T}'$ may now be obtained from $b_{v'}$ and $e_{v'}$ (this latter is 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 (section 3).

Proposition 4.1 *The following two facts are true:*

- *$\mathcal{T}'$ contains only those suffixes of $\{s_1, \ldots, s_N\}$ that start at a position between d_{min_1} and d_{max_1} of the end position in $\{s_1, \ldots, s_N\}$ of an occurrence of m_1;*
- *the above algorithm solves Problem 1.*

Proof Omitted in this extended abstract. □

Restoring the tree $\mathcal{T}$ as it was before the operations described above requires picking each address of a node z that was stored in L, restoring the value of b_z 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 sollicited 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.

4.2 Complexity

To calculate the complexity of Algorithm 1, we need to consider, for each node between levels $2k + d_{min}$ and $2k + d_{max}$ in $\mathcal{T}$, the number of times it could represent the node-occurrence of a model of length k. This number is at most:

$$\sum_{i=d_{min_1}}^{d_{max_1}} v_{2k+i} \mathcal{V}(e, 2k) \leq \Delta v_{2k+d_{max_1}} \mathcal{V}(e, 2k)$$

where Δ denotes the value $d_{max_1} - d_{min_1} + 1$ and $v_{2k+d_{max_1}}$ is the number of tree nodes at depth $2k + d_{max_1}$. As we saw, $v_{2k+d_{max_1}}$ is never more than Nn. The number in the right-hand side of the inequality above is also an upper bound to the total number of visits to nodes in $\mathcal{T}$. Since each visit requires $O(N)$ operations, the time complexity of Algorithm 1 is therefore $O(\Delta N v_{2k+d_{max_1}} \mathcal{V}(e, 2k))$. Space complexity is $O(N^2 n)$ as for the extraction of single models.

In Algorithm 2, the single models composing either two parts of a structured model may be built in at most $N v_k \mathcal{V}(e, 2k)$ operations. The total number of operations needed to modify the suffix tree $\mathcal{T}$ into $\mathcal{T}'$ before the identification of a second part at a right distance of the first is upper-bounded by:

$$\underbrace{\left(\sum_{i=d_{min_1}}^{d_{max_1}} N(e+1) v_{2k+i} \mathcal{V}(e, k)\right)}_{\text{total number of visits to } w \text{ and } z \text{ for all } m_1} +$$

$$\underbrace{(N(e+1) v_k \mathcal{V}(e, k))}_{\text{total number of propagations from } z \text{ to root for all } m_1}$$

$$\leq \Delta N(e+1) v_{2k+d_{max_1}} \mathcal{V}(e, k)$$

Restoring $\mathcal{T}$ to start the extraction of another structured model from a different first part takes $O(N v_k \mathcal{V}(e, k))$

operations using $O(Nv_k)$ 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 is $O(Nv_k\mathcal{V}(e,2k) + \Delta N(e+1)v_{2k+d_{max1}}\mathcal{V}(e,k) + Nv_k\mathcal{V}(e,k))$. Space complexity is higher than for Algorithm 1: $O(N^2n + N(e+1)v_k + Nv_k)$ where $v_k \leq Nn$. The second factor 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 factor is for array L.

The proof of these complexity results is omitted in this extended abstract.

In either case, the complexity obtained is much better than the one given by a naive approach to Problem 1 that would consist in extracting and storing all valid single models of length k (given q and e), and then in verifying, *a posteriori*, which pairs of such models could represent valid structured models (given an interval of distance $[d_{min}, d_{max}]$). Such a naive approach would result in a $O(\Delta(nN\mathcal{V}(e,k))^2)$ time algorithm.

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

4.3.1 Sketch of the Algorithm

Extending Algorithm 1 to extract structured models composed of $p > 2$ parts (always considering, to simplify matters, that each part has a fixed length k), that is solving Problem 2, is immediate. After extracting the first $2 \leq i < p$ parts of a structured model $((m_1, \ldots, m_p), ((d_{min_1}, d_{max_1}), \ldots, (d_{min_{p-1}}, d_{max_{p-1}})))$, one jumps down in the tree $\mathcal{T}$ (following normal tree arcs) to get to the d_{min_i}- to d_{max_i}-descendants of a node-occurrence of $((m_1, \ldots, m_i), ((d_{min_1}, d_{max_1}), \ldots, (d_{min_{i-1}}, d_{max_{i-1}})))$ (which inherit the error rate from their ancestor) then continues the extraction from there.

Extending Algorithm 2 to solve Problem 2 is equally straightforward. The operations done to modify the tree between building $m_{i \geq 1}$ and m_{i+1} are the same as those described in section 4.1.2 except that what is propagated up to level k and then, from that level, up the tree to the root are boolean arrays that have already a dimension $(e+1) \times N$ instead of N if $i > 2$. Furthermore, up to $(p-1)$ arrays L may be needed to restore the tree after each modification it undergoes.

4.3.2 Sketch of Complexity

Let us denote d_{min} and d_{max} the average of $d_{min_{1 \leq i \leq p}}$ and $d_{max_{1 \leq i \leq p}}$ respectively.

Using a same reasoning as before, it is not too difficult to see that Algorithm 1 requires time proportional to $\Delta N v_{pk+(p-1)d_{max}}\mathcal{V}(e,pk)$. The space complexity remains the same as for solving Problem 1.

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 $Nv_k\mathcal{V}(e,pk)$. The total number of tree-modification operations is upper-bounded by:

$$\left(\sum_{i=d_{min}}^{d_{max}} N(e+1)v_{2k+i}\mathcal{V}(e,k)\right) + (N(e+1)v_k\mathcal{V}(e,k)) \\ \leq \Delta N(e+1)v_{pk+(p-1)d_{max}}\mathcal{V}(e,k)$$

The total time complexity of Algorithm 2 is therefore of $O(Nv_k\mathcal{V}(e,pk) + \Delta N(e+1)v_{pk+(p-1)d_{max}}\mathcal{V}(e,k))$.

The space complexity is $O(N^2n + N(e+1)v_k + N(e+1)(p-1)v_k)$.

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

4.4.1 Sketch of the Algorithm

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 $((m_1, m_2), (d_{min_1}, d_{max_1}, \delta_1))$ has been extracted, we jump as before to nodes w in V_2 given by equation (1) in section 4.1.1. This time however, we must know exactly at what level is located each node w in relation to v. This information is required to verify that:

- there exists d_1, with $d_{min_1} + \delta_1 \leq d_1 \leq d_{max_1} - \delta_1$, such that level($w$) - level($v$) + 1 is equal to $d_1 \pm \delta_1$;
- (more particularly) d_1 is **the same** for pairs of occurrences (one occurrence for each part of the structured model) present in at least q **distinct sequences**.

We therefore want to be able to count the number of distinct sequences for each restricted interval $d_1 \pm \delta_1$ separately. To be able to do this, we need to have at each node w an array of dimension not N but $((d_{max_1} - d_{min_1} - (2 * \delta_1)) \times N)$. The node-occurrences at each extension step of the second part of a model are added for each cell $i \in (d_{max_1} - d_{min_1} - (2 * \delta_1))$ 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).

The same additional information as described above is needed to solve Problem 3 using an extension of Algorithm 2. This information concerns this time however the nodes z at level k reached from w by jumping back up the tree (following suffix links). For this, we need to have at each node z an array of dimension $((d_{max_1} - d_{min_1} - (2 * \delta_1)) \times (e+1) \times N)$. If it is the first time a node z is reached from w, the (i, e_v) cells of b_z for $i \in [\max\{d_{min_1} + \delta_1$, level($w$)−level($z$) + 1 −

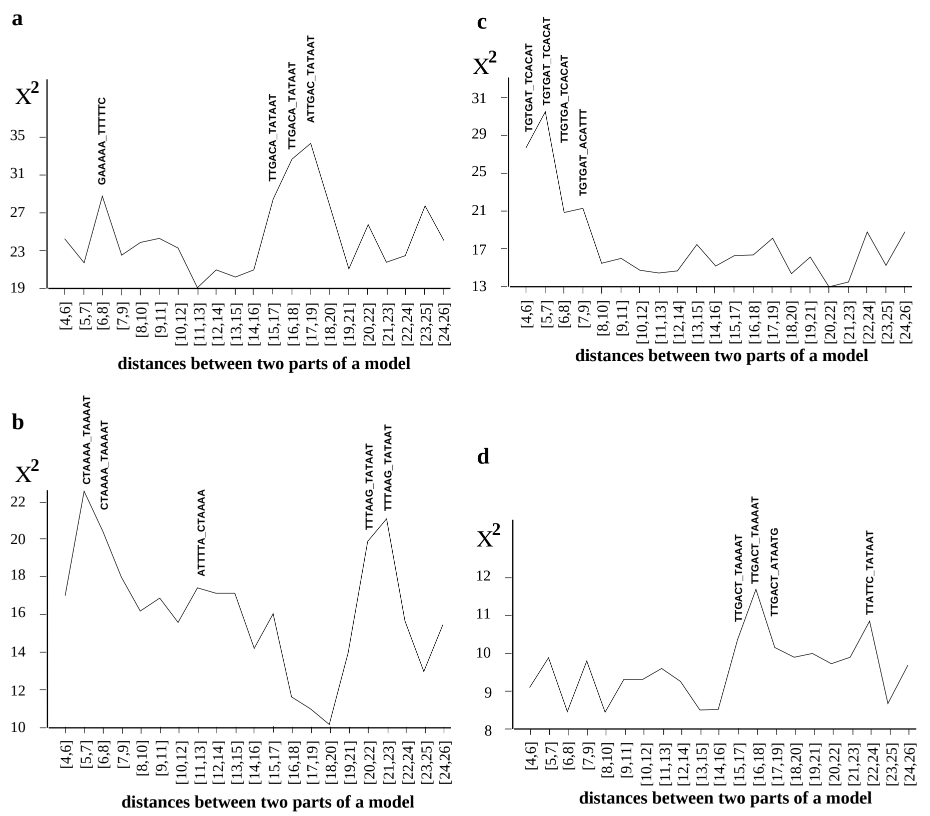

Figure 4: An application of the algorithms (in the context of Problem 1) to the identification of promoter sequences in *B. subtilis* and *H. pylori*, *E. coli*: Intervals of distance allowed between the two parts of a structured model plotted against the statistical value obtained by the most significant model found in Sets Gs (Fig. 4a), Gp (Fig. 4b), Gc (Fig. 4c) and Ec (Fig. 4d) with up to 1 substitution permitted and a quorum of 4% for *B. subtilis*, 6% for *H. pylori* and 2% for *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.

$\delta_1\}$,min$\{d_{max_1} - \delta_1$, level$(w)-$level$(z) + 1 + \delta_1\}]$ is set equal to b_w and all the other cells are initialized to zero, otherwise b_w is added to the (i, e_v) cells of b_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 $d_{min_1}+\delta_1 \leq l \leq d_{max_1}-\delta_1$ and $1 \leq j \leq e$, $z_b[l][j] = \bar{z}_b[l][j] \cup \hat{z}_b[l][j]$.

4.4.2 Sketch of Complexity

The time complexity of Algorithm 1 for solving Problem 3 becomes $O(\Delta\Delta' N v_{2k+d_{max_1}} \mathcal{V}(e, 2k))$ where $\Delta' = d_{max_1} - d_{min_1} - (2*\delta_1)$. The space complexity is $O(nN^2 + \Delta\Delta' N v_{2k+d_{max_1}})$.

The time complexity of Algorithm 2 for solving the same problem is $O(\Delta' N v_k \mathcal{V}(e, 2k) + \Delta\Delta' N(e+1) v_{2k+d_{max_1}} \mathcal{V}(e, k))$. The space complexity is $O(N^2 n + \Delta' N(e+1) v_k + \Delta' N v_k)$.

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 here (see [25] for that) but to illustrate their use for extracting promoter models from a set of bacterial sequences extracted from a whole genome. The problem we are solving here is 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. 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 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 [25] and [26].

5.2 Results

The results are presented in the graphics of Figure 4. 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. 4a), Gp (Fig. 4b) and Gc (Fig. 4c) 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 quori at which significant models, or any models at all were found)[1]. 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.

[1]Statistical pertinence of the models found was 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 [17], 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.

Figure 4a shows that the algorithm is able to detect the consensus given in the literature [10] 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.

This result increases our confidence that the models found for *H. pylori* are also biologically pertinent besides being statistically significant (Figure 4b). The models corresponding to the second peak in particular have been suggested in more detail in [25] 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 [18] in *E. coli* (the same as for *B. subtilis*), TTGACAx(17±1)TATAAT, is not identified (Figure 4c). 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 motif believed to be palindromic, that of the CRP binding site [1] [5] [12] [20].

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 when the algorithms are run on a set of well-established *E. coli* sequences containing an experimentally determined transcription start or, sometimes, promoter (Figure 4d). We call this set Ec, the E denoting "Experimental". These sequences were obtained from Ozoline [16]. 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 [15] 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), especially that *E. coli* is believed to have less promoter sequence families than *B. subtilis* (8 as opposed to 18 for *E. coli*). 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 behaviour (including current limitations) and biological interest may be found in [25] and [26].

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. Last but not least, they wish to thank 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.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

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

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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. 1999. submitted to *J. Mol. Biol.*

[26] 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.

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

[28] 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.