Suffix Tree Characterization of Maximal Motifs in Biological Sequences

Maria Federico[1,3] and Nadia Pisanti[2,3]

[1] Dip. di Ingegneria dell'Informazione, Univ.di Modena e Reggio Emilia, Italy
[2] Dip. di Informatica, Univ. di Pisa, Italy
[3] Supported by the MIUR project MAIN STREAM

Abstract. Finding motifs in biological sequences is one of the most intriguing problems for string algorithms designers as it is necessary to deal with approximations and this complicates the problem. Existing algorithms run in time linear with the input size. Nevertheless, the output size can be very large due to the approximation. This makes the output often unreadable, next to slowing down the inference itself. Since only a subset of the motifs, i.e. the *maximal* motifs, could be enough to give the information of all of them, in this paper, we aim at removing such redundancy. We define notions of maximality that we characterize in the suffix tree data structure. Given that this is used by a whole class of motifs extraction tools, we show how these tools can be modified to include the maximality requirement on the fly without changing the asymptotical complexity.

Keywords: Suffix trees, Maximal Motifs, Biological Sequences.

1 Introduction

Finding frequent patterns (motifs) in biological sequences has myriads of applications in molecular biology. Following the hypothesis that sequence similarity is often a necessary condition for function correlations, there have been suggested in literature many versions, for as many various biological applications, of the problem of finding motifs as particularly frequent patterns in a biological sequence, or as patterns surprisingly shared by several distinct sequences. The motifs search is *approximated*, that is, distinct occurrences of the same motif are not necessarily identical, but just *similar*, according to a given similarity notion. From the computational complexity point of view, this makes the task of finding over-represented patterns harder, whatever is the type of approximation one uses. The *Hamming distance* is defined between patterns of the same length and it simply consists of the number of differences that occur between them. One usually sets a maximum allowed distance and then requires that the motifs differ by at most that number of letters substitutions.

Finding approximate motifs is a computationally challenging task because the output itself can be very big, especially with the approximation: its size can be exponential with respect to a parameter that measures the approximation

M. Elloumi et al. (Eds.): BIRD 2008, CCIS 13, pp. 456–465, 2008.

(the maximum distance, the degeneracy degree of the degenerated alphabet, the number of don't care symbols used, etc.). This is a big drawback that, next to make the inference task possibly too slow, often also leads to a poor usability of the results as they are too large to be investigated with a naked eye. The difficulty to make use of the results of some motifs finding tools is often due to the fact that there are many motifs that satisfy the requirements, while only some of them are significant or, more in general, only some of them contain enough information to actually represent all the others. In this paper, we aim at eliminating most of the redundancy that make unreadable the output of existing methods that find approximate motifs. We define some notions of maximality for exact and approximate motifs and give for all of them a characterization on the suffix tree data structure. This allows us to show how to adapt a whole class of algorithms based on suffix tree for which available tools exist, to infer maximal motifs only, without additional complexity. We thank Esko Ukkonen for sending us a copy of [11].

2 Preliminary Definitions

We consider strings that are finite sequences of characters drawn from an alphabet Σ. In particular we will focus out attention on the DNA alphabet $\Sigma = \{A,\ C,\ G,\ T\}$. We denote by $s[i]$ the character at position i in a string s and by $|s|$ the length of s. Consecutive characters of s form a *substring* of s. The substring of s that starts from position i and ends at position j is denoted by $s[i..j]$, where $1 \leq i \leq j \leq |s|$. Given a string x drawn from the same alphabet as s (or from one of its subsets), we say that $s[i..j]$ *exactly occurs* at position i in s if and only if $x = s[i, i + |x| - 1]$. In this case, we also say that $s[i, i + |x| - 1]$ is an *occurrence* of x in s. The Hamming distance between two strings x and y, denoted as $d_H(x, y)$, is the smallest number of letter substitutions that transform x into y (or vice versa as the distance is symmetric). Given an integer $e \geq 0$, we say that a substring y of a string s is an *e-occurrence* of a string x, if and only if $d_H(x, y) \leq e$. In this case we will also talk about an *approximate occurrence*, or simply an *occurrence*, of x in s. The list of all occurences of a pattern x in s is denoted by $L_{(e,x)}$ and is called *positions set*.

Definition 1. *Given a sequence s, a* quorum *$q \geq 2$, and $e \geq 0$, a pattern m is a* motif *iff $|L_{(e,m)}| \geq q$.*

If $e = 0$ we speak about *exact motifs*, because no differences between motifs and their occurrences are allowed; otherwise, when $e > 0$, we call them *approximate motifs*. The traditional motifs extraction problem gives as input: (i) the string in which one wants to find the repeated motif (or the set of strings in which one wants to find the common motif); (ii) the quorum; (iii) the (minimal) length ℓ required for the motif; (iv) optionally, an approximation measure (e.g. the Hamming distance), and the value of e for the approximation measure. The requested output is simply the set of all patterns of length (at least) ℓ that have at least q (possibly approximated) occurrences in s, that is, the complete set

of motifs. Within this traditional framework, the output can be very noisy as it contains redundant data. In this paper, we suggest a way to overcome this drawback by introducing a notion of maximality for motifs, thus identifying a subset of interesting representatives, and an efficient way to detect directly only those. To this purpose, we first introduce the notion of length extension of a motif. With *left extension* (resp. *right extension*) of m, we mean a pattern m' obtained by the concatenation of m with characters at its left (resp. right), and so such that m is a substring of m'. If there exists a right or left extension m' of a motif m which is also a motif, then we will say that m is *included* in m'.

Definition 2. *Let m and α be patterns of s. The pattern $m' = m\alpha$ (resp. $m' = \alpha m$) is a* mandatory right *(resp.* mandatory left*)* extension *of m iff all the occurrences of m in s are followed (resp. preceded) by α. In this case, we call $k = |\alpha|$ the* degree of the extension.

Definition 2 above is the same for both exact and approximate motifs. In the latter case, if m' is a mandatory right/left extension of m, then the two motifs m and m' have the same number of occurrences and also the total number of letters mismatches is the same because the right/left extension of m does not introduce further substitutions between the motif and its occurrences. It follows that if m is a motif, then also m' is a motif and vice versa. We will name *mandatory extension* (without specifying whether it is left or right) an extension on possibly both sides. We can observe that for a motif there exists at most one left and one right mandatory extension with a certain degree d. It is intuitive to observe that if the occurrences of a motif m are not all followed (resp. preceded) by the same character, then there can not be a mandatory right (resp. left) extension of degree 1 of m, and hence neither a mandatory right (resp. left) extension of higher degree can exist.

Notions of motif maximality have been defined in [3] for exact motifs. Furthermore, there have been notions of maximality defined for approximate motifs when the approximation is achieved using a degenerate alphabet, the edit distance, and don't care symbols. We give here a notion of maximality for approximate motifs with Hamming distance. Other (different) maximality notions for this type of approximate motifs exist in literature but are not meant for the general case as ours. In [4] the notion is restricted to the case of tandem repeats. The notion of maximality for motifs approximated with Hamming distance given in [5] does not apply to the whole occurrences set of the motif, but rather only for the special case of *repeats*, that are pairs of occurrences.

Definition 3. *A motif m is* right *(resp.* left*)* maximal *iff it has no mandatory right (resp. left) extension of degree 1. A motif m is* maximal *iff it is both right maximal and left maximal.*

For the particular case of exact motifs and $q = 2$, this notion of maximality coincides with that already introduced in [3]. This maximality property may not be enough to significantly bound the number of motifs. It can thus be useful to use an even more strict notion of maximality in extension of a motif (also already introduced in [3] for the special case of $q = 2$ and exact motifs). Moreover, in

some applications long patterns with few occurrences can be more interesting of short patterns with a lot of occurrences. For these reasons, we formulate the notion of *supermaximality*[1].

Definition 4. *A motif m is* supermaximal *iff it is not a substring of another motif.*

In other words, a motif is supermaximal if, as one tries to extend it in any way, then the quorum property does not hold anymore. Note that the supermaximal motifs are a subset of the maximal ones.

3 Characterization of Motif Maximality on Suffix Tree

In this section we will give a characterization of maximal and supermaximal motifs of a string s on the suffix tree of the string itself. We will do this both for exact and for approximate motifs. For a formal definition of the suffix tree and its properties, we remind to [3]. We just recall here that it is a tree data structure that indexes a text such that there is a root-leaf path per each suffix of the text, and thus each root-node path for a node u corresponds to a substring of the text, to which we will refer to *the word spelled by u*, or *path-label of u*. Given that for some substrings the path does not end at a node but rather inside an edge, we will also talk about *the word spelled by a path*.

For exact motifs, in [3] there is already a characterization of maximal and supermaximal motifs on suffix trees for the special case in which $q = 2$, that is, when any pattern occurring two times or more is a motif. For the purpose of the suffix tree characterization, setting the quorum equal to two simplifies the task because every internal node corresponds to a pattern that satisfies the quorum and thus it is a motif. In this section, we first describe Gusfiled's result and then show the trivial generalization to the case of $q \geq 2$ that involves a search in a specific area of the suffix tree of the input string. Let s be a sequence and T its suffix tree. It is known that T can be built in linear time and space ([7,10]). Let us start with observing that an exact motif m, not necessarily maximal, labels a single path on suffix tree from the root that can end at an internal node or inside an edge having an internal node as destination. Using the mildest possible repetitiveness constraint ($quorum = 2$) Gusfield in [3] showed that there exists linear time algorithm based on suffix tree to find all maximal and supermaximal motifs. We now briefly describe Gusfield's result for maximal exact motifs. Each internal node of T, has at least two children and the edges out of an internal node are labeled with nonempty substrings of s that start with different characters. Therefore, if m labels an internal node it means that there are at least two occurrences of m in s followed by different characters, and hence m is right maximal. On the contrary, if m labels a path which ends inside an edge, then all its occurrences are followed by the character at depth $(|m| + 1)$ along that edge in T, and hence m has a mandatory right extension and thus it is not maximal.

[1] Being our definition the natural extension of that in [3] to the case of approximate motifs, we keep the same name.

The *left character of a leaf* of T is the character preceding the suffix of s at the position (where starts the suffix) represented by that leaf. A node v is *left diverse* if at least two leaves in the subtree rooted at v have different left characters, so if m labels a left diverse node it means that there are at least two occurrences of m in s preceded by different characters. Recall that, by definition, a leaf cannot be left diverse ([3]). Gusfield indeed proves that maximal motifs label exactly paths on the suffix tree that start from the root and lead to left diverse nodes of T. Let us now show how to extend the idea to the case of maximal exact motifs m occurring at least $q \geq 2$ of times in the input sequence. In the suffix tree T, for any internal node v that spells the pattern m, the positions of occurrences of m in the input sequence are represented by the (starting positions of the suffixes that label the) leaves in the subtree rooted at v. For each internal node v, let $Lv[v]$ denote the number of leaves in the subtree rooted at v. If v is a leaf, then we set $Lv[v] = 1$. It is possible to annotate all the internal nodes of T with the value of $Lv[v]$ within the linear time complexity by a simple traversal of the suffix tree (as shown in [9]). A pattern is a motif if it labels a path on the suffix tree that ends to an internal node v such that $Lv[v] \geq q$, or inside an edge that ends at an internal node for which this is the case. A motif is right maximal if it labels a path on T that ends at an internal node, and it is left maximal if such node is left diverse. Summing up, an exact motif is maximal if and only if it labels an internal node v of T such that $Lv[v] \geq q$ and v is left diverse, because right and left mandatory extensions of degree 1 can not exist for m. Gusfield also provides a characterization on suffix tree of supermaximal exact motifs, for the special case in which $q = 2$. If an exact motif m labels a path ending inside an edge, then it is not supermaximal either, because it has a right extension of degree 1 which is a motif preserving the same occurrences of m. Gusfield furthermore proves that supermaximal exact motifs label internal nodes v of T such that all the children of v are leaves and each has a distinct left character. In such case, indeed, a motif can not be furtherly extended without breaking the quorum constraint, because none of its extensions has two occurrences. This idea can be extended to the case of supermaximal exact motifs occurring at least $q \geq 2$ times in the input sequence. For this purpose, we introduce the notion of *right* and *left q-limited node.*

Definition 5. *Let T be the suffix tree for the sequence s. A node v of T that spells a pattern m is* right q-limited *(resp.* left q-limited*) iff m has no right (resp. left) extension of degree 1 which occurs at least q times in s. We say that a node is* q-limited *if it is right q-limited and left q-limited.*

Note that *not* being right q-limited (or left q-limited) is a property that propagates upward: if an internal node v is not right (resp. left) q-limited, then neither is any of its ancestors in the tree. Clearly, if $Lv[v] < q$ then v is q-limited. A right extension of degree 1 of m can label a child of v or a path ending inside an edge with a child of v as destination. It follows that v is right q-limited if and only if its children are nodes v' such that $Lv[v'] < q$.

The characterization on the suffix tree of the left q-limited property is a bit less immediate and it involves the so-called suffix link [3], that is a pointer that

connects a node v spelling ax (with $a \in \Sigma$ and $x \in \Sigma^*$) to the node u that spells x. In other words, this pointer provides a link from node v to node u such that v spells a left extension of degree 1 of the path-label of u. Note that each pattern x has at most $|\Sigma|$ left extensions of degree 1 and thus a node u that spells x is reached by at most as many suffix links. Moreover, the left extensions of x do not always label nodes, but they can also label paths ending inside edges. Node u is reached by as many suffix links as the number of left extensions of degree 1 of x which label a node. If there exists a left extension which labels a path ending inside an edge that leads to node v', then there exists a suffix link from v' to a descendant of u whose left extension of degree 1 is the path-label of v'. It follows that if all the left extensions of degree 1 of the pattern spelled by a node label paths ending inside edges, then this node is not reached by any suffix link. In particular, we can observe that an internal node u, labeled by x, is reached by a suffix link only if at least two of its children are such that both their path-labels are preceded by the same character α at some (possibly all) of their occurrence positions in the input sequence. Moreover, there exists only one suffix link directed to the leaf node representing the suffix at position i in the input sequence s and it starts from the leaf representing the suffix at position $i-1$ in s.

Due to what we just showed about suffix links and to the fact that not being left q-limited is a property that propagates upward in the tree, we observe that a node v is left q-limited only if $Lv[u] < q$ holds for each node u from which a suffix link to v originates and its children are left q-limited nodes.

Theorem 1. *Given a quorum q, a sequence s and its suffix tree T, the pattern m labeling the path to a node v of T is a supermaximal exact motif iff $Lv[v] \geq q$ and v is q-limited.*

Considering that the distinct occurrences of an approximate motif label different paths on suffix tree, the characterization on suffix tree provided for maximal exact motifs can be simply extended to approximate motifs.

Theorem 2. *Let T be the suffix tree for string s. An approximate motif m is right maximal iff:*

1. *at least an occurrence-path of m labels a node of T, or*
2. *all the occurrence-paths of m end inside edges of T and at least two of them have different characters at depth $(|m|+1)$.*

Reminding that if a node of T is not left diverse then all the leaves in its subtree have the same left character, we give a characterization also for the left maximality of approximate motifs.

Theorem 3. *Let T be the suffix tree for a string s. An approximate motif m is left maximal iff:*

1. *at least an occurrence-path of m labels a left diverse node or ends inside an edge ending at a left diverse node, or*

2. *there are at least two distinct occurrence-paths of m ending (inside an edge ending) at a node that is not left diverse, and such that the leaves reached following these paths have not the same left character.*

Let us now show the characterization on suffix tree of supermaximal approximate motifs. In the case of exact motifs we used the notion of right and left q-limited node to verify supermaximality of a motif. When we consider approximate motifs, this notion does not suffice because they could have multiple occurrence-paths. Nevertheless, we can observe that if there exists an occurrence-path of an approximate motif m ending at an internal node which is not q-limited or inside an edge with a destination node that is not q-limited, then m is not supermaximal. In order to check whether an approximate motif can be extended keeping on satisfying the quorum constraint, one must take into account, per each occurrence, the number of mismatches with the motif. In details, given the mismatches threshold e, let x be an occurrence of a motif m with positions set $L_{(e,x)} = \{p_{x_1}, \ldots, p_{x_h}\}$ and such that $d_H(m, x) = d$. The positions set of the right (resp. left) extension $m\alpha$ (resp. αm) of m with any character $\alpha \in \Sigma$ includes positions p_{x_i} (resp. $p_{x_i} - 1$) where x is followed (resp. preceded) by:

- α, if $d = e$; the other occurrences are lost because the mismatches threshold e is exceeded, or
- any $\beta \in \Sigma$, if $d < e$; if $\beta \neq \alpha$, an extra mismatch between the extension $m' = m\alpha$ (resp. αm) and its occurrences $x\beta$ (resp. βx) is introduced.

If x labels a path ending inside an edge (u, v), or at an internal node u and let v be any child of u, then $m' = m\alpha$ has an occurrence-path ending inside (u, v) or at v, only if the character at string-depth $(|m| + 1)$ along (u, v) in T is α, or if it is $\beta \neq \alpha$ and $d < e$. Concerning left extensions, we have to follow suffix links directed to u and to its descendants. Actually, not all these suffix links are interesting but only those whose source node is not a child of a node from which a suffix link directed to u, or to a descendant closest to u, comes. Denoted by $sln[u]$ the set of these nodes, $m' = \alpha m$ has an occurrence-path ending at string-depth $(|m| + 1)$ along the edge ending at any node $v \in sln[u]$, only if the label of v starts with α, or if it starts with $\beta \neq \alpha$ and $d < e$.

Theorem 4. *An approximate motif m is supermaximal iff, for every $\alpha \in \Sigma$, $m' = \alpha m$ (resp. $m' = \alpha m$) has occurrence-paths ending at nodes, or inside edges with destination nodes, ${u_1}', \ldots, {u_h}'$ such that $\sum_{i=1}^{h} Lv[{u_i}'] < q$.*

4 Inferring Maximal Motifs with Suffix Tree

The first (exact) algorithm working on suffix trees was introduced for the extraction of motifs with mismatches in [9]. Motifs are considered in lexicographical order starting from the empty word, and they are extended on the right as long as the quorum is satisfied until either a valid motif of maximal length is found (if the required length is reached), or the quorum is no longer satisfied. At each moment, all paths spelling approximated occurrences of the current motif are taken

into account. The number of the motif's occurrences is then computed as the sum of $Lv[v]$ for all nodes or destination nodes v of edges at which such occurrence-paths end. The algorithm exploits the property that these occurrence-paths on the suffix tree are such that those of the motif $m\alpha$ (where $m \in \Sigma^*$ and $\alpha \in \Sigma$) are found just going along the occurrence-paths of m and checking whether there is a character α following or a new mismatch can be introduced. Assuming that the required length of the motif is ℓ, and that at most e mismatches are allowed, the algorithm has worst case time complexity in $O(t_\ell \nu(e, \ell))$, where t_ℓ is the number of tree nodes at depth ℓ, and $\nu(e, \ell)$ is the number of words of length ℓ that differ in at most e letters from a word m of length ℓ. Finally, the space complexity is $O(t_\ell)$. The algorithm above was extended in [6] to the case of *structured motifs*, that is motifs composed of two or more parts lying at a certain given distance. The resulting tool, named SMILE, was applied to promoter signals detection in [12]. Moreover, using a data structure which is an enriched version of the suffix tree, basically the same framework has been used in [1]. Finally, the tool presented in [8], which resulted [2] to have very good performances, uses an algorithm that is basically an heuristic version of [9]. The definitions of maximality introduced in this paper and the characterizations on the suffix tree can be used by all the algorithms and tools mentioned above to output directly only (super)maximal motifs, with the consequent obvious improvement in readability and significance. Moreover, our results also apply to the versions of the problem that require motifs *common* to a set of input strings (rather than repeated within the same unique input string).

We now provide a brief description of the operations needed to extend existing motif discovery algorithms in order to extract only (super)maximal motifs and we show that the additional complexity cost due to motif (super)maximality check is negligible. Let us first consider maximal exact motif extraction. In [3], Gusfield presents a linear time algorithm to find left diverse nodes of a suffix tree T. A bottom-up traversal of T is performed, and, for each node v, the algorithm stores either that v is left diverse, or the common left character of every leaf in the subtree rooted at v. Hence, assuming that we have a suffix tree whose nodes are annotated with this information, the additional cost to select only maximal motifs among all exact motifs is constant: for every motif found, it is enough to verify whether it labels a left diverse node of T. If we search for maximal approximate motifs m, the existing extraction algorithms can be simply extended in order to test out the conditions of Theorems 2 and 3. For every occurrence-path of m, if it ends at a node v or else if it ends inside an edge with destination node v and the character at string-depth $(|m| + 1)$ along that edge is different from the one of already identified occurrence-paths of m that end inside edges, then m is right maximal. Moreover, if v is a left diverse node, or else if the left character with which v is annotated is different from the one of already identified occurrence-paths of m, then m is left maximal. In both cases, the condition to check consists of two character comparisons, and then the additional cost to extract only maximal approximate motifs is constant.

Consider now the extraction of supermaximal exact motifs. All nodes v of T can be annotated with right and left q-limited property by a simple bottom-up traversal of T. Therefore, like for maximal motifs, the additional cost to extract only supermaximal exact motifs is constant. The extension of existing motif discovery algorithms in order to extract only supermaximal approximate motifs is a little more complex. It requires that for every node v of T, in addition to right and left q-limited information, we also have the set $sln[v]$ of nodes from which suffix links directed to v or to its descendants originate. This set can be implemented by an array of $|\Sigma|$ positions. At position i of $sln[v]$ there is the node from which a suffix link to v (or to one of its descendant) originates, and whose label starts with the ith character in Σ. Assuming to have at start, for every node v of T, the array $sln[v]$ storing only nodes from which suffix links directed to v come (if any), that can be built within the linear time complexity of suffix tree construction, the complete set $sln[v]$ for all nodes of T can be found with a bottom-up traversal of T with an additional cost of $|\Sigma|^2$ per each node. The existing extraction algorithms can be extended to output only supermaximal approximate motifs m in the following manner. For every occurrence-path of m identified on the suffix tree T, if it ends at a node that is not right or left q-limited or inside an edge with a destination node of this type, then m is not supermaximal. Moreover, m is not supermaximal if the number of its occurrences which are at Hamming distance strictly less than e from m exceeds the quorum q, because in this case all right and left extensions of m preserve such occurrences and so they are motifs as well. These checks introduce a constant additional cost to the extraction algorithms. After finding all the occurrence-paths of m, if none of the described cases is verified, then, if m is a motif, we must also count, for every character α in Σ, how many occurrences of m at Hamming distance equal to e from m are preserved by right and left extensions of m with α. This counting can be made examining occurrence-paths of m labeled by such occurrences as showed in Section 3. If there exists a right or left extension of degree 1 of m which occurs more than q times, then m is not supermaximal. Therefore the operations needed to count the occurrences preserved by every right and left extension of m with a character α have a cost proportional to the number of occurrence-paths of m whose label is at Hamming distance e from m. This is due to the property of the suffix tree such that the edges from a node (at most $|\Sigma|$) are lexicographically sorted and to the implementation of set $sln[v]$ as an array of $|\Sigma|$ positions: the cost for each path is constant because, in the worst case, it simply consists of a comparison between two characters. If ℓ is the motif length, these occurrence-paths are at most $p = \binom{\ell}{e} \cdot (|\Sigma|-1)^e$ and hence the additional cost to verify if a motif is supermaximal is proportional to $O(p|\Sigma|)$. Moreover, notice that if a motif is supermaximal, then the extraction algorithm can avoid to furtherly extend it, because none of its right extensions can be a motif. Thus, the overhead introduced by the supermaximality check is balanced by a reduction of the number of intermediate length motifs that have to be extended during the extraction process. We expect that the approach we suggested could sensibly

reduce the number of output motifs without changing their significance and with a constant or negligible extra time complexity. However, we are aware that in a worst case scenario the improvement could be none: one can always design a string in which all motifs are maximal. Nevertheless, biological sequences contain many more repetitions than randomly generated sequences which, on their turn, averagely would be far from containing only maximal motifs.

5 Conclusions

In order to remove the redundancy in the output of existing algorithms for finding motifs, we defined notions of (super)maximality for exact and approximate motifs. For all of them we gave a characterization on the suffix tree data structure. This allowed us to show how to adapt a whole class of algorithms based on suffix tree for which available tools exist, to infer (super)maximal motifs only. We proved that the additional computational cost due to the on the fly check of (super)maximality requirements is negligible. Therefore, our results suggest a way to improve motifs extraction tools providing outputs which are more readable and usable by biologists.

References

1. Carvalho, A.M., Freitas, A.T., Oliveira, A.L., Sagot, M.-F.: An efficient algorithm for the identification of structured motifs in dna promoter sequences. IEEE/ACM Trans. Comput. Biology Bioinform. 3(2), 126–140 (2006)
2. Tompa, M., et al.: Assessing computational tools for the discovery of transcription factor binding sites. Nature Biotechnology 23(1), 137–144 (2005)
3. Gusfield, D.: Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, New York (1997)
4. Kolpakov, R.M., Kucherov, G.: Finding approximate repetitions under hamming distance. In: ESA, pp. 170–181 (2001)
5. Kurtz, S., Ohlebusch, E., Schleiermacher, C., Stoye, J., Giegerich, R.: Computation and visualization of degenerate repeats in complete genomes. In: ISMB, pp. 228–238 (2000)
6. Marsan, L., Sagot, M.-F.: Algorithms for extracting structured motifs using a suffix tree with application to promoter and regulatory consensus identification. J. Comp. Bio. 7, 345–360 (2001)
7. McCreight, E.: A space-economical suffix tree construction algorithm. J. ACM 23(2), 262–272 (1976)
8. Pavesi, G., Mereghetti, P., Mauri, G., Pesole, G.: Weeder web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. Nucleic Acids Research 32, 199–203 (2004)
9. Sagot, M.-F.: Spelling approximate repeated or common motifs using a suffix tree. In: LATIN, pp. 374–390 (1998)
10. Ukkonen, E.: On-line construction of suffix trees. Algorithmica 14(3), 249–260 (1995)
11. Ukkonen, E.: Structural analysis of gapped motifs of a string. In: MFCS, pp. 681–690 (2007)
12. Vanet, A., Marsan, L., Labigne, A., Sagot, M.-F.: Inferring regulatory elements from a whole genome. an application to the analysis of the genome of helicobacter pylori sigma 80 family of promoter signals. J. of Mol. Biol. 297, 335–353 (2000)