Computing the Repetitions in a Weighted Sequence

Costas S. Iliopoulos[1], Laurent Mouchard[2], Katerina Pedikuri[3,4] and Athanasios K. Tsakalidis[3,4]

[1] Department of Computer Science, King's College London Strand, London WC2R 2LS, England
e-mail: csi@dcs.kcl.ac.uk

[2] ABISS, Atelier Biology, Informatics, Statistics and Sociolinguistics, Université de Rouen, 76821 Mont Saint Aignan Cedex, France
e-mail: Laurent.Mouchard@univ-rouen.fr

[3] Research Academic Computer Technology Institute, 61 Riga Feraiou Str., 26221 Patras, Greece
e-mail: tsak@cti.gr

[4] Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece
e-mail: perdikur@ceid.upatras.gr

Abstract. We present an $O(n \log n)$ algorithm for computing the set of repetitions in a weighted sequence with probability of appearance larger than $1/k$, where k is a given constant.

1 Introduction

The key problem today in sequencing a large string of DNA is that only a small amount of DNA can be sequenced in a single read. That is, whether the sequencing is done by a fully automated machine or by a more manual method, the longest unbroken DNA substring that can be reliably determined in a single laboratory procedure is about 300 to 1000 (approximately 500) bases long [Celera1, Celera2]. A longer string can be used in the procedure but only the initial 500 bases will be determined. Hence to sequence long strings or an entire genome, the DNA must be divided into many short strings that are individually sequenced and then used to assemble the sequence of the full string. The critical distinction between different large-scale sequencing methods is how the task of sequencing the full DNA is divided into manageable subtasks, so that the original sequence can be reassembled from sequences of length 500.

Reassembling DNA substrings introduces a degree of uncertainty for various positions in a biosequence. This notion of uncertainness was initially expressed with the use of "don't care" characters denoted as "$*$". A *don't care* symbol has the property of matching with any symbol in the given alphabet. For example the string $p = AC*C*$ matches the pattern $q = A*DCT$. In some cases scientists determine the appearance of a symbol in a position of a sequence by assigning a probability of appearance for

every symbol. In other words a *don't care* symbol is replaced by a list of probabilities of appearance for a set of characters. Such a sequence is called a weighted sequence. Other immediate applications in molecular biology include: using sequences containing degenerate bases, IUB codes [IUB], where a letter can replace several bases (for example, a B will represent a G, T or C and a H will represent A, T or C); using logo sequences [SS90] which are more or less related to consensus: either from assembly or from blocks obtained by a multiple alignment program.

In this paper we present an efficient algorithm for computing all possible repetitions of primitive words in a weighted sequence. The structure of the paper is as follows. In Section 2 we give all the basic definitions used in the rest of the paper, in Section 3 we present our algorithm while in Section 4 we give a brief time complexity analysis of the proposed method. Finally in Section 5 we conclude and discuss our research interest in open problems of the area.

2 Background

A lot of work has been done for identifying the repetitions in a word. In [Cro81], [Apo83], [Mai84] and [Sto98], authors have presented efficient methods that find occurrences of squares in a string of length n in time $O(nlogn)$ plus the time to report the detected squares. Moreover in [Kol99a] and [Kol99b] authors presented efficient algorithms to find maximal repetitions in a word. In the area of computational biology, algorithms for finding identical repetitions in biosequences are presented in [Kur99], [Tsu99] and [Mar83]. In this section we will give all the basic definitions used in the paper.

2.1 Basic Definitions

Let Σ be a finite alphabet which consists of a set of characters (or symbols). The cardinality of an alphabet denoted by $|\Sigma|$ expresses the number of distinct characters in the alphabet. A *string* or *word* is a sequence of zero or more characters drawn from an alphabet. The set of all words over the alphabet Σ is denoted by Σ^+. A word w of length n is represented by $w[1..n] = w[1]w[2]\cdots w[n]$, where $w[i] \in \Sigma$ for $1 \leq i \leq n$, and $n = |w|$ is the length of w. The empty word is the empty sequence (of zero length) and is denoted by ε; we write $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$. Moreover a word is said to be *primitive* if it cannot be written as v^e with $v \in \Sigma^+$ and $e \geq 2$.

A factor f of length p is said to occur at position i in the word w if $f = w[i, \cdots i + p - 1]$. In other words f is a substring of length p occurring at position i in word w.

A word has a repetition when it has at least two consecutive equal factors. More precisely, a repetition in w is defined as a triple (i, p, e) so that $w[i, \cdots i+p-1]$=$w[i+p, \cdots i+2*p-1] = \cdots = w[i+(e-1)*p, \cdots i+e*p-1]$. The integers p and e are called respectively the *period* and *exponent* of the repetition.

In the case that for a given position of a word w we consider the presence of a set of characters with a given probability of appearance each we define the sense of a weighted word w, defined as follow:

Definition 1. *A weighted word* $w = s_1 s_2 \cdots s_n$ *is a continuous set of couples* $(s, \pi_i(s))$, *where* $\pi_i(s)$ *is the probability of having the character* s *at position* i. *For every position* $1 \leq i \leq n$, $\Sigma\pi_i(s) = 1$.

For example, if we consider the DNA alphabet $\Sigma = \{A, C, G, T\}$ the word w= [(A,0.5),(C,0.25),(G,0.25),(T,0)] [(A,0),(C,1),(G,0),(T,0)] [(A,1),(C,0),(G,0),(T,0)],

represents a word having three letters: the first one is either A,C,G with respective probabilities 0.5, 0.25 and 0.25, the second one is always a C, while the third letter is necessarily an A, since its probability of presence is 1. That means that in a given biological sequence one of the following words: ACA, CCA, GCA might appear with probability 0.5, 0.25 and 0.25 each. We observe that the probability of presence of a word is the cumulative probability which is calculated by multiplying the relative probabilities of appearance of each character in every position. For the above example the probability of the word ACA to appear in positions 1 to 3 can be analyzed as follows: $\pi(ACA) = \pi_1(A) * \pi_2(C) * \pi_3(A)$=0.5*1*1=0.5. The definition of a weighted factor can be easily extended.

A weighted sequence has a repetition when it has at least two identical occurrences of a factor (weighted or not). The probability of appearance of the factor may vary according to the position it appears. In biological problems scientists are interested in discovering all the repetitions of all possible words having a probability of appearance larger than a predefined constant.

2.2 Equivalent Classes of Repetitions

In our methodology, in order to record the repetitions of all possible words we use a list $(L_p)_{p\geq 1}$ of equivalent repetitions of length p on the positions of a weighted sequence, defined as follows:

Definition 2. *Let x be a weighted sequence of length $|x|$=n; then $(i, j) \in L_p$ iff $i+p \leq n$, $j+p \leq n$ and $x_i \cdots x_{i+p-1} = x_j \cdots x_{j+p-1}$, while $\pi(x_i \cdots x_{i+p-1}) \geq 1/k$ and $\pi(x_j \cdots x_{j+p-1}) \geq 1/k$.*

So, two positions in x are equivalent when the factors of x of length p starting at i and j respectively are equal although the respective probabilities of appearance can vary. The positions of appearance of the factors as well as the respective probabilities are stored in a set of classes C^p.

Definition 3. *Let x be a weighted sequence of length $|x|$=n; then the (C^p_f) class is the ordered list of at least 2 couples $(i_f, \pi_i(f))$, which includes all positions of appearance of the factor f of length p in the weighted sequence. We exclude all couples with probability less than $1/k$.*

Moreover we also define a function on the positions of x, which gives for every position the next position in the same equivalence class.

Definition 4. *$D_p(i)$ = the least integer $k > 0$, so that $(i, i + k) \in L_p$.* (If there is no such k the function is not defined).

One can easily check that any list L_{p+1} is a refinement of L_p ($L_{p+1} \leq L_p$), since list L_{p+1} contains all possible repetitions of length p that can be extended by one character. Furthermore there clearly exists a smallest integer N, $1 \leq N \leq n$, so that $L_1 \geq L_2 \cdots \geq L_N$. Thus the computation of the equivalences L_p can be done using the values of L_{p-1}, the respective classes C^{p-1} and a proper *choicefunction* f.

Definition 5. *A choicefunction f is a function*

$f : \{C'_1, \cdots, C'_k\} \longrightarrow \{C_1, \cdots, C_k\}$, with the properties: for any $C' \in \{C'_1, \cdots, C'_k\}$ $[f(C') \subset C'$ and for any $C \in \{C_1, \cdots, C_k\} C \subset C' \Longrightarrow |C| \leq |f(C')|]$,

where $\{C'_1, \cdots, C'_k\}$ and $\{C_1, \cdots, C_k\}$ the equivalence classes of L_{p-1} and L_p respectively.

So f associates to each $E_{p-1} - class$ one of its $E_p - subclasses$ of maximal size. Given a choice function f, each L_p class $f(C')$ is called a big_class; the others are called $small\_classes$. By definition, all the L_1-classes are small.

Now we define a new sequence $(S_p)_{p\geq 1}$ of equivalences on the positions of x as follows:

Definition 6. $(i,j) \in S_p$ *iff for any small class* L_p*-class* C^p, $i \in C^p$ *iff* $j \in C^p$.

Lemma. For any $p \geq 1$, $(i,j) \in L_{p+1}$ iff $(i,j) \in L_p$ and $(i+1, j+1) \in S_p$.

For more information on the proof of the Lemma the reader can refer to [Cro81].

3 Computing the Repetitions

In this paper we address the problem of computing the set of repetitions in a weighted biological sequence. More formally the problem can be stated as follows:

Problem *Given a weighted sequence* X *and an integer* k *find all the repetitions of all possible words having a probability of appearance larger than* $1/k$.

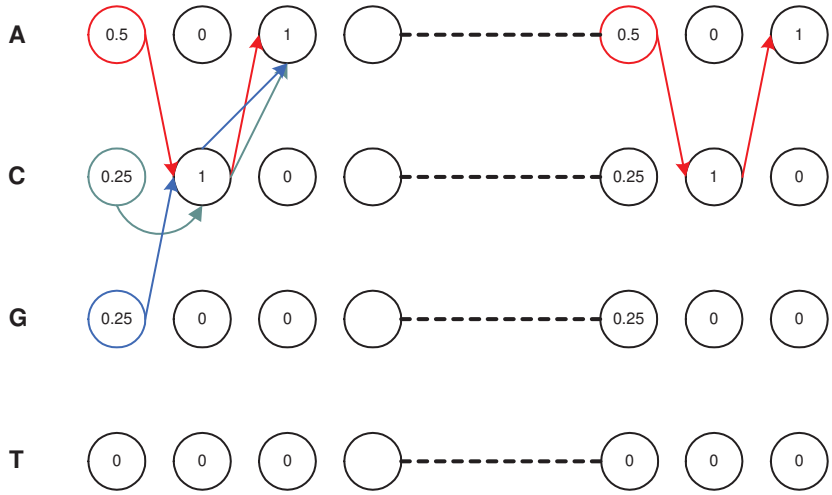

Figure 1: Graphical approach of the problem.

In a graphical approach the problem can be represented as in the Figure 1. For each position of the weighted sequence we write down the probability of appearance of each character of the alphabet. For the DNA alphabet which constitutes of 4 characters we write down 4 respective probabilities. The probability of appearance of a word is the cumulative probability calculated following the respective directed path.When the probability is larger than $1/k$, the directed path is a schema that can be extended by one character, in the following step and graphically we search for a repeated schema. In the above Figure the *red directed path* has a probability of appearance larger than $1/2$, $(k{=}2)$ thus we search for such repeated schemas.

Solution. *For every character* s *in the alphabet we define a class* C^1 *as the ordered list of couples* $(i_s, \pi_i(s))$*, which includes all equivalent positions of appearance of the character* s *in the weighted sequence. We exclude all couples with probability less than* $1/k$*. The set of* C^1 *classes forms the* L_1 *list for all possible repetitions of length one. We continue by computing* D_1 *for each position in the sequence.All* L_1*-classes are small. The process is continued by computing all* C^p *classes for* $p \geq 2$

and updating L_p thus forming D_p. The process stops when we reach the maximal (in length) repeated words with probability of appearance larger than $1/k$.

The above solution uses ideas from the algorithm presented by Crochemore (see [Cro81]). The major difference is the *choice function* that we have used in order to incorporate the notion of probability of appearance in repetitions. A schema of the algorithm is presented below.

FIND-WEIGHTED REPETITIONS(X,k)
Compute all possible repetitions of any length with probability larger than $1/k$
FOR all $s \in \Sigma$ **DO**
  create the small classes C^1 of couples $(s, \pi_i(s))$,
  where $\pi_i(s)$ is the probability of having the character s at position i.
  IF $\pi_i(s) \leq 1/k$ exclude it from the respective class
Compute for $p = 1$ L_p and D_p;
WHILE $\bigcup small\_classes \neq 0$, **DO**
  report the repetitions of period p.
  $p \longleftarrow p + 1$; if $p > |x|/2$ return repetitions;
  $L_p \longleftarrow L_p \cap S_p$; update D_p;
  $small\_classes \longleftarrow$ {indices of small $L_p - classes$}
END FIND-WEIGHTED REPETITIONS

Example Suppose we want to find all repetitions of the weighted sequence: $X=$ ACTT[(A,0.5),(C,0.5)]TC[(A,0.5),(C,0.3),(T,0.2)]TTT, with probability larger than $1/4$. We will illustrate the steps following the above presented algorithm.

1. For all characters $s \in \Sigma_{DNA} = \{A, C, G, T\}$ create the C^1 classes.
   $C^1_A = (1_A, 1)(5_A, 0.5)(8_A, 0.5)$.
   $C^1_C = (2_C, 1)(5_C, 0.5)(7_C, 1)(8_C, 0.3)$.
   $C^1_G = empty$.
   $C^1_T = (3_T, 1)(4_T, 1)(6_T, 1)(9_T, 1)(10_T, 1)(11_T, 1)$.

2. Define L_1 class as the union of C^1 classes and the values D_1.
   $L_1 = C^1_A \cup C^1_C \cup C^1_T$.
   $D_1 = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$.

3. Since $\bigcup small\_classes \neq 0$ we will compute all possible repetitions of length $p \geq 2$, using the lemma we presented in subsection 2.2.
   $C^2_{AT} = (5_{AT}, 0.5)(8_{AT}, 0.5)$.
   $C^2_{CT} = (2_{CT}, 1)(5_{CT}, 0.5)(8_{CT}, 0.3)$.
   $C^2_{TC} = (4_{TC}, 0.5)(6_{TC}, 1)$.
   $C^2_{TT} = (3_{TT}, 1)(9_{TT}, 1)(10_{TT}, 1)$.

4. Define L_2 class as the union of C^2 classes and the values D_2.
   $L_2 = C^2_{AT} \cup C^2_{CT} \cup C^2_{TC} \cup C^2_{TT}$.
   $D_2 =$ {not defined, 1, 1, 1, 1, 2, not defined, 1, 1, not defined, not defined}.

5. Following the above procedure we conclude that the repetitions with probability larger than 1/k are:.
   $L_3 = C^3_{CTT} = (2_{CTT}, 1)(8_{CTT}, 0.3)$

Theorem The above algorithm computes all repetitions in a weighted sequence X of length $|n|$.

Proof. It is easy to see that the algorithm stops. The length of L_1 in the algorithm is bounded by $O(|\Sigma|^{|X|})$. As far as it concerns the values of the list L_p for $p \geq 2$, are computed using the Lemma in subsection 2.2 and the values of L_{p-1} list. Each list of repetitions $p+1$ is at most half the size of the list of repetitions of length p.

4 Time Complexity Analysis

The time complexity analysis of our algorithm is based on the combination of the following two facts:

1. The well known "smaller-half trick" used also in [Cro81], [Apo83], [Sto98], for finding tandem repeats. According to the "smaller-half trick" each list of repetitions of length $p+1$ is at most half the size of the list of repetitions of length p.

2. The probability of existence of a factor f in a weighted sequence X is the cumulative probability which is calculated by multiplying the relative probabilities of appearance of each character/symbol in every position. Note that we interested in repetitions with probability greater than $1/k$. It is not difficult to see that given a position i of x, then there is only a constant number of different substrings that can occur at position i with probability greater than $1/k$. (The proof follows).

   For every weighted sequence w of length n, $w[1..n] = w[1]w[2]\cdots w[n]$, each position $w[i]$ for $1 \leq i \leq n$, is the starting position of a weighted *factor* iff the respective character s has $\pi(s_i) \geq 1/k$. Therefore the maximum probability of appearance for the rest of the characters in position i is bounded by $p = 1-1/k$. Assume that the number of starting positions inside a weighted factor, produced from position i is l. In order this *factor* to be interesting its probability of appearance must be grater than $1/k$. This is mathematically formulated as follows:

   $p^l \geq 1/k \longrightarrow l \leq log_p(k)$.

   That means that the number of weighted positions inside a *weighted factor* is bounded by a constant and thus the number of different substrings that can occur at position i with probability greater than $1/k$ is also a constant number.

Based on the above two facts the time complexity of our algorithm for computing the set of repetitions in a weighted sequence with probability of appearance larger than $1/k$ is $O(n \log n)$.

5 Conclusions

Our future direction is focused on defining the notion of borders for a weighted sequence and developing efficient algorithms for computing the covers and the seeds of weighted sequences.

Moreover we are studying the same problem using the suffix tree as the fundamental data structure. The basic idea behind this approach is to incorporate the notion of probability of appearance in the path labels and in the leaves in the suffix tree of a weighted sequence [Ili03].

Another potential application of our algorithm is in defining a basis for the repeated motifs of a weighted sequence. In our algorithm we create in an exhaustive way all possible repetitions with probability larger than $1/k$. We can use all primitive repetitions and a set of allowed operations in order to define a basis that efficiently produces all repeated motifs. As any repeated word can be expressed as an array of primitive repetitions, it is often desirable to find only primitive repetitions.

References

[Cro81] Crochemore, M.: An Optimal Algorithm for Computing the Repetitions in a Word. Information Processing Letters, Vol.12 (5), (1981) 244-250.

[Celera1] Celera Genomics: The Genome Sequence of Drosophila melanogaster, Science 287, (2000) 2185-2195

[Celera2] Celera Genomics: The Sequence of the Human Genome, Science 291, (2001) 1304-1351.

[IUB] Nomenclature Committee of the International Union of Biochemistry (NC-IUB). Nomenclature for incompletely specified bases in nucleic acid sequences, Eur. J. Biochem. 150(1985) 1-5.

[SS90] Schneider T. D., Stephens R. M.: Sequence Logos: A New Way to Display Consensus Sequences, Nucleic Acids Res. 18, (1990) 6097-6100.

[Knu77] Knuth, D.E., Morris, J.H., Pratt, V.R.: Fast pattern matching in strings, SIAM J. Comput., (6), (1977) 322-350.

[Apo83] Apostolico, A., Preparata, F.P.,: Optimal off-line detection of repetitions in a string. Theoretical Computer Science, (22), (1983) 297-315.

[Mai84] Main, M.G., Lorentz, R.J.,: An $O(nlogn)$ algorithm for finding all repetitions in a string. Journal of Algorithms, (5), (1984) 422-432.

[Sto98] Stoye, J., Gusfield, D.,: Simple and flexible detection of contiguous repeats using a suffix tree. In proceedings of the 9th Annual Symposium on Combinatorial Pattern matching (CPM), volume 1448 of Lecture Notes in Computer Science, (1998) 140-152.

[Kol99a] Kolpakov, R., Kucherov, G.,: Finding maximal repetitions in a word in linear time. Proceedings of IEEE Foundations of Computer Science, (1999).

[Kol99b] Kolpakov, R., Kucherov, G.,: On maximal repetitions in words. Proceedings of Foundamentals of Computation Theory, (1999) 374-385.

[Mar83] Martinez, H.,: An Efficient Method for Finding Repeats in Molecular Sequences. Nucleic Acid Research, (11), (1983) 4626-4634.

[Tsu99] Tsunoda, T., Fukagawa, M., Takagi, T.,: Time and memory efficient algorithm for extracting palindromic and repetitive subsequences in nucleic acid sequences. Pacific Symposium on Biocomputing, (4), (1999) 202-213.

[Kur99] Kurtz, S., Schleiermacher, C.,: REPuter: fast computation of maximal repetas in complete genomes. Bioinformatics, (15), (1999) 426-427.

[Ili03] Iliopoulos, C., Makris, Ch., Panagis, I., Perdikuri, K., Theodoridis, E., Tsakalidis, A.,: Computing the Repetitions in a Weighted Sequence using Weighted Suffix Trees. European Conference On Computational Biology (ECCB 2003), (accepted).