Entropic Profiles, Maximal Motifs and the Discovery of Significant Repetitions in Genomic Sequences

Laxmi Parida[1], Cinzia Pizzi[2], and Simona E. Rombo[3]

[1] IBM T. J. Watson Research Center
[2] Department of Information Engineering, University of Padova
[3] Department of Mathematics and Computer Science, University of Palermo

Abstract. The degree of predictability of a sequence can be measured by its entropy and it is closely related to its repetitiveness and compressibility. Entropic profiles are useful tools to study the under- and over-representation of subsequences, providing also information about the scale of each conserved DNA region. On the other hand, compact classes of repetitive motifs, such as maximal motifs, have been proved to be useful for the identification of significant repetitions and for the compression of biological sequences. In this paper we show that there is a relationship between entropic profiles and maximal motifs, and in particular we prove that the former are a subset of the latter. As a further contribution we propose a novel linear time linear space algorithm to compute the function Entropic Profile introduced by Vinga and Almeida in [18], and we present some preliminary results on real data, showing the speed up of our approach with respect to other existing techniques.

1 Introduction

Sequence data is growing in volume with the availability of more and more precise, as well as accessible, assaying technologies. Patterns in biological sequences is central to making sense of this exploding data space, and its study continues to be a problem of vital interest. Natural notions of maximality and irredundancy have been introduced and studied in literature in order to limit the number of output patterns without losing information [3, 4, 6, 10, 11, 14–17]. Such notions are related to both the length and the occurrences of the patterns in the input sequence. Maximal patterns have been successfully applied to the identification of biologically significant repetitions, and compressibility of biological sequences, to list a few areas of use.

Different flavors of patterns, based either on combinatorics or statistics, can usually be shown to be a variation on this basic concept of maximal patterns. In particular, it is well known that the degree of predictability of a sequence can be measured by its entropy and, at the same time, it is also closely related with its repetitiveness and compressibility [9]. Entropic profile was introduced [7, 8, 18] to study the under- and over-representation of segments, and also the scale of each conserved DNA region.

D. Brown and B. Morgenstern (Eds.): WABI 2014, LNBI 8701, pp. 148–160, 2014.
© Springer-Verlag Berlin Heidelberg 2014

Due to the fundamental nature of maximality, a natural question arises about a possible relationship between maximal patterns and entropic profiles. We explore this question in the paper and show that entropic profiles are indeed a subset of maximal patterns. Based on this inshight, we improve the running time of the detection of entropic profiles by proposing an efficient algorithm to extract entropic profiles in $O(n)$ time and space. The algorithm exploits well known properties of the suffix tree to group together the subwords that are needed to compute the entropy for a specific position and for the input sequence as a whole.

Finally, we present an experimental validation of the proposed algorithm performed on the whole genome of *Haemophilus influenzae*, showing that our approach outperforms the other existing techniques in terms of time performance.

The manuscript is organized as follows. In the next section we recall some basic notions about DNA sequence entropic profiles and maximal motifs, while in the next section we show the relationship occurring between them. Section 4 presents our linear time linear space algorithms, and some preliminary experimental comparisons are discussed in Section 5. The paper ends with a summary of results and some further considerations.

2 Background

Let $x = x_1 \dots x_n$ be a string defined over an alphabet Σ. We denote by $x_i \dots x_j$ the subword of x starting at position i and ending at position $j > i$, and by $c([i, j])$ the number of occurrences of $x_i \dots x_j$ in x.

2.1 Maximal Motifs

Among all the candidate over-represented subwords of an input string, those presenting special properties of maximal saturation have been proved to be a special compact class of motifs with high informative content, and they have been shown to be computable in linear time [12]. We next recall some basic definitions.

Definition 1. *(Left-maximal motif) The subword $x' = x_i \dots x_j$ of x is a* left-maximal motif *if it does not extist any other subword $x'' = x_{i-h} \dots x_j$ $(0 < h \leq i)$ such that $c([i, j]) = c([i - h, j])$.*

Definition 2. *(Right-maximal motif) The subword $x' = x_i \dots x_j$ of x is a* right-maximal motif *if there exist no subword $x''' = x_i \dots x_{j+k}$ $(0 < k < n - j)$ such that $c([i, j]) = c([i, j + k])$.*

Definition 3. *(Maximal motif) The subword x' is a* maximal subword *if it is both left- and right- maximal.*

Maximal motifs are those subwords of the input string which cannot be *extended* at the left or at the right without loosing at least one of their occurrences.

2.2 Entropic Profiles

Entropic profiles may be estimated according to different entropy formulations. The definitions on entropic profiles recalled here are taken from the seminal papers [7, 8, 18], where the Rényi entropy of probability density estimation and the Parzen's window method applied to Chaos Game Representation/Universal Sequence Maps are exploited.

Let L be the chosen length resolution and ϕ be a smoothing parameter.

Definition 4. *(Main EP function) The main EP function is given by:*

$$\hat{f}_{L,\phi}(x_i) = \frac{1 + \frac{1}{n}\sum_{k=1}^{L} 4^k \phi^k \cdot c([i-k+1, i])}{\sum_{k=0}^{L} \phi^k}$$

Definition 5. *(Normalized EP) Let $m_{L,\phi}$ be the mean and $S_{L,\phi}$ be the standard deviation using all positions $i = 1 \ldots n$. The normalized EP is:*

$$EP_{L,\phi}(x_i) = \frac{\hat{f}_{L,\phi}(x_i) - m_{L,\phi}}{S_{L,\phi}}$$

where:

$$m_{L,\phi} = \frac{1}{n}\sum_{i=1}^{n} \hat{f}_{L,\phi}(x_i) \text{ and } S_{L,\phi} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} (\hat{f}_{L,\phi}(x_i) - m_{L,\phi})^2}$$

The main entropy function $\hat{f}$ is shown to be computable in linear time in [5]. In that work, however, the normalized entropy as defined in the original papers is not considered. A different normalization is defined instead:

$$FastEP_{L,\phi} = \frac{f_{L,\phi}(i)}{\max_{0 \le j < n[f_{L,\phi}(j)]}}$$

3 Entropic Profiles vs Maximal Motifs

We now discuss the relationship between entropic profiles and maximal motifs. The following theorem holds.

Theorem 1. *The entropic profiles scoring maximum values of the main EP function $\hat{f}$ are left-maximal motifs of the input string.*

Proof. Let i be a generic position of the input string and L' be the length of the subword x' starting at $i - L' + 1$ and ending at i, such that x' scores the maximum value of entropy at the position i. Then the following inequalities hold,

with respect to the two subwords x'' and x''' of length $L'+1$ and $L'-1$, ending at i and starting at $i-L'$ and at $i-L'+2$, respectively:

$$\begin{cases} \frac{n+\sum_{k=1}^{L'} 4^k \phi^k c([i-k+1,i])}{\sum_{k=0}^{L'} \phi^k} \geq \frac{n+\sum_{k=1}^{L'+1} 4^k \phi^k c([i-k+1,i])}{\sum_{k=0}^{L'+1} \phi^k} \\ \frac{n+\sum_{k=1}^{L'} 4^k \phi^k c([i-k+1,i])}{\sum_{k=0}^{L'} \phi^k} \geq \frac{n+\sum_{k=1}^{L'-1} 4^k \phi^k c([i-k+1,i])}{\sum_{k=0}^{L'-1} \phi^k} \end{cases}$$

As shown in the Appendix, the two inequalities above can be rewritten as:

$$\begin{cases} \frac{n+\sum_{k=1}^{L'} 4^k \phi^k c([i-k+1,i])}{\sum_{k=0}^{L'} \phi^k} \geq \frac{4^{L'+1}\phi^{L'+1} c([i-L',i])}{\phi^{L'+1}} \\ \frac{n+\sum_{k=1}^{L'-1} 4^k \phi^k c([i-k+1,i])}{\sum_{k=0}^{L'-1} \phi^k} \leq \frac{4^{L'}\phi^{L'} c([i-L'+1,i])}{\phi^{L'}} \end{cases}$$

leading to the following relation between the number of occurrences of x' and x'':

$$c([i-L'+1]) \geq 4\ c([i-L',i]).$$

Let us now suppose that x' is not left-maximal. From Definition 3, it follows that all the occurrences of x' should be covered from another subword x'' extending x' at the left of at least one character. This would mean that $c([i-L'+1])$ should be equal to $c([i-L',i])$, that is, a contraddiction. □

Note that not necessarily a left-maximal motif corresponds to a peak of entropy, as shown by the following example.

Example 1. Let $\phi = 10$ and consider the following input string:

0	1	2	3	4	5	6	7	8	9	10	11
T	C	A	A	C	G	G	C	G	G	C	T

We wonder if the maximal motif $CGGC$, ending at positions 7 and 10, corresponds to a peak of entropy at one of those positions. We have that $\hat{f}_{3,10}(7) = 9.85$, $\hat{f}_{4,10}(7) = 39.38$ and $\hat{f}_{4,10}(7) = 76.8$, therefore $CGGC$ has not a peak of $\hat{f}$ at that position. The same values occur for the position $i = 10$.

4 Methods

In this section the available algorithms to compute entropic profiles are discussed, and faster algorithms for entropic profiles computation and normalization are presented. We recall that we want to analyze an input string x of length n by means of entropic profiles of resolution L and for a fixed ϕ.

4.1 Existing Algorithms

There are two algorithms available in literature that compute entropic profiles.

The algorithm described in [8] is a faster version of the original algorithm proposed by Fernandes et al. [7]. It relies on a truncated suffix trie data structure, which is quadratic both in time and space occupation, enhanced with a list of side links that connect all the nodes at the same depth in the tree. This is needed to speed up the normalization because, in the formulas used to compute mean and standard deviation [7], the counting of subwords of the same length is a routine operation. With this approach the maximum value of L had to be set to 15.

The other method, presented in [5], uses a suffix tree on the reverse string to obtain linear time and space computation of the absolute values of entropy for some paramethers L and ϕ. These values are then normalized with respect to the maximum value of entropy among all the substrings of length L. To obtain the maximum value $\max_L$, in correspondence of a given L, all values $\max_l$, where $1 \leq l < L$, are needed. The algorithm has a worst case complexity $O(n^2)$, but being guided by a branch-and-cut technique in practice substantial savings are possible.

A key property of both suffix tries and suffix tree [12] is that, once the data structure is built on a text string x, the occurrences of a pattern $y = y_1 \dots y_m$ in x can be found by following the path labelled with $y_1 \dots y_m$ from the root of the tree. If such a path exists, the occurrences are given by the indexes of the leaves of the subtree rooted at the node in which the path ends. Moreover, being the suffix tree a compact version of a suffix trie, we have for it the further property that all the strings corresponding to paths that end in the "middle" of an arc between two nodes share the same set of occurrences. Figure 1 shows an example of trie and suffix tree.

4.2 Preprocessing

For the computation of the values needed to obtain both the absolute and the normalized values of entropy, we perform the same preprocessing procedure described in [5]. We recall here the main steps as we will need the annotated suffix tree for the subsequent description of the speed up to compute the mean and the standard deviation.

Consider the suffix tree T built on the reverse of the input string x. In such a tree, strings that are described from paths ending at the same locus share the same set of ending positions in the original input string. Hence, they are exactly the strings we need to consider when computing the values of entropy. Some care needs to be taken to map the actual positions during the computation, but this will not affect the time complexity. Therefore, in the following discussion we will just refer to the standard association between strings and positions in a suffix tree, keeping in mind they are actually reversed.

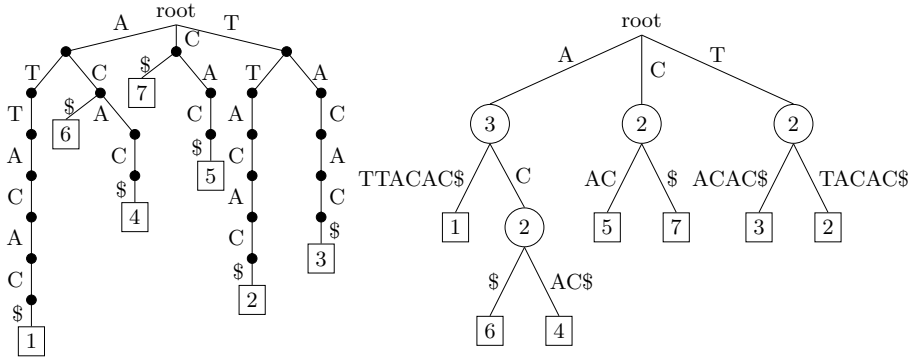

Fig. 1. A suffix trie (left) and a suffix tree (right) built on the same string $x = ATTACAC\$$. The leaves correspond to positions in x. The internal nodes in the suffix tree hold the number of occurrences of the strings that have the node as a locus.

The main observation in [5] is that in the reverse tree the absolute value of the EP function for $n - i$ is equal to:

$$f_{L,\phi}(x_i) = \frac{1 + \frac{1}{n}\sum_{k=1}^{L} 4^k \phi^k \cdot c([i, i+k-1])}{\sum_{k=0}^{L} \phi^k}$$

In the suffix tree T each node v is annotated with a variable *count(v)* which stores the number of occurrences of the subword $w(v)$, given by the concatenation of labels from the root to the node v. This can be done in linear time with a bottom-up traversal by setting up the value of the leaves to 1, and the value of the internal nodes to the sum of the values of their children.

Each node v is also annotated with the value of the main summation in the entropy formula. Let i be the position at which occurs the string $w(v)$:

$$main(v) = \sum_{k=1}^{L} 4^k \phi^k \cdot c([i, i+k-1])$$

Note that once this value is available the absolute value of entropy for $w(v)$ can be computed in constant time:

$$\frac{1 + \frac{1}{n} main(v)(1-\phi)}{1 - \phi^{L+1}}$$

Now let $h(v)$ be the length of $w(v)$ and *parent(v)* be the parent node of v. The annotation takes linear time with a pre-order traversal of the tree that passes the contribution of shorter prefixes to the following nodes in the path:

$$main(v) = main(parent(v)) + \sum_{k=h(parent(v))+1}^{h(v)} (4\phi)^k count(v)$$

When *main(parent(v))* is known, the value of *main(v)* can be computed in constant time, since $count(v)$ does not depend on k:

$$main(v) = main(parent(v)) + count(v)\frac{(4\phi)^{h(parent(v))+1} - (4\phi)^{h(v)+1}}{1 - 4\phi}$$

4.3 Efficient Computation of Entropy and Normalizing Factors

Once the annotation is complete, one can retrieve the entropy for a substring $x[i, i + L - 1]$ by following the path of length L from the root. If it ends at a node v the value of $main(v)$ is retrieved, and the absolute value of entropy is computed, otherwise the additional factor:

$$\sum_{k=h(parent(v))+1}^{L} (4\phi)^k count(v) = count(v)\frac{(4\phi)^{h(parent(v))+1} - (4\phi)^{L+1}}{1 - 4\phi}$$

needs to be added to $main(parent(v))$.

In [5] each string of length L starting at each position one wants to analyze is searched for in the suffix tree, and the value of entropy is computed as described above (and normalized according to the maximum value of entropy for length L). In discovery frameworks, where no information about the motif position is known in advance, and there are potentially as many positions to analyze as the length of the input string, this might not be the fastest solution.

On the other hand, by exploting well known properties of the suffix tree [12] it is possible to propose a different approach that is as simple as powerful, and allowed us to obtain linear time and space algorithms not only for the computation of the absolute value of entropy, but also for its normalization through mean and standard deviation.

Absolute Value of Entropy. We can collect the absolute value of entropy for all positions in the input string with a simple traversal of the tree at depth L (in terms of length of strings that labels the paths). The steps to follow when computing the entropy once we reach the last node of the path are the same we already described for computing the entropy of a given substring. Differently from before, when we reach the last node of a path we also store the value of entropy in an array of size n (or any other suitable data structure) at the positions corresponding to the leaves of the subtree rooted at the node, which are the occurrences of the string that labels the path.

Moreover, as a byside product of this traversal, we can also collect information to compute the mean and the standard deviation in linear time.

The Mean. Consider the mean first. We need to sum up the values of entropy over all possible substrings of length L in the input string. Indeed we can re-write

the formula considering the contribution of all different subwords of length L. Let w be one of such subwords, $f_{L,\phi}(w)$ be the corresponding entropy, v_w be its locus and D_L be the set containg all the different subwords of length L in x. The mean can be rewritten as:

$$m_{L,\phi} = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_i) = \frac{1}{n}\sum_{w \in D_L} count(v_w) \times \hat{f}_{L,\phi}(w)$$

Therefore, when traversing the tree, we also keep a variable in which we add the value of entropies found at length L, multiplied by the value of $count(\cdot)$ stored at their locus.

The Standard Deviation. The standard deviation can be rewritten as:

$$S_{L,\phi} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} (\hat{f}_{L,\phi}(x_i) - m_{L,\phi})^2} = \sqrt{\frac{1}{n-1}\left(\sum_{i=1}^{n} (\hat{f}^2_{L,\phi}(x_i)) - nm^2_{L,\phi}\right)}$$

Again we aggregate the contribution coming from the same subwords, so that $\sum_{i=1}^{n} \hat{f}^2_{L,\phi}(x_i)$ becomes:

$$\sum_{w \in D_L} (count(v_w) \times \hat{f}^2_{L,\phi}(w))$$

To compute this sum, when traversing the tree we keep a variable in which we add the square of the entropies we compute at length L, multiplied by the value of $count(\cdot)$ stored at their locus.

Once the above summation and the mean have been computed with a single traversal at depth L of our tree, we have all the elements needed to compute the standard deviation in constant time.

The Maximum. As a side observation, one can also note as, in terms of asyntotic complexity, the maximum value of entropy can also be retrieved in linear time with a tree traversal without need to compute the value of $\max_l, 1 \leq l < L$.

4.4 Practical Considerations

We described our algorithms in terms of suffix tree, but we do not really need the entire tree. A truncated suffix tree [2, 13], with a truncation factor equal to the maximum L one is willing to investigate, would be sufficient. Alternatively, an enhanced suffix array [1] allowing a traversal of the virtual LCP tree could also be used.

Note also that if we keep track of the frontier at depth L, i.e., the last nodes of the paths we visit when traversing the tree to compute $f_{L,\phi}$, we can compute the entropic profiles for longer L without starting from the root. Indeed, even in the case we have to start from the root, the preprocessing step does not need to be repeated.

5 Experimental Analysis

In this section we present the results of the experimental analysis we performed on the whole genome of *Haemophilus influenzae*, which is one of the most extensively analyzed in this context. For all the considered methods, the time performance evaluations we show do not include the preprocessing step, i.e., the construction of the exploited data structures (suffix trees or suffix tries), whereas the time needed to annotate the tree is always included. All the tests were run on a laptop with a 3.06GHz Core 2 Duo and 8Gb of Ram.

Figure 2 shows a comparison among the original EP function computation by Vinga et al [18] (denoted by EP in the following), FastEP by Comin and Antonello [5] and our approach, that is, LinearEP. In particular, the running times in milliseconds are shown for $\phi = 10$, $L = 10$ and increasing values of n. As it is clear from the figure, LinearEP outperfomes the other two methods, thus confirming the theoretic results.

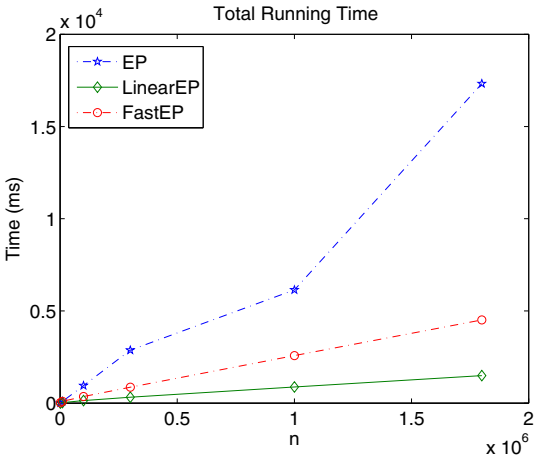

Fig. 2. Comparison among EP, FastEP and LinearEP ($\phi = 10$, $L = 10$) for increasing values of n. Total running time includes the computation of the normalizing factors, and the normalized EP values for the whole sequence.

We recall that both EP and LinearEP compute the normalized EP function according to the same formulation, that is, with respect to the mean and standard deviation (for a fixed lenght L), whereas FastEP computes a different normalization with respect to the maximum value of the main EP function. We then performed also a direct comparison between EP and LinearEP for the computation of mean and standard deviation. Figure 3 shows that, except for $n = 1{,}000$, LinearEP is faster than EP in such a computation (however, the total time is lower for LinearEP also for $n = 1{,}000$, as shown in Figure 2).

We finally compared EP and LinearEP on a window of lenght 1,000, for L varying between 3 and 15. Note that 15 is a technical limit imposed by the

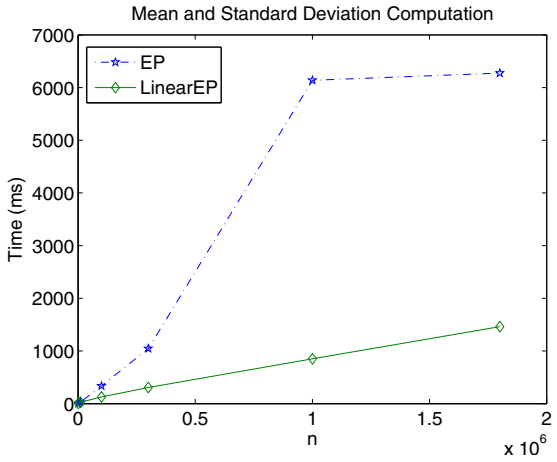

Fig. 3. Comparison between EP and LinearEP for the computation of mean and standard deviation

software of EP. We do not have such limitation. Indeed we tested our algorithm till $L = 100$ obtaining results close to those for $L = 15$. Figure 4 shows the results for the time needed to compute the mean and the standard deviation. The first observation is that, in both cases, the performances of EP do not change significantly for increasing values of L, whereas the running times of LinearEP sensibly increase for increasing values of L. This is due to the fact that n is fixed and L varies. To compute mean and standard deviation LinearEP traverses a

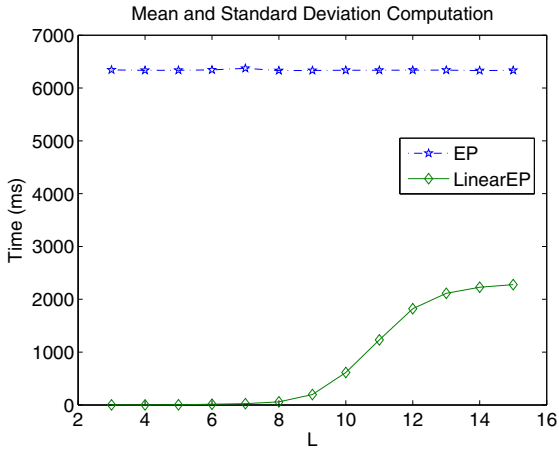

Fig. 4. Computation of mean and standard deviation for EP and LinearEP ($L \in [3, 15]$)

portion of the tree that is dependent on L, while EP computation of mean and standard deviation is mainly dependent on the sequence length, that is fixed. The running times of LinearEP are from three magnitude order to three times faster than those of EP.

For sake of completeness of the presented results, we add some details about the preprocessing steps performed by each of the considered software tools. In particular, both our prototype and the software of [5] build a full suffix tree although they do not need it in principle. Moreover, the algorithm of [18] is implemented in C, while the others are implemented in Java. The efficiency of our linear algorithm for the extraction of entropic profiles overcomes the known gap between these two languages, but this does not hold for the suffix tree construction. Indeed, building the suffix trie needed to run EP took around 2.7 seconds, while building the full suffix tree, for both other software tools, took around 12 seconds.

6 Concluding Remarks

The research proposed here includes two main contributions. The first contribution is the study of possible relationships between two classes of motifs analyzed in the literature and both effective in singling out significant biological repetitions, that are, entropic profiles and maximal motifs. We proved that entropic profiles are a subset of maximal motifs, and, in particular, that they are left-maximal motifs of the input string. The second contribution of the present manuscript is the proposal of a novel linear time linear space algorithm for the extraction of entropic profiles, according to the original normalization reported in [7]. Experimental validations confirmed that the algorithm proposed here is faster than the others in the literature, including a recent approach where a different normalization was introduced [5].

From these contributions interesting considerations emerge. First of all, we observe that entropic profiles are related to a specific length which one can only guess when doing de novo discovery. So one could think of extracting maximal motifs first, and then investigate entropic profiles in the regions of the maximal motifs and for values of L around the maximal reported length. The process of discovery of the entropic profiles would be further improved then. Other improvements in the entropic profiles extraction could come from the exploitation of more efficient data structures such as enhanced suffix arrays [1]. In this regard, we note that the preprocessing step can also be speeded since, as already pointed out in the previous section, a full suffix tree is not necessary for the computation. Finally, open challenges still remain open about further issues concerning maximal motifs and entropic profiles. Notably among them, one may wonder if entropic profiles do not recover the complete information, so maximal motifs are more reliable when it comes to discovery problems or if, on the contrary, entropic profiles cover the complete information, i.e., they are a refinement of maximal motifs and should be preferred.

Acknowledgments. The research by C. Pizzi and S. E. Rombo was partially supported by the Project "Approcci composizionali per la caratterizzazione e il mining di dati omici" (PRIN 20122F87B2). C. Pizzi was also partially supported by the "Progetto di Ateneo" Univ. of Padova CPDA11023.

References

1. Abouelhoda, M.I., Kurtz, S., Ohlebusch, E.: Replacing suffix trees with enhanced suffix arrays. Journal of Discrete Algorithms 2, 53–86 (2004)
2. Allali, A., Sagot, M.-F.: The at most k deep factor tree. Technical Report (2004)
3. Apostolico, A., Parida, L.: Incremental paradigms of motif discovery. Journal of Computational Biology 11, 1 (2004)
4. Apostolico, A., Pizzi, C., Ukkonen, E.: Efficient algorithms for the discovery of gapped factors. Algorithms for Molecular Biology 6, 5 (2011)
5. Comin, M., Antonello, M.: Fast Computation of Entropic Profiles for the Detection of Conservation in Genomes. In: Ngom, A., Formenti, E., Hao, J.-K., Zhao, X.-M., van Laarhoven, T. (eds.) PRIB 2013. LNCS, vol. 7986, pp. 277–288. Springer, Heidelberg (2013)
6. Federico, M., Pisanti, N.: Suffix tree characterization of maximal motifs in biological sequences. Theoretical Computer Science 410(43), 4391–4401 (2009)
7. Fernandes, F., Freitas, A.T., Vinga, S.: Detection of conserved regions in genomes using entropic profiles. INESC-ID Tec. Rep. 33/2007
8. Fernandes, F., Freitas, A.T., Almeida, J.S., Vinga, S.: Entropic Profiler – detection of conservation in genomes using information theory. BMC Research Notes 2, 72 (2009)
9. Herzel, H., Ebeling, W., Schmitt, A.O.: Entropies of biosequences: The role of repeats. Physical Review E 50, 5061–5071 (1994)
10. Grossi, R., Pietracaprina, A., Pisanti, N., Pucci, G., Upfal, E., Vandin, F.: MADMX: a strategy for maximal dense motif extraction. Journal of Computational Biology 18(4), 535–545 (2011)
11. Grossi, R., Pisanti, N., Crochemore, M., Sagot, M.-F.: Bases of motifs for generating repeated patterns with wild cards. IEEE/ACM Transactions on Computational Biology and Bioinformatics 2(1), 40–50 (2005)
12. Gusfield, D.: Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge University Press (1997)
13. Na, J.C., Apostolico, A., Iliopulos, C.S., Park, K.: Truncated suffix trees and their application to data compression. Theoretical Computer Science 304(1-3), 87–101 (2003)
14. Parida, L.: Pattern Discovery in Bioinformatics: Theory & Algorithms. Chapman & Hall/CRC (2007)
15. Parida, L., Pizzi, C., Rombo, S.E.: Characterization and Extraction of Irredundant Tandem Motifs. In: Calderón-Benavides, L., González-Caro, C., Chávez, E., Ziviani, N. (eds.) SPIRE 2012. LNCS, vol. 7608, pp. 385–397. Springer, Heidelberg (2012)
16. Parida, L., Pizzi, C., Rombo, S.E.: Irredundant tandem motifs. Theoretical Computer Science 525, 89–102 (2014)
17. Rombo, S.E.: Extracting string motif bases for quorum higher than two. Theoretical Computer Science 460, 94–103 (2012)
18. Vinga, S., Almeida, J.S.: Local Renyi entropic profiles of DNA sequences. BMC Bioinformatics 8, 393 (2007)

Appendix

Let us start from the following two inequalities:

$$\begin{cases} \frac{N+\sum_{k=1}^{L'} 4^k\phi^k c([i-k+1,i])}{\sum_{k=0}^{L'}\phi^k} \geq \frac{N+\sum_{k=1}^{L'+1} 4^k\phi^k c([i-k+1,i])}{\sum_{k=0}^{L'+1}\phi^k} \\ \frac{N+\sum_{k=1}^{L'} 4^k\phi^k c([i-k+1,i])}{\sum_{k=0}^{L'}\phi^k} \geq \frac{N+\sum_{k=1}^{L'-1} 4^k\phi^k c([i-k+1,i])}{\sum_{k=0}^{L'-1}\phi^k} \end{cases}$$

from which:

$$\begin{cases} \frac{N+\sum_{k=1}^{L'} 4^k\phi^k c([i-k+1,i])}{\sum_{k=0}^{L'}\phi^k} \geq \frac{N+\sum_{k=1}^{L'} 4^k\phi^k c([i-k+1,i])+4^{L'+1}\phi^{L'+1}c([i-L',i])}{\sum_{k=0}^{L'}\phi^k+\phi^{L'+1}} \\ \frac{N+\sum_{k=1}^{L'-1} 4^k\phi^k c([i-k+1,i])+4^{L'}\phi^{L'}c([i-L'+1,i])}{\sum_{k=0}^{L'-1}\phi^k+\phi^{L'}} \geq \frac{N+\sum_{k=1}^{L'-1} 4^k\phi^k c([i-k+1,i])}{\sum_{k=0}^{L'-1}\phi^k} \end{cases}$$

Let us consider $A = N + \sum_{k=1}^{L'} 4^k\phi^k c([i-k+1,i])$, $B = \sum_{k=0}^{L'}\phi^k$, $C = 4^{L'+1}\phi^{L'+1}c([i-L',i])$, $D = \phi^{L'+1}$; $A' = N + \sum_{k=1}^{L'-1} 4^k\phi^k c([i-k+1,i])$, $B' = \sum_{k=0}^{L'-1}\phi^k$, $C' = 4^{L'}\phi^{L'}c([i-L'+1,i])$ and $D' = \phi^{L'}$. Then:

$$\begin{cases} \frac{A}{B} \geq \frac{A+C}{B+D} \Longrightarrow \frac{A}{B} \geq \frac{C}{D} \\ \frac{A'+C'}{B'+D'} \geq \frac{A'}{B'} \Longrightarrow \frac{A'}{B'} \leq \frac{C'}{D'} \end{cases}$$

The two inequalities above can be then rewritten as:

$$\begin{cases} \frac{N+\sum_{k=1}^{L'} 4^k\phi^k c([i-k+1,i])}{\sum_{k=0}^{L'}\phi^k} \geq \frac{4^{L'+1}\phi^{L'+1}c([i-L',i])}{\phi^{L'+1}} \\ \frac{N+\sum_{k=1}^{L'-1} 4^k\phi^k c([i-k+1,i])}{\sum_{k=0}^{L'-1}\phi^k} \leq \frac{4^{L'}\phi^{L'}c([i-L'+1,i])}{\phi^{L'}} \end{cases}$$

$$\begin{cases} N + \sum_{k=1}^{L'-1} 4^k\phi^k c([i-k+1,i]) \geq \frac{\sum_{k=0}^{L'}\phi^k}{\phi^{L'+1}} 4^{L'+1}\phi^{L'+1}c([i-L',i]) - 4^{L'}\phi^{L'}c([i-L'+1,i]) \\ N + \sum_{k=1}^{L'-1} 4^k\phi^k c([i-k+1,i]) \leq \frac{\sum_{k=0}^{L'-1}\phi^k}{\phi^{L'}} 4^{L'}\phi^{L'}c([i-L'+1,i]) \end{cases}$$

Then:

$$\frac{\sum_{k=0}^{L'}\phi^k}{\phi^{L'+1}} 4^{L'+1}\phi^{L'+1}c([i-L',i]) - 4^{L'}\phi^{L'}c([i-L'+1,i]) \leq \frac{\sum_{k=0}^{L'-1}\phi^k}{\phi^{L'}} 4^{L'}\phi^{L'}c([i-L'+1,i])$$

$$\sum_{k=0}^{L'}\phi^k 4c([i-L',i]) - \phi^{L'}c([i-L'+1,i]) - \sum_{k=0}^{L'-1}\phi^k c([i-L'+1,i]) \leq 0$$

$$(\phi^{L'} + \sum_{k=0}^{L'-1}\phi^k)c([i-L'+1,i]) \geq 4\sum_{k=0}^{L'}\phi^k c([i-L',i])$$

$$c([i-L'+1,i]) \geq 4c([i-L',i])$$