Fast Computation of Entropic Profiles for the Detection of Conservation in Genomes

Matteo Comin and Morris Antonello

Department of Information Engineering, University of Padova,
Via Gradenigo 6/A, Padova, Italy
comin@dei.unipd.it

Abstract. The information theory has been used for quite some time in the area of computational biology. In this paper we discuss and improve the function Entropic Profile, introduced by Vinga and Almeida in [23]. The Entropic Profiler is a function of the genomic location that captures the importance of that region with respect to the whole genome. We provide a linear time linear space algorithm called Fast Entropic Profile, as opposed to the original quadratic implementation. Moreover we propose an alternative normalization that can be also efficiently implemented. We show that Fast EP is suitable for large genomes and for the discovery of motifs with unbounded length.

Keywords: pattern discovery, information theory, computational biology.

1 Introduction

The concept of information theory was originally introduced by Claude E. Shannon as a tool to systematically analyze data flow in general communication systems [20]. The theory has been extended and subsequently applied to many fields including DNA sequence analysis [24]. Methods of Information theory focusing on DNA sequence compression have found differences between coding and noncoding sequences [17] and they have been applied also for classification [3,4]. In [12] the authors applied the mutual information to discover SNPs that are significantly associated with diseases. Also compression based classification relying on mutual information can be successfully applied to phylogeny [2]. Moreover the identification of splicing mutations can benefit from the use of Information Theory[18]. In [11] sequence motifs are modeled based on the maximum entropy principle. Such models can be utilized to discriminate between signals and decoys. In [5] an entropic segmentation method is discussed to detect borders between coding and noncoding DNA. These are just a few examples of the computational biology applications inspired by information theory.

In this paper we discuss and improve the function Entropic Profile, introduced by Vinga and Almeida in [23]. The concept of Entropic Profiler was introduce to analyze DNA sequences. The Entropic Profiler is a function of the genomic location that captures the importance of that region with respect to the whole genome. This score is based on the Shannon entropies of the words distribution. This method proved useful for the identification of conserved genomic regions.

A. Ngom et al. (Eds.): PRIB 2013, LNBI 7986, pp. 277–288, 2013.

Other types of sequence profile have also been previously explored like Sequence Logos [19], that provide the information content per position. This method, however, requires the alignment of a set of sequences and thus it is not suited for a single sequence. Moreover this approach does not comply to the alignment-free paradigm like [8].

One of the most important requirements is the development of efficient methods for the analysis of whole genomes that can scale gracefully with the size of input. In this paper we study the use of Suffix Tree for the computation of the Entropic Profiler. We show that the same function can be evaluated in linear time and space as opposed to the quadratic implementation of EP [23]. This will allow the use of longer genomes and the discovery of motifs with unbounded length, removing the limitations of the current implementation. Moreover we propose an alternative normalization that can be also efficiently implemented within the Suffix Tree structure. The resulting implementation will be named Fast Entropic Profile (FastEP). We show that FastEP proved useful for the detection of conserved signals.

1.1 Entropic Profiler

Although DNA is a flexible three-dimensional molecule interacting in a dynamic environment, its digital information can be represented by a one dimensional character string of G's, A's, T's and C's. Following this standard assumption, two of its most striking features are the extent to which repeated L-tuples occur and the variety of repeated structures it contains. These topics have been discussed extensively and various mechanisms try to explain the functional and evolutionary role of repeats. The degree of predictability and randomness of a substring is described by its entropy [23]. Entropic Profiles (EP) are plots estimated by this local entropy formulation, defined for each position/symbol, from the complete sequence of DNA. The original definition is based on the distribution of words that end at a particular location i. Let s be the input genome of length $|s| = n$, we define $s[i, i+k-1]$ as the word of length k that starts at position i. Let $c[i, i+k-1]$ be the number of time the word $s[i, i+k-1]$ appears in the genome s. The function local entropy for position i is defined as:

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

where ϕ is a normalization parameter. This function can be interpreted as a linear combination of suffix counts up to a given length L, with different weights. It computes, for each location of the sequence, the information about the abundance of the corresponding L-tuple suffix inside the entire sequence. For ease of explanation we redefine the above formula to evaluate the statistic of words starting at position i, instead of ending at position i.

$$f_{L,\phi}(i) = \frac{1 + 1/n \sum_{k=1}^{L} 4^k \phi^k c[i, i+k-1]}{\sum_{k=0}^{L} \phi^k} \tag{2}$$

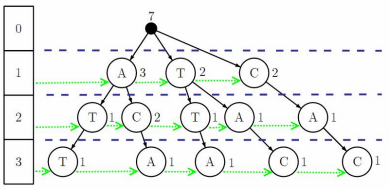

Fig. 1. Truncated suffix tree, L=3, and side links of the word ATTACAC

Note that the function $g_{L,\phi}(i)$ is equivalent to compute $f_{L,\phi}(n-i)$ for the reverse of s. This function is then normalized to allow the comparison of different parameter combinations. EP values are normalized as a z-score: $EP_{L,\phi}(i) = \frac{f_{L,\phi}(i) - m_{L,\phi}}{s_{L,\phi}}$, where the mean is $m_{L,\phi} = \frac{1}{n}\sum_{i=1}^{n} f_{L,\phi}(i)$ and the standard deviation $s_{L,\phi} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(f_{L,\phi}(i) - m_{L,\phi}\right)^2}$.

We will discuss an alternative normalization in section 3. The original implementation of the entropic profiler is based on a truncated suffix trie, see Figure 1. A standard trie, storing the collection of n suffixes of the entire DNA sequence, has the following properties:

- the number of nodes is $O(n^2)$.
- the height is equal to the length of the longest string, that is the length of the whole sequence, n.
- word matching for a pattern of length L takes $O(L)$ time.
- constructing the entire trie takes $O(n^2)$ time.

The counters at each node represent the number of occurrences of the corresponding word. This allows the main EP function to be worked out by simply word matching. All nodes at the same depth are connected by side links in order to speed up the normalization, otherwise the computation of $m_{L,\phi}$ and $s_{L,\phi}$ would involve the repeated calculation of the main EP function for all positions.

There are two problems with this implementation. The first issue is that it is space inefficient. Specifically, there may be a lot of nodes that have only one child, and the existence of such nodes is a waste. The second problem is that the Entropic Profiler can be computed only for small L. In fact in [23] the function EP can explored only for motif shorter than 15 bases, and thus the trie is truncated at depth 15. These observations have prompted the idea to consider instead of a trie its compressed version also known as Suffix Tree.

1.2 Preliminaries on Suffix Trees

The Suffix Tree is one of the most studied data structures and it is fundamental for string processing. It stores a string in such a way that enables the implementation of efficient searches. Traditionally the suffix tree has been used in very different fields, spanning from data compression [26,3] to clustering [10] and classification [9,8]. The use of suffix tree has become very popular in the field of bioinformatics allowing a number of string operations, like detection of

repeats [14], local alignment [16], the discovery of regulatory elements [6,7] and extensible patterns [1]. The optimal construction of suffix tree has already been addressed by [22,15], that provided algorithms in linear time and space. Figure 2 shows an example of suffix tree for the string $s = TCGGCGGCAAC$. We can observe that each suffix of the string s is present in the tree as a labeled path from the root to a leaf.

2 Fast Entropic Profiler

This section we describe how the entropic profiler can be efficiently computed using the suffix tree. Let assume that we have already computed the suffix tree of the input string s using the algorithm of Ukkonen [22]. We extend this structure so that every node v contains a variable $count(v)$ that stores the number of times that the word represented by v appears in s. With a simple $O(n)$ traversal of the tree we can compute the variable $count(v)$ of each internal node v, where $count(l) = 1$ if l is a leaf.

The goal is to find an efficient way to compute the main EP function 2 for every possible substring and parameter combination. If the substring taken into consideration is encoded by the suffix tree, there are two main cases: it may be spelled out by the concatenation of the edge-labels on the path from the root to a node or not. In the latter case the substring ends between two nodes.

The function $f_{L,\phi}(i)$ for each sequence belonging to the former case can be preprocessed and stored in a variable $entropy(v)$, for each node v. Now assume that the node v represents the string $s[i, i + L - 1]$ then the variable $entropy(v)$ will contain $\sum_{k=1}^{L} 4^k \phi^k c[i, i + k - 1]$, the main sum of $f_{L,\phi}(i)$. Once $entropy(v)$ is available we can calculate $f_{L,\phi}(i)$ in constant time. The following preprocessing is a preorder traversal of the tree that computes the value of $entropy(v)$ for all nodes. Let assume that $par(v)$ is the parent node of v, and that $h(v)$ is the length of the string spelled out by the concatenation of the node-labels on the path from the root to that node. In other words $h(v)$ is the length of the string represented by the node v.

Preprocess(T,v)
A suffix tree T and a node v are given.
begin [visit]
if v is the root **then**
$entropy(v) = 0$
else
$entropy(v) = entropy(par(v)) + count(v) \sum_{k=h(par(v))+1}^{h(v)} [4^k \phi^k]$
end if
for all child w of v **do**
begin [recursive traversal]
Preprocess(T,w);
end for

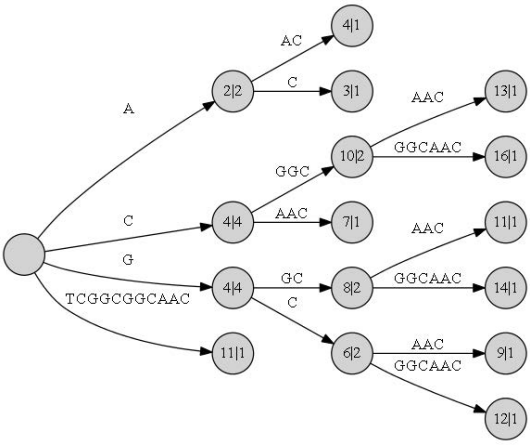

Fig. 2. Suffix tree of the string TCGGCGGCAAC. Every copy of the terminal symbol \$ is removed from the edge labels. The nodes are labeled with the corresponding values of *entropy*|*count*, where for simplicity $4\phi = 1$.

Let's consider the string $TCGGCGGCAAC$ and the suffix tree in Figure 2. The main sum for the function $f_{4,\phi}(2)$ is $\sum_{k=1}^{4} 4^k \phi^k c[2, 2+k-1]$. For ease of explanation we write $c[s[i,j]]$ instead of $c[i,j]$. This sum can be expanded in: $4\phi c[C] + (4\phi)^2 c[CG] + (4\phi)^3 c[CGG] + (4\phi)^4 c[CGGC]$. Now the information contained in the suffix tree allows us to simplify this sum. We can note that every time we see CG it is always followed by a GC, thus $c(CG) = c(CGG) = c(CGGC)$, that is also $count(v)$, where v represent the word $CGGC$. Finally if we consider that $entropy(C) = 4\phi c[C]$ that is also the node $par(v)$. Thus the previous sum can be simplified in : $4\phi c[C] + ((4\phi)^2 + (4\phi)^3 + (4\phi)^4)c[CGGC] = entropy(par(CGGC)) + count(CGGC) \sum_{2}^{4} [4^k \phi^k]$. This is equivalent to the formula used in the preprocessing, where part of the summation is simplified thanks to the suffix tree. Using the properties of the geometric series we can observe that $\sum_{k=h(par(v))+1}^{h(v)} [4^k \phi^k]$ is equivalent to $[(4\phi)^{h(par(v))+1} - (4\phi)^{h(v)+1}]/[1-4\phi]$. Thus each visit takes time $O(1)$, and the total time spent in this preprocessing is $O(n)$, linear the number of nodes.

After this preprocessing, the EP function can be retrieved efficiently for all words represented by some node in the tree T. The following algorithm computes the EP function of any word $s[i, i+L-1]$ of length L using as input the suffix tree T.

FastEP (Input: T, i, L, ϕ; Output:$f_{L,\phi}(i)$)
Search the input word $s[i, i+L-1]$ in the suffix tree T.
if it is represented by the node v **then**
the algorithm **returns** the preprocessed value of the variable $entropy(v)$ of the internal node v.
end if

if the search ends within an edge, between the two nodes u and v **then**
the algorithm **returns** the preprocesses value of $entropy(u)$
plus the correction factor $count(v) \sum_{k=h(u)+1}^{L} 4^k \phi^k$.
end if

In summary if the query word is represented in the suffix tree by a node v it is enough to return $entropy(v)$, otherwise we need to add a correction factor that is proportional to the number of times the word as a whole appears, and thus using $count(v)$. Again from the output of this procedure we can compute in constant time the Entropic Profile function (formula 2). Thus FastEP after a linear time linear space preprocessing can evaluate a certain position or equivalently a specific pattern in constant time. The original implementation requires $O(n^2)$ time and space to answer the same query.

3 Fast Entropic Profiler Normalization

The aim of this section is to provide an alternative normalization of EP such that, in order to be computed, it does not require to process all positions of s and for all L. Algebraic considerations [23] allow the mean $m_{L,\phi}$ to be rewritten as:

$$m_{L,\phi} = \frac{(\phi - 1)(m^2 + \sum_{i=1}^{L} C^2[k])}{m^2(\phi^{L+1} - 1)} \tag{3}$$

where $C^2[k]$ stands for the sum of the squared counts of all distinct words of size k in the whole sequence. Similarly, the standard deviation $s_{L,\phi}$ becomes:

$$s_{L,\phi} = \sqrt{\frac{1}{m-1}\left(\frac{S[L]}{\left(\frac{\phi^{L+1}-1}{\phi-1}\right)^2} - m_{L,\phi}^2 \cdot m\right)} \tag{4}$$

where the recursive function $S[L]$, depending on the number of distinct word of length L, is fairly intricate. Even if L-tuples are less than the length of the whole sequence n, this kind of normalization takes still $O(n^3)$ time and $O(n^2)$ space.

There are several alternatives to the above normalization. In this paper we propose to define FastEP, $FastEP_{L,\phi}(i)$ as :

$$FastEP_{L,\phi}(i) = \frac{f_{L,\phi}(i)}{\mathbf{max}_{0 \le j < n}[f_{L,\phi}(j)]} \tag{5}$$

where the function $\mathbf{max}_{0 \le j < n}[f_{L,\phi}(j)]$ returns the maximum value of $f_{L,\phi}$ over all words of size L. Similarly to the original normalization this formulation allows to compare the entropic profile scores for words of different length. In fact FastEP assumes values in the range $[0, 1]$.

3.1 Finding the Maximum Entropy $f_{L,\phi}$ for all L Using a Branch and Bound Approach

In the following we discuss a branch and bound strategy to efficiently recover the values of $\mathbf{max}_{0\leq j<n}[f_{L,\phi}(j)]$ for all L, or simply $\mathbf{max}_L$. Instead of naively comparing each word of length L, the search for the maximum FastEP can be restricted to some regions of the tree. Again for ease of explanation we will consider only the sum $\sum_{k=1}^{L} 4^k\phi^k c[i, i+k-1]$, as the main $f_{L,\phi}(j)$ can be trivially derived.

If $L > 1$, two definitions are needed to define which regions of the tree must be taken into consideration and which can be pruned:

Definition 1. *The minimum potential maximum mpm_L defines a lower bound to the maximum $f_{L,\phi}(j)$ for all L:*

$$mpm_L = \boldsymbol{max}_{L-1} + 4^L\phi^L$$

Definition 2. *The maximum potential maximum $MPM_L(v)$, where $L > 1$ and v is a node such that $h(v) < L$, is defined as:*

$$MPM_L(v) = entropy(v) + [count(v) - 1] * \sum_{k=h(v)+1}^{L} 4^k\phi^k$$

The maximum potential maximums, MPM bounds, are progressively computed and they allow to prune the search space for the maximum EP. The maximum potential maximum $MPM_L(v)$ is associated to any node v. At each step they define an upper bound to the maximum FastEP obtainable for a path starting from the root and passing trough the node v. In fact, if a $MPM_L(v)$ is less than mpm_L that region can be discarded and not considered. Otherwise if $MPM_L(v)$ is greater than mpm_L we extend this path to the child of v as long as these nodes have height not greater than L.

The following numerical example, which computes the values of $\mathbf{max}_L$ for L from 1 to 2, clarifies these concepts. Let's consider the example of Figure 2 where for simplicity we use $4\phi = 1$. For $L = 1$ it is enough to consider the most frequent character, that is G or C, that produces $\mathbf{max}_1 = entropy(C) = 4$. If L=2 it must be $\mathbf{max}_2 \geq \mathbf{max}_1 + 1 = 5$, where the second term is the minimum potential maximum $mpm_2 = 4 + 1 = 5$. Now for $L = 2$ we have that:

A: $MPM_2(A) = 2 + 1 = 3 < mpm_2 = 5 \rightarrow$ NOT acceptable path;
C: $MPM_2(C) = 4 + 3 = 7 > mpm_2 = 5 \rightarrow$ acceptable path;
G: $MPM_2(G) = 4 + 3 = 7 > mpm_2 = 5 \rightarrow$ acceptable path;
T: $MPM_2(T) = 1 + 1 = 2 < mpm_2 = 5 \rightarrow$ NOT acceptable path;

Two nodes are left out because a priori the maximum for $L = 2$ cannot be found traversing those nodes of the tree. Thus, after following every acceptable path, the value $\mathbf{max}_2$ is worked out by simply comparing:

CA: $entropy(CC) = 4 + 1 = 5$
CG: $entropy(CG) = 4 + 2 = 6$
GC,GG: $entropy(GC) = entropy(GG) = 4 + 2 = 6 \rightarrow \mathbf{max}_2 = 6$

Note that at this step no more nodes are traversed, but since $h(v) < L$ we just take the path with the maximum value of *counts*. In summary we can observe that to obtain $\mathbf{max}_L$ it requires $\mathbf{max}_{L-1}$, thus overall $\mathbf{max}_L$ can be computed in L steps. If $L = n$ in the worse case we can traverse the entire suffix tree, that is $O(n)$ nodes. Thus overall the n values of $\mathbf{max}_L$ can be computed in $O(n^2)$ time and $O(n)$ space. There are some tricks that one can use in the implementation to speedup further this process. We can note that if a node is part of an acceptable path while calculating $\mathbf{max}_L$ it will be also traversed for $\mathbf{max}_{L+1}$. Thus we don't need to traverse that part of the tree from the root, but we can just start from the latest nodes visited for $\mathbf{max}_L$. Another observation is that the value of mpm_L should be reset if the previous maximum ends in a leaf. For comparison with the original approach, based on truncated tries, the normalization process can take $O(n^3)$ time and $O(n^2)$ space, whereas our branch and bound strategy requires $O(n^2)$ times and linear space.

3.2 Expected and Real Efficiency

The expected fraction of nodes in the tree that are pruned can be computed as the following probability:

$$P(\sum_{k=1}^{L} 4^k \phi^k c[i, i+k-1] < mpm_L)$$

Given that $c[i, i+k-1]$ is a $Binomial(n, p_{w_k})$, for large values of n it can be approximated as a $Normal(np_{w_k}, np_{w_k}(1-p_{w_k}))$. Also the sum can be approximated with

$$\sum_{k=1}^{L} 4^k \phi^k c[i, i+k-1] \rightarrow \mathcal{N}(\overline{\mu}, \overline{\sigma}^2)$$

where $\overline{\mu} = \sum_{k=1}^{L} 4^k \phi^k n p_{w_k} = n \sum_{k=1}^{L} \phi^k$ and $\overline{\sigma}^2 = \sum_{k=1}^{L} 4^k \phi^k n p_{w_k}(1-p_{w_k}) = n \sum_{k=1}^{L} \phi^k (1 - 1/4^k)$.

In practice the expected efficiency depends on the distribution of words in the string s, that will determine mpm_L. For example Figure 3 reports the number of nodes visited while computing max_L for all L for the string $TCGGCGGCAAC$. Similar results are obtained also for longer random sequences (data not shown). In general small values of ϕ drastically prune the tree.

4 Results

The Fast Entropic Profiler was tested in several DNA sequences, but in this section we report the results for two genomes. Here we illustrate an example of

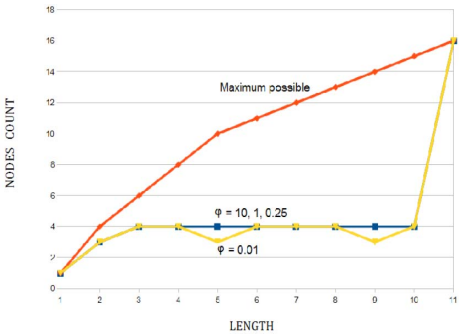

Fig. 3. Number of nodes visited for different values of ϕ while computing max_L for all possible L for the string TCGGCGGCAAC

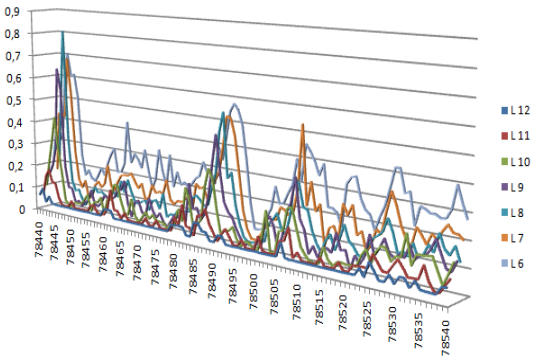

Fig. 4. Example of study of the E-Coli genome starting at position 78440 for various values of L

study around a target position. We can select a window length to study a certain range of values around the position. Also the length L can be chosen and in this case we search for pattern of length from 6 to 12. Note that after computing the values for $L = 12$ all other values for $L < 12$ can be computed in constant time. Figure 4 shows the output results for the Escherichia coli K12 genome with $\phi = 10$, starting position 78440 and window length of 100.

The figure reports the values of FastEP for all positions in the range [78440-78540]. For each position several values are reported varying the parameter L. The most important peak is at position 78445 and the value of L that maximizes this peak is $L = 8$. This highly rated motif is in fact GCTGGTGG, which corresponds to a Chi site, a region that modulates the activity of RecBCD (an enzyme involved in the chromosomal repair)[21]. It is important to notice that this pattern can be discovered just by looking at the histogram, and by analyzing the values L that maximize the score for this position, and without a previous knowledge of the length of the motif under study.

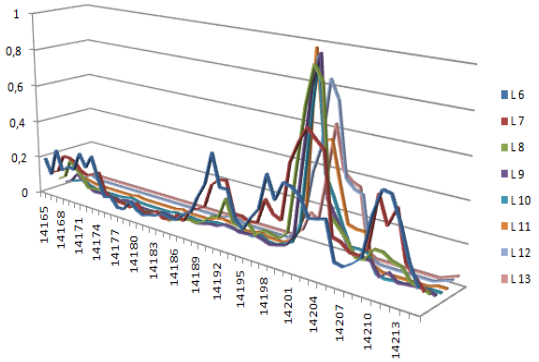

Fig. 5. Example of study of the H.Influenza genome starting at position 14165 for various values of L

Table 1. Running times in second for EP and FastEP

		FastEP		
Size	EP	Single Run	New Query	New Parameters
1 Mbases	12	4	0,09	1,5
1 Kbases	0,346	0,066	0,021	0,032

In Figure 5 a similar results is shown for the H.Influenza genome. We study the positions from 14165 to 14215 with $\phi = 10$ for various values of L. The most important peak is obtained at position 14202 for $L = 9$, that corresponds to the pattern AAGTGCGGT. This well known pattern represents an uptake signal sequence (USS+) involved in the horizontal gene transfer [13].

In a second series of experiments we test the time performance on a common laptop with a 1.5GHz Centrino and 2Gb of Ram. Table 1 reports the average times over 10 runs for two genomes of length 1kbases and 1Mbases. For all runs we use $L = 10$, $\phi = 10$ and a window of 100. In column "EP" is reported the time for the original method. For FastEP three times are illustrated. The construction and query correspond to the column "Single Run". A new query, e.g. a new starting position or a shorter L, is represented by the column "New Query". If a larger L or a new value of ϕ are required the inner structure is updated in a time reported in the last column. On a single run FastEP is always faster than the original method. If multiple queries are required the advantage becomes immediately embarrassing. The small space requirements and the improved performance will enable the study on large genomes.

Moreover in the original implementation the parameter L can not be greater than 15, whereas FastEP does not have limitation and can search for longer patterns.

5 Conclusions

To summarize we improve the original Entropic Profile with a faster and more flexible implementation that can search for longer patterns in a genome. We proposed a new normalization that can be efficiently computed within the inner structure of FastEP. We provide some examples where FastEP is used for the detection of conserved signals in a genome.

Acknowledgments. M. Comin was partially supported by the Ateneo Project CPDA110239. S. Mazzocca implemented the software FastEP.

References

1. Apostolico, A., Comin, M., Parida, L.: Varun: Discovering Extensible Motifs under Saturation Constraints. IEEE/ACM Transactions on Computational Biology and Bioinformatics 7(4), 752–762 (2010)
2. Apostolico, A., Comin, M., Parida, L.: Mining, compressing and classifying with extensible motifs. Algorithms for Molecular Biology 1, 4 (2006)
3. Apostolico, A., Comin, M., Parida, L.: Bridging Lossy and Lossless Compression by Motif Pattern Discovery. In: Ahlswede, R., Bäumer, L., Cai, N., Aydinian, H., Blinovsky, V., Deppe, C., Mashurian, H. (eds.) General Theory of Information Transfer and Combinatorics. LNCS, vol. 4123, pp. 793–813. Springer, Heidelberg (2006)
4. Apostolico, A., Comin, M., Parida, L.: Motifs in Ziv-Lempel-Welch Clef. In: Proceedings of IEEE DCC Data Compression Conference, pp. 72–81. Computer Society Press (2004)
5. Bernaola-Galván, P., Grosse, I., Carpena, P., Oliver, J., Román-Roldán, R., Stanley, H.: Finding Borders between Coding and Noncoding DNA Regions by an Entropic Segmentation Method. Physical Review Letters 85(6), 1342–1345
6. Comin, M., Parida, L.: Subtle motif discovery for the detection of DNA regulatory sites. In: Proceeding of Asia-Pacific Bioinformatics Conference, pp. 27–36 (2007)
7. Comin, M., Parida, L.: Detection of Subtle Variations as Consensus Motifs. Theoretical Computer Science 395(2-3), 158–170 (2008)
8. Comin, M., Verzotto, D.: Alignment-Free Phylogeny of Whole Genomes using Underlying Subwords. BMC Algorithms for Molecular Biology 7, 34 (2012)
9. Comin, M., Verzotto, D.: Whole-Genome Phylogeny by Virtue of Unic Subwords. In: Proceedings of 23rd International Workshop on Database and Expert Systems Applications, BIOKDD, pp. 190–194 (2012)
10. Comin, M., Verzotto, D.: The Irredundant Class Method for Remote Homology Detection of Protein Sequences. Journal of Computational Biology 18(12), 1819–1829 (2011)
11. Gene, Y., Burge, C.: Maximum Entropy Modeling of Short Sequence Motifs with Applications to RNA Splicing Signals. Journal of Computional Biology 11(2-3), 377–394 (2004)
12. Hagenauer, J., Dawy, Z., Gobel, B., Hanus, P., Mueller, J.: Genomic Analysis using Methods from Information Theory. In: Information Theory Workshop, pp. 55–59 (2004)

13. Karlin, S., Mrazek, J., Campbell, A.: Frequent oligonucleotides and peptides of the Haemophilus influenzae genome. Nucleic Acids Res. 24, 4263–4272 (1996)
14. Kurtz, S., Choudhuri, J., Ohlebusch, E., Schleiermacher, C., Stoye, J., Giegerich, R.: Reputer: The manifold applications of repeat analysis on a genome scale. Nucleic Acids Res. 29(22), 4633–4642 (2001)
15. McCreight, E.M.: A space-economical suffix tree construction algorithm. Journal of ACM 23, 262–272 (1976)
16. Meek, C., Patel, J., Kasetty, S.: Oasis: An online and accurate technique for local-alignment searches on biological sequences. In: Proceedings of 29th International Conference on Very Large Databases, pp. 910–921 (2003)
17. Menconi, G., Marangoni, R.: A compression-based approach for coding sequences identification. I. Application to prokaryotic genomes. J. Comput Biol. 13(8), 1477–1488 (2006)
18. Nalla, V., Rogan, P.: Automated Splicing Mutation Analysis by Information Theory. Human Mutaion 25, 334–342 (2005)
19. Schneider, T., Stormo, G., Gold, L., Ehrenfeucht, A.: Information content of binding sites on nucleotide sequences. Journal of Molecular Biology 188, 415–431 (1986)
20. Shannon, C.: A Mathematical Theory of Communication. Bell System Technical Journal 27(3), 379–423 (1948)
21. Sourice, S., Biaudet, V., El Karoui, M., Ehrlich, S.D., Gruss, A.: Identification of the Chi site of Haemophilus influenzae as several sequences related to the Escherichia coli Chi site. Mol. Microbiol. 27, 1021–1029 (1998)
22. Ukkonen, E.: On-line construction of suffix trees. Algorithmica 14(3), 249–260 (1995)
23. Vinga, S., Almeida, J.S.: Local Rényi entropic profiles of DNA sequences. BMC Bioinformatics 8, 393 (2007)
24. Yockey, H.: Origin of life on earth and Shannon's theory of communication. Comput. Chem. 24(1), 105–123 (2000)
25. Waterman, M.S.: An Introduction to Computational Biology: Maps, Sequences and Genomes. Chapman Hall (1995)
26. Ziv, J., Lempel, A.: A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory 23(3), 337–343 (1977)