A Linear Algorithm for the Detection of Evolutive Tandem Repeats

Richard Groult*[1], Martine Léonard[1] and Laurent Mouchard†[2]

[1] LIFAR - ABISS, Faculté des Sciences, 76821 Mont Saint Aignan Cedex, France

[2] UMR 6037 - ABISS, Faculté des Sciences, 76821 Mont Saint Aignan Cedex, France and Dept. Computer Science, King's College London, London WC2R 2LS, England

e-mail: {Richard.Groult,Martine.Leonard,Laurent.Mouchard}@univ-rouen.fr

Abstract. We present here a linear algorithm for the detection of evolutive tandem repeats. An evolutive tandem repeat consists in a series of almost contiguous copies, every copy being similar (using Hamming distance in this article) to its predecessor and successor. From a global view point, evolutive tandem repeats extend the traditional approximate tandem repeat where each copy has to be in a neighborhood of a given model. Due to the lack of algorithms, these repeats have been discovered in genomic sequences only recently. In this article, we present a two-stage algorithm, where we first compute an array containing all the Hamming distances between candidates, then we visit this array to build a complete evolutive tandem repeat from insulated pairs of copies. Moreover, we explain how it is still consistent with the usual technique devoted to dynamic programming which consists in filling a comparison matrix and backtracking through it to find an optimal alignment.

Keywords: linear algorithm, evolutive tandem repeats, Hamming distance

1 Introduction

The notion of approximate tandem repeat is generally well-defined, from the formal view point [2, 12], it uses a consensus model, every copy participating to this repeat being very similar to the consensus. An *evolutive tandem repeat* has no need for a consensus model, the first and the last copies might be completely different but every time we are considering two successive copies participating to the repeat, they are very similar to each other: finding evolutive tandem repeats is obviously much more complicated than detecting generic tandem repeats for which usual well-known structures, such as suffix trees, can be used during a preprocessing stage [9].
Evolutive tandem repeats have been phrased by molecular biologists, for example in [4], and have been observed in real DNA sequences (see Appendix A for a complete example, detected in *A. thaliana*). In [5], we gave a formal definition of evolutive tandem repeats with jumps then we described a quadratic space and time algorithm

*Supported by a French Ministry of Research grant.

†Partially supported by Programme inter-EPST Bio-informatique and by GenoGRID (ACI GRID).

which detects all the maximal. Even if numerous models and algorithms searching for various kinds of repeats have been developed [1, 3, 10, 11, 8, 12], none of these algorithms are able to locate evolutive tandem repeats, as far as we know, we therefore designed a quadratic algorithm for their detection, it was based on the construction of two graphs and their visits.
Since we are looking for local repetitions having approximatively the average length of mini (or even micro) satellites and because we are also looking for a certain number of copies (having three or less copies in an evolutive tandem repeats is meaningless), we are here interested in searching for copies whose length may vary from 4 to 64 [6], that is usually thousands times less than the size of the sequences we are studying. We present in this article a $O((\ell_{max} - \ell_{min} + 1) \times (j_{max} - j_{min} + 1) \times |w|)$-time and $O(j_{max} - j_{min} + 1)$-space algorithm where and ℓ_{min} and ℓ_{max} (resp. j_{min} and j_{max}) are the minimal and maximal values of the length of the copies (resp. the jump between two copies) and w is the studied sequence. More precisely, since length and jump values are very small (with respect to the length of the sequence which can be counted in millions of base pairs), we still have an overall linear time-complexity. So in practice, the time complexity is in $O(C \times |w|)$, where $C \leq (61 \times (j_{max} - j_{min})$.
In section 2, we recall some basic definitions and introduce the evolutive tandem repeats. In section 3, we present the ideas of our algorithm. In section 4, we explain the connection with comparison matrices. In section 5, we present experimental results and finally, in section 6, we conclude.

2 Preliminaries

Let Σ be an alphabet and Σ^* its associated free monoid. A *word* (resp. *non empty word*) over Σ is an element of Σ^* (resp. Σ^+). The letter of a word w occurring at position i is denoted by w_i. The *length* $|w|$ of a word w is the number of letters of w, i.e. $w = w_1 \cdots w_{|w|}$. We will denote by Σ^ℓ the set of all possible words of length ℓ over Σ. We denote by $u.v$ (or simply uv) the concatenation of two words u and v. Consider $w = p.f.s$ for some $p, f, s \in \Sigma^*$. Such p, f, s are respectively *prefix*, *factor* and *suffix* of w. We denote $f = w[i, j] = w_i w_{i+1} \cdots w_{j-1} w_j$ for $1 \leq i \leq j \leq |w|$. The concatenation of n copies of u is denoted by u^n.
There exist several distances one can use for the analysis of genomic sequences. In this article, we will consider the *Hamming distance*: the Hamming distance between two words of equal length is the number of positions at which their corresponding letters differ: for $u, v \in \Sigma^\ell$, $d_H(u, v) = \text{Card}\{i \in \{1, \ldots, \ell\} \mid u_i \neq v_i\}$.

Definition 2.1 (Evolutive tandem repeat)
An *evolutive tandem repeat with jumps* (e.t.r. for short) is a tuple $(v, \varepsilon, (j_{min}, j_{max}), \ell, n, (p_i)_{1 \leq i \leq n})$ where v is a word, ε is the maximal number of errors between two consecutive copies, $[j_{min}, j_{max}]$ is the range of the length of a jump (overlap or gap between two consecutive copies) with $(j_{max} - j_{min} + 1) \leq \ell/2$, ℓ is the length of the copies, n is the number of copies, p_i are the starting positions of the copies $c_i = v[p_i, p_i + \ell - 1]$ and

$$\begin{cases} p_1 = 1,\ p_n + \ell - 1 = |v|, \\ j_{min} \leq p_{i+1} - (p_i + \ell) \leq j_{max},\ \forall i \in \{1, \ldots, n-1\}, \\ d_H(c_i, c_{i+1}) \leq \varepsilon,\ \forall i \in \{1, \ldots, n-1\}. \end{cases}$$

Example 2.1 Let consider the word $v = \underline{aaa}t\overline{aac}\underline{agc}\overline{cgc}$.
$(v, 1, (-1, 1), 3, 4, (1, 5, 8, 10))$ is an e.t.r. with jumps: $p_1 = 1$, $p_2 = 5$ (gap), $p_3 = 8$ and $p_4 = 10$ (overlap) corresponding to $c_1 = aaa$, $c_2 = aac$, $c_3 = agc$ and $c_4 = cgc$ (see FIG. 1).

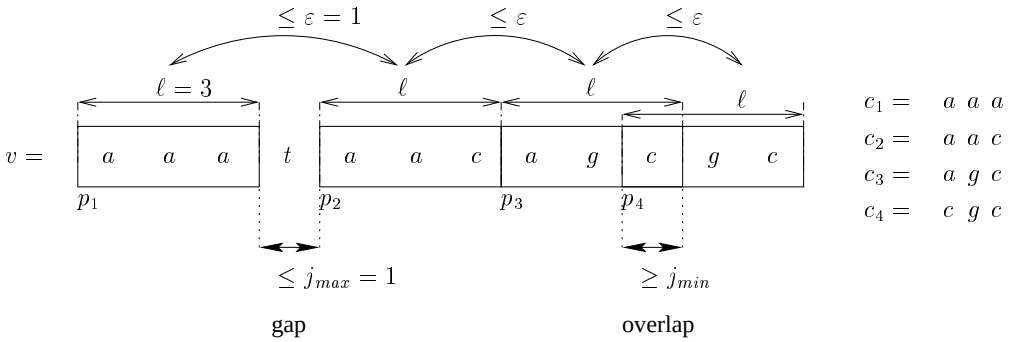

FIG. 1: Example of an evolutive tandem repeat with jumps

We will consider only in what follows maximal e.t.r., that is e.t.r. which is not embedded in a longer one: consider for example a word $w = gaaagacgaggcgg$ and $\ell = 3$. The e.t.r. $etr_1 = (\underline{aag}\overline{acg}\underline{agg}, 1, (-1, 1), 3, 3, (1, 4, 7))$ is not maximal in w since the repeat $etr_2 = (\underline{aag}\overline{acg}\underline{agg}\overline{cgg}, 1, (-1, 1), 3, 4, (1, 4, 7, 10))$ contains more copies. In this case, we say that etr_2 "contains" etr_1 and remark that etr_2 is a maximal e.t.r. in w.
In a previous article [5], we first considered all factors of w having the same length. For each factor, we computed the set of its starting positions using an equivalence relation on positions in w. Then, we built a graph for which nodes are these sets and there exists an edge between two nodes if the corresponding factors are slightly different in the meaning of the Hamming distance. Next, we computed a second graph namely the ℓ-position graph defined as follows:

Definition 2.2 (ℓ-position graph) Let w be a word and ε and $jump$ integers. The *ℓ-position graph* corresponding to w, ε and $jump$ is the oriented graph $PG_\ell(w, \varepsilon, jump) = (N, E)$ where

$$\begin{cases} N = \{1, ..., |w| - \ell + 1\} \text{ and} \\ E = \{(i, i', i' - (i + \ell)) \text{ for } (i, i') \in N \times N,\ i < i' \\ \qquad \text{such that } |i' - (i + \ell)| \leq jump, \\ \qquad d_H(w[i, i + \ell - 1], w[i', i' + \ell - 1]) \leq \varepsilon\}. \end{cases}$$

Nodes are labeled with all the positions $\{1, \ldots, |w| - \ell + 1\}$ of factors of length ℓ and there exists an edge labeled with d between two nodes if the corresponding positions are close in w and if the Hamming distance between their associated factors, denoted d is not greater than a given ε. We used a quadratic time but linear space algorithm to compute it. In what follows we denote by (i, i', d) an edge labeled d from the node i to the node i'.
Finally, we looked for all the longest paths in the ℓ-position graph to find maximal e.t.r.

3 A Linear – Time and Space – Algorithm

In a previous article [5], we described a quadratic space and time algorithm which detects all maximal e.t.r. in a word w. In what follows, we present a linear time and space algorithm that starts with the filling of a "position" array and follows on with the visit of this array in an attempt to find regularities. We will first draw the "big-picture" and will consolidate the description by explaining the structures we used and the strategies we developed.
The first important idea consists in considering every ℓ-mer (factor of length ℓ) as a sliding window. Since we have to compute the distances between pairs of factors, we have to use two sliding windows f and f' (see FIG. 2): one window, f', ending at position i will correspond to the right-most factor (moving sequentially from left to right, one position at a time) while the other window, f, will correspond to the candidates for a pair (ending at a position in the interval $[i-\ell-j_{max}, i-\ell-j_{min}]$). Therefore, we only have to consider $j_{max}-j_{min}+1$ possible positions for the left sliding window, for each given position of the right sliding window and focus on the computation of $(j_{max}-j_{min}+1)\times(|w|-\ell+1)$ distances, that is a linear-time and space construction of a "position" array (emulating the position graph we defined in [5]).

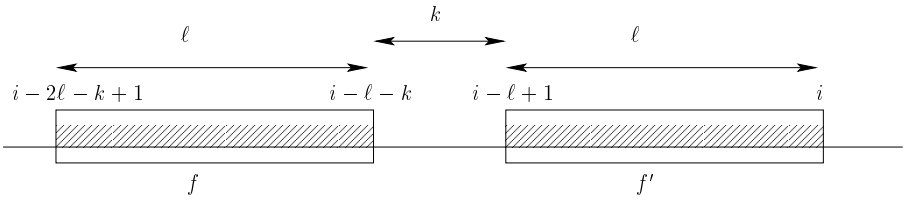

FIG. 2: The two sliding windows f and f'

The second important idea is the computation of the Hamming distance by itself: if the Hamming distance between the factors of length ℓ ending at position i and i' is known then the Hamming distance between the factors ending at position $i+1$ and $i'+1$ can be computed in $O(1)$-time because $(\ell-1)$ comparisons have already been done. It will speed up the filling of the position array (see FIG. 3).

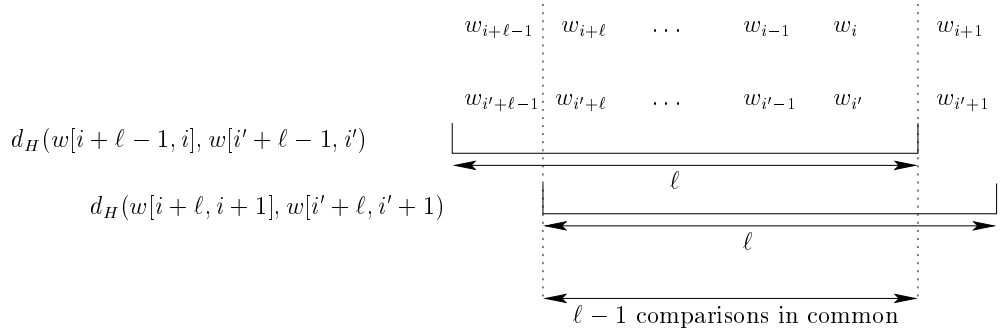

FIG. 3: Computing Hamming distance on incremental positions

Finally we only have to visit the position array and search for a series of acceptable values (smaller than ε) located at appropriate positions (the distance between two consecutive positions has to belong to $[\ell+j_{min}, \ell+j_{max}]$).

A Two-stage Algorithm

We first have to compute the Hamming distances between every possible pairs of candidates and fill the position array D that contains all these computations.

Definition 3.1 Let $w = w_1 \dots w_n$ be a word over Σ, ℓ an integer and $k \in \{j_{min}, \dots, j_{max}\}$. We define $D_k^{w,\ell}(i)$ by

$$D_k^{w,\ell}(i) = \begin{cases} 0, & \forall i \in \{1, \dots, \ell + k\} \\ d_H(w[1, i-\ell-k], w[\ell+k+1, i]), & \forall i \in \{\ell+k+1, \dots, 2\ell+k-1\} \\ d_H(w[i-2\ell-k+1, i-\ell-k], w[i-\ell+1, i]), & \forall i \in \{2\ell+k, \dots, |w|\} \end{cases}$$

We assume now that $D_k^{w,\ell}(i-1)$ has been previously computed and we would like to compute $D_k^{w,\ell}(i)$, i.e we know $d_H(w[i-2\ell-k, i-\ell-k-1], w[i-\ell, i-1])$ and we would like to compute $d_H(w[i-2\ell-k+1, i-\ell-k], w[i-\ell+1, i])$.

We therefore define two additional functions:

- $\forall a, b \in \Sigma,\ \mathbb{1}_a(b) = 0$ if $b = a$, 1 otherwise;
- $\forall k \in \{j_{min}, \dots, j_{max}\},\ E_k^{w,\ell}(i) = \mathbb{1}_{w_{i-\ell-k}}(w_i)$ if $i \in \{\ell+k+1, \dots, |w|\}$, 0 otherwise.

Lemma 3.1 Let w be a word over Σ, ℓ an integer and $k \in \{j_{min}, \dots, j_{max}\}$. We have:

$$D_k^{w,\ell}(i) = \begin{cases} 0, & \forall i \in \{1, \dots, \ell+k\}, \\ D_k^{w,\ell}(i-1) + E_k^{w,\ell}(i), & \forall i \in \{\ell+k+1, \dots, 2\ell+k-1\}, \\ D_k^{w,\ell}(i-1) + E_k^{w,\ell}(i) - E_k^{w,\ell}(i-\ell), & \forall i \in \{2\ell+k, \dots, |w|\}. \end{cases}$$

Proof 1 Let $k \in \{j_{min}, \dots, j_{max}\}$ and $i \in \{2\ell+k, \dots, |w|\}$. If $i > 2\ell + k$ then $D_k^{w,\ell}(i-1) = d_H(w[i-2\ell-k, i-\ell-k-1], w[i-\ell, i-1])$ and therefore

$$\begin{aligned} & D_k^{w,\ell}(i) \\ &= d_H(w[i-2\ell-k+1, i-\ell-k], w[i-\ell+1, i]) \\ &= d_H(w[i-2\ell-k+1, i-\ell-k-1], w[i-\ell+1, i-1]) + \mathbb{1}_{w_{i-\ell-k}}(i) \\ &= d_H(w[i-2\ell-k, i-\ell-k-1], w[i-\ell, i-1]) - \mathbb{1}_{w_{i-2\ell-k}}(i-\ell) + \\ &\quad \mathbb{1}_{w_{i-\ell-k}}(i) \\ &= D_k^{w,\ell}(i-1) - E_k^{w,\ell}(i-\ell) + E_k^{w,\ell}(i). \end{aligned}$$

If $i = 2\ell + k$ then $D_k^{w,\ell}(i) = d_H(w[1, i-\ell-k], w[\ell+k+1, i]) = d_H(w[1, i-\ell-k-1], w[\ell+k+1, i-1]) + \mathbb{1}_{w_{i-\ell-k}}(w_i) = D_k^{w,\ell}(i-1) + E_k^{w,\ell}(i)$.
But we have $E_k^{w,\ell}(i-\ell) = E_k^{w,\ell}((2\ell+k)-\ell) = E_k^{w,\ell}(\ell+k) = 0$, so $D_k^{w,\ell}(i) = D_k^{w,\ell}(i-1) - E_k^{w,\ell}(i-\ell) + E_k^{w,\ell}(i)$.
We prove the other case in the same manner. □

The size of the arrays D (where $D[k][i] = D_k^{w,\ell}(i)$) and E (where $E[k][i] = E_k^{w,\ell}(i)$) is $(j_{max} - j_{min} + 1) \times |w|$. In order to fill these two arrays, we now use a $O((j_{max} - j_{min} + 1) \times |w|)$-time and space algorithm.

Example 3.1
This example (see FIG. 4) has been obtained with $w = aaataagttatcaatccaaatcgtgtca$, $\ell = 4$, $j_{min} = -1$, $j_{max} = 1$ and $\varepsilon = 2$:
For example $D_{-1}^{w,4}(7) = d_H(w[1,4], w[4,7]) = d_H(aaat, taag) = 2$, $D_0^{w,4}(17) = d_H(w[10, 13], w[14, 17]) = d_H(atca, atcc) = 1$ and $D_1^{w,4}(28) = d_H(w[20,23], w[25,28]) = d_H(atcg, gtca) = 2$.

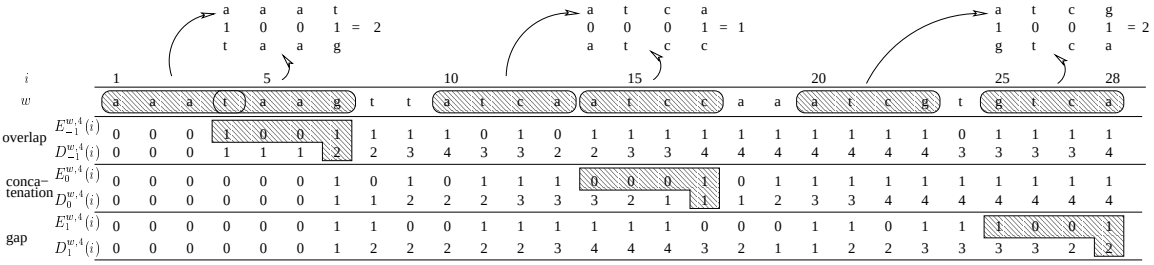

FIG. 4: D and E arrays

The space complexity can be improved as follows.
Since the values $E[k][i]$ are independent, we can decrease the space complexity by ignoring the filling of the array E and by computing $E[k][i]$ only when needed without increasing the time complexity.
Moreover, for a given ℓ, we only need the last value $D_k^{w,\ell}(i-1)$ in order to compute $D_k^{w,\ell}(i)$ (see Lemma 3.1), thus we will only store the last column of the array D. Finally (see FIG. 5), we obtain a $O((j_{max} - j_{min} + 1) \times |w|)$-time and $O(j_{max} - j_{min} + 1)$-space algorithm (D is an array of size $O(j_{max} - j_{min} + 1)$). If we are looking for all e.t.r. for copies of length $\ell \in [\ell_{min}, \ell_{max}]$,the complexity is $O((\ell_{max} - \ell_{min} + 1) \times (j_{max} - j_{min} + 1) \times |w|)$. From a practical point of view, $(\ell_{max} - \ell_{min} + 1) \leq 61$ is much lower than $|w|$ and the time complexity is still linear: $O(C \times |w|)$, where $C \leq 61 \times (j_{max} - j_{min})$.

Construction of the Longest Paths

The two arrays are compact representations of the graphs we depicted in [5], and if we refer to the traditional graph vocabulary, we can associate a cell in the position array and a node in the position graph.

CONSTRUCTION OF THE ARRAY CONTAINING THE LONGEST PATHS$(w, \ell, j_{min}, j_{max}, \varepsilon)$

```
1  for ℓ ← ℓ_min to ℓ_max do
2   for i ← 1 to |w| do
3    C[i] ← −1
4    L[i] ← 0
5    for k ← j_min to j_max do
6     if (i ≤ ℓ + k) then
7      D[k] ← 0
8     elseif (i ≤ 2ℓ + k) then
9      D[k] ← D[k] + 𝟙_{w_{i−ℓ−k}}(w_i)
10    else D[k] ← D[k] + 𝟙_{w_{i−ℓ−k}}(w_i) − 𝟙_{w_{i−2ℓ−k}}(w_{i−ℓ})
11    if (i ≥ 2ℓ + k) and (D[k] ≤ ε) and (L[i − 2ℓ − k + 1] + 1 > L[i − ℓ + 1]) then
12     L[i − ℓ + 1] ← L[i − 2ℓ − k + 1] + 1
13     C[i − ℓ + 1] ← i − 2ℓ − k + 1
14 return (C, D)
```

FIG. 5: Construction of the array containing the longest paths

When $D_k^{w,\ell}(i) \leq \varepsilon$ and $i \geq 2\ell + k$, the arc between nodes $(i - 2\ell - k + 1)$ and $(i - \ell + 1)$ is added only if it creates a longest path to node $(i - \ell + 1)$, moreover the previously

existing, previously unique arc ending in $i-\ell+1$ is removed: let a path of length c ending in $(i-\ell+1)$, if the length of the path ending in $(i-2\ell-k+1)$ plus 1 is greater than c, then thearc ending in $(i-\ell+1)$ is removed and the arc from $(i-2\ell-k+1)$ to $(i-\ell+1)$ is created.
Finally each node i has at most one arc ending in i and therefore the ℓ-position graph is stored in an array C of integers, where $C[i]$ is the index of the head of the arc $(C[i], i)$, and -1 otherwise. We use an array L of integers, where $L[i]$ is the length of the longest path ending in i.
Let C and L be arrays of integers of size $|w|$ (see algorithm FIG. 5).
The determination of the longest paths, corresponding to the maximal e.t.r., uses the traditional algorithm.

Computation of the Distance between Two Factors of Length $\ell+1$

Lemma 3.2 (Computation of $D_k^{w,\ell+1}(i)$) Let ℓ, j_{min}, j_{max} and k be integers. We have $\forall k \in \{j_{min}, \ldots, j_{max}\}$, $i \in \{2\ell+k, \ldots, |w|\}$, $D_k^{w,\ell+1}(i) = D_{k+1}^{w,\ell}(i) + E_{k+1}^{w,\ell}(i-\ell)$, (see FIG. 6).

Proof 2 Let $\ell, j_{min}, j_{max}, i$ and k integers such that $k \in \{j_{min}, \ldots, j_{max}\}$ and $i \in \{2\ell+k, \ldots, |w|\}$. We have

$$\begin{aligned} D_k^{w,\ell+1}(i) &= d_H(w[i-2(\ell+1)-j+1, i-(\ell+1)-k], w[i-(\ell+1)+1, i]) \\ &= d_H(w[i-2\ell-k-1, i-\ell-k-1], w[i-\ell, i]) \\ &= d_H(w[i-2\ell-k, i-\ell-k-1], w[i-\ell+1, i]) + \mathbb{1}_{w_{i-2\ell-k-1}}(w_{i-\ell}) \\ &= d_H(w[i-2\ell-(k+1)+1, i-\ell-(k+1)], w[i-\ell+1, i]) + \\ &\qquad \mathbb{1}_{w_{i-2\ell-k-1}}(w_{i-\ell}) \\ &= D_{k+1}^{w,\ell}(i) + E_{k+1}^{w,\ell}(i-\ell). \end{aligned}$$

□

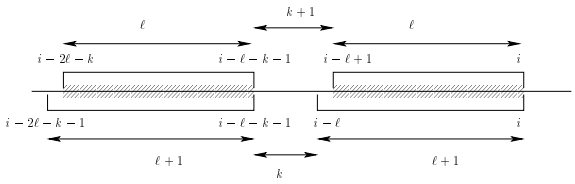

FIG. 6: Computation of $D_k^{w,\ell+1}(i)$

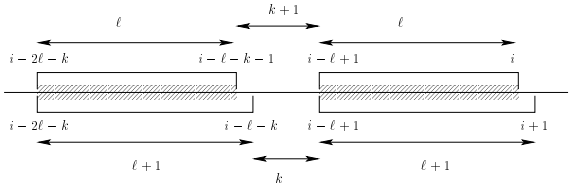

FIG. 7: Computation of $D_k^{w,\ell+1}(i+1)$

Lemma 3.3 (Computation of $D_k^{w,\ell+1}(i+1)$) Let ℓ, j_{min} and j_{max} be integers. We have $\forall k \in \{j_{min}, \ldots, j_{max}\}$, $i \in \{2\ell+k, \ldots, |w|\}$ $D_k^{w,\ell+1}(i+1) = D_{k+1}^{w,\ell}(i) + E_{k+1}^{w,\ell}(i+1)$, (see FIG. 7).

Proof 3 According to Lemma 3.2, $D_k^{w,\ell+1}(i+1) = D_{k+1}^{w,\ell}(i+1) + E_{k+1}^{w,\ell}(i-\ell+1)$ and by Definition 3.1, $D_{k+1}^{w,\ell}(i+1) = D_{k+1}^{w,\ell}(i) - E_{k+1}^{w,\ell}(i-\ell+1) + E_{k+1}^{w,\ell}(i+1)$, therefore, $D_k^{w,\ell+1}(i+1) = D_{k+1}^{w,\ell}(i) + E_{k+1}^{w,\ell}(i+1)$.

□

Lemma 3.4 (Computation of $D_k^{w,\ell+1}(i)$) Let ℓ, j_{min} and j_{max} be integers. We have $\forall k \in \{j_{min}, \ldots, j_{max}\}$, $i \in \{2\ell + k, \ldots, |w|\}$

$$\begin{aligned} D_k^{w,\ell+1}(i) &= D_{k+1}^{w,\ell}(i) + E_{k+1}^{w,\ell}(i-\ell) \\ &= D_{k+1}^{w,\ell}(i-1) + E_{k+1}^{w,\ell}(i). \end{aligned}$$

4 Evolutive Tandem Repeats and Comparison Matrices

Comparison Matrices

We will now explain the connection between the arrays we are computing and using, and well-known techniques used by several algorithms devoted to sequence comparison.

A traditional technique in sequence comparison consists in the construction and the visit of the two-dimension matrix, where a cell (i, i') contains the comparison score, i.e. the distance, between a factor ending at position i in one sequence and a factor ending at position i' in the other sequence.

Computing the positions of all the approximate repeats in one sequence can be carried out by comparing the sequence with itself, that is by constructing a specific symmetric square matrix, like the one we are presenting in Fig. 8. Note that Fig. 9 represents the arrays D and E corresponding to the three white diagonals of Fig. 8.

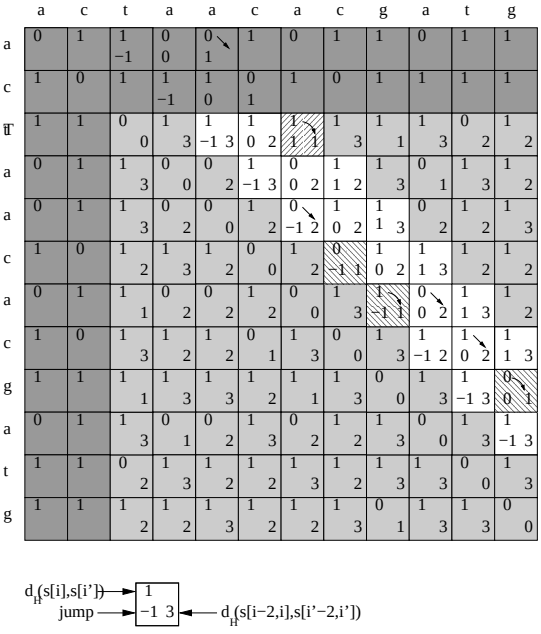

Fig. 8: Matrix and its diagonals for $\ell = 3$, $j_{min} = -1, j_{max} = 1$ and $\varepsilon = 1$

i	1					5					10	
w	a	c	t	a	a	c	a	c	g	a	t	g
$E_{-1}^{w,3}(i)$	0	0	1	1	1	1	0	0	1	1	1	1
$D_{-1}^{w,3}(i)$	0	0	1	2	3	3	2	3	3	2	3	3
$E_{0}^{w,3}(i)$	0	0	0	0	1	1	0	1	1	0	1	0
$D_{0}^{w,3}(i)$	0	0	0	0	1	2	2	2	2	2	2	3
$E_{1}^{w,3}(i)$	0	0	0	0	0	0	1	1	1	1	1	1
$D_{1}^{w,3}(i)$	0	0	0	0	0	0	1	2	3	3	3	3

Fig. 9: The arrays D and E corresponding to the three white diagonals

In this matrix, the content of a cell (i, i') contains informations corresponding to $d_H(w[i-2, i], w[i'-2, i'])$. One can observe four different kinds of cells: dark gray cells correspond to undefined distances ($i < \ell$ or $i' < \ell$, the factors are not long enough

to compute $d_H(w[i-2,i], w[i'-2,i'])$, therefore only $d_H(w[i], w[i'])$ is reported in the upper left corner), light gray cells correspond to useless cells such that $i'-i < \ell + j_{min}$ or $i'-i > \ell + j_{max}$, white cells contain three values as expressed in FIG. 8 and are the only cells that are really needed and finally dashed cells tick copies participating to a potential e.t.r. (for example, the dashed cell $(3,7)$ states that $d_H(w[1,3], w[5,7]) \leq \varepsilon$, that is $d_H(act, aca) \leq 1$, which is correct).

Remark 4.1 Dashed cells contributing to a diagonal indicate a potential larger repeat: $(3,9)$ and $(4,10)$ (corresponding respectively to $d_H(act,\ acg) \leq 1$ and $d_H(cta,\ cga) \leq 1$) can establish the existence of a longer repeat (in this example $d_H(acta,\ acga) \leq 1$) but more generally, dashed cells (i,i') and $(i+1,i'+1)$, that is $d_H(w[i-2,i], w[i'-2,i']) \leq 1$ and $d_H(w[i-1,i+1], w[i'-1,i'+1]) \leq 1$, does not imply necessarily that $d_H(w[i-2,i+1], w[i'-2,i'+1]) \leq 1$ (consider $(6,8)$ and $(7,9)$ for example).

Assume now that we are searching for *approximate* tandem repeats of length $\ell = 3$, with an error rate $\varepsilon = 1$ and $j_{min} = -1, j_{max} = 1$, once we have built our matrix, the hunt for the repeats can be carried out by visiting one row at a time and reporting regions containing cells with a lower right value smaller than ε every at least $\ell + j_{min} = 3 - 1 = 2$ and at most $\ell + j_{max} = 3 + 1 = 4$ positions. In this matrix (see Fig. 10), if we consider the third row, one can find such cells in columns 3, 7 and 9 and therefore deduce that there exists an approximate repetition starting at position 1 and ending at position 9: as a matter of fact, *actaacacg* is an approximate tandem repeat with jumps, the letter a located at position 4 corresponds to a gap between copies $c_1 = act$ and $c_2 = aca$, the letter a locates at position 7 corresponds to an overlap between copies $c_2 = aca$ and $c_3 = acg$. This is more or less the concept Sagot and Myers used in [12] for finding microsatellites.

Evolutive Tandem Repeats

Finding evolutive tandem repeats with jumps is slightly different, the location of a copy participating to the e.t.r. depends only on the location of its predecessor, ℓ, the length of the copies and j_{min}, j_{max} the acceptable jump between two consecutive copies.
Consider a copy belonging to the e.t.r. that ends at position i, its successor must ends at a specific position (between $i+\ell+j_{min}$ and $i+\ell+j_{max}$) in the matrix, we therefore have to search for a dashed cell at positions (i,i') for $i+\ell+j_{min} \leq i' \leq i+\ell+j_{max}$. If there exists such a cell, it gives us a significant information about the way the copies are connected: if $i+\ell+j_{min} \leq i' \leq i+\ell-1$ there is an overlap of length $i+\ell-i'$ between the copies, if $i' = i+\ell$ the copies are contiguous, if $i+\ell+1 \leq i' \leq i+\ell+j_{max}$ there exists a gap of length $i'-i-\ell$ between the copies. Therefore, for every row i, we only have to consider $(j_{max} - j_{min}) + 1$ cells. In order to find e.t.r. we therefore have to compute and visit the diagonals starting in columns $i+\ell+j_{min}$ to $i+\ell+j_{max}$. That leads to computing and visiting only $O((j_{max} - j_{min} + 1) \times |w|)$ cells.
The left-most diagonal, starting in cell $(1, \ell + j_{min} + 1)$, corresponds to the maximal authorized overlap, while the right-most diagonal, starting in cell $(1, \ell + j_{max} + 1)$, corresponds to the maximal authorized gap. We can therefore build a matrix that sums up all these informations as depicted in FIG. 8. The three white diagonals are

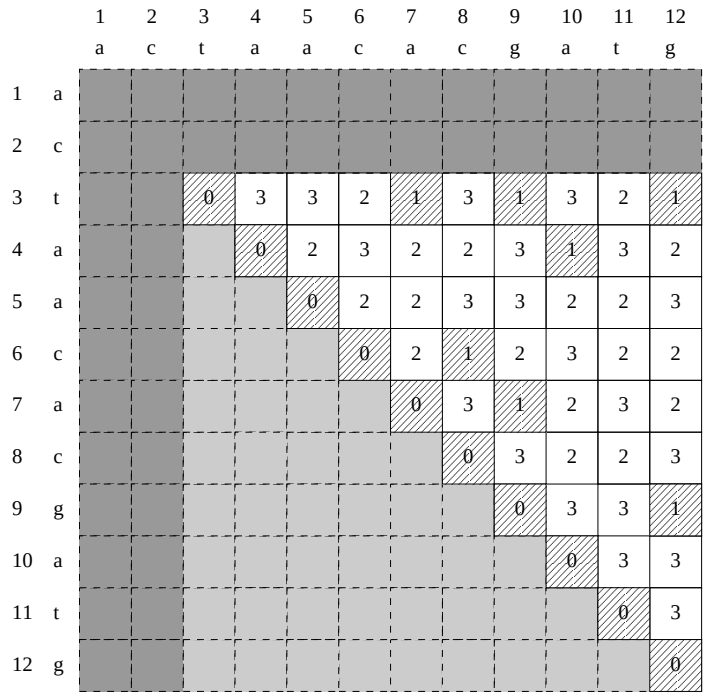

FIG. 10: Two dimension matrix corresponding to the comparison of *actaacacgatg* with itself, for $\ell = 3$ and $\varepsilon = 1$

the only ones that need to be computed (even if in this matrix, we show all the cells). Moreover, the computation of the three diagonals is equivalent to the computation of the D and E arrays.

5 Experimental Results

We have implemented and tested this algorithm on various sequences, we built random sequences over the alphabet $\{a, c, g, t\}$ and no e.t.r. has been detected (for the same rapameters as below), it appears that this kind of repetition is not an artifact. Moreover we focused on real sequences from *A. thaliana* and for testing purpose we used sequences with length varying from 10kb to 200kb (see FIG. 11).
The average behaviour of the timing curves corresponds to that we were expecting. Time and space consumptions enabled us to search for e.t.r. in whole chromosomes, we studied more specifically *A. thaliana* which possesses five chromosomes (their length varying from 17 to 29Mb) and an example is presented in Appendix A.

6 Conclusion and Perspectives

In this article, we presented a both space and time linear algorithm for the detection of evolutive tandem repeats. Furthermore, we implemented this approach, developed a web interface (see FIG. 12, `http://abiss.crihan.fr/~rgroult/index.php`) that

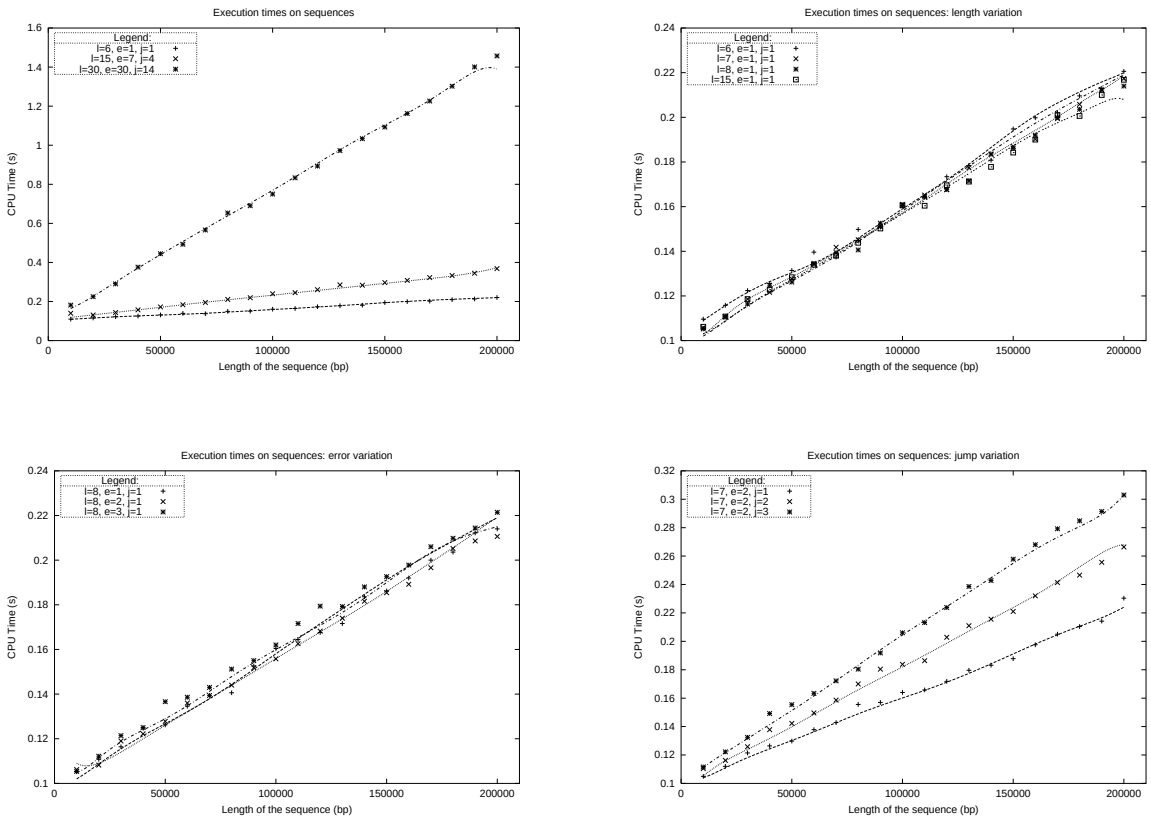

FIG. 11: Execution times on sequences, where l is the length of the copies, e is the maximal Hamming distance and j is the jump

presents the copies, the alterations and sums up informations relative to the repeats. We are now looking for this kind of repeats in complete genomes, we found several interesting e.t.r. that are not inherited from approximate tandem repeats. We are still in the process of studying the way it works, from the biologist viewpoint and we are trying to figure out their role, preferred location and number in different genomes. Since considering Hamming distance is somehow restrictive, we are moving forward by designing an algorithm that makes use of Levenshtein distance (which allows indels as well as substitution) instead of Hamming distance.

References

[1] G. Benson. An algorithm for finding tandem repeats of unspecified pattern size. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Proceedings of the 2nd Annual International Conference on Computational Molecular Biology (RECOMB-98)*, pages 20–29, New York, Mar.22–25 1998. ACM Press.

[2] G. Benson. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res.*, 27(2):573–580, 1999.

[3] O. Elemento, O. Gascuel, and M.-P. Lefranc. Reconstructing the duplication history of tandemly repeated genes. *Molecular Biology and Evolution*, (19):278–288, 2002.

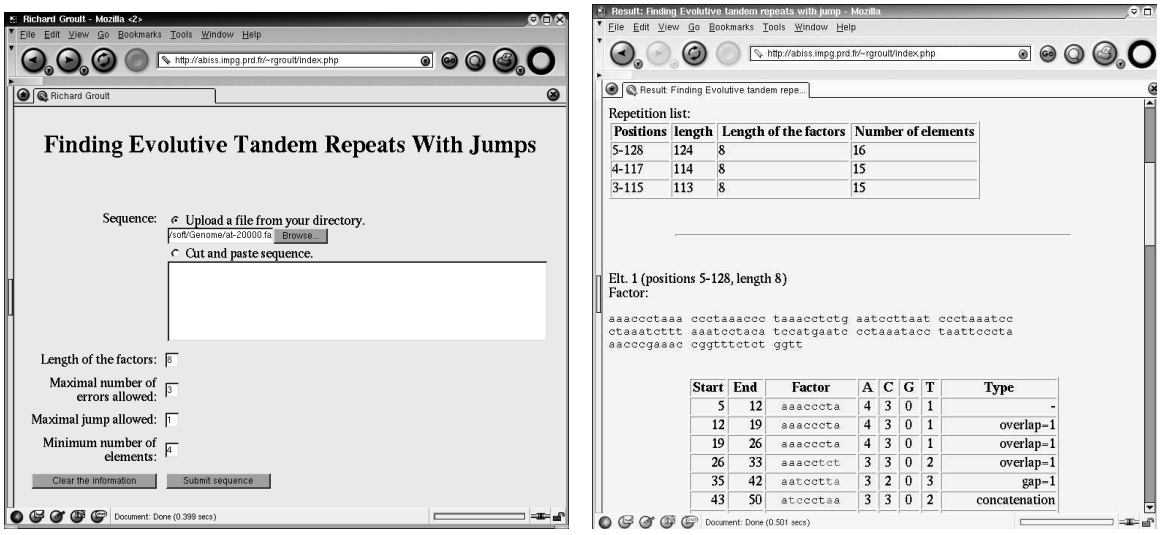

FIG. 12: HTML interface

[4] D. Golstein and C. Schlotterer. *Microsatellites: Evolution and Applications*. Oxford University Press, 1999.

[5] R. Groult, M. Léonard, and L. Mouchard. Evolutive tandem repeats using hamming distance. In *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science*, pages 292–304, Warszawa - Otwock, Poland, Aug. 2002. Lecture Notes in Computer Science 2420, K. Diks, W. Rytter (Eds.), Springer.

[6] A. Jeffreys. Higly variable minisatellites and DNA fingerprints. *Biochem. Soc. Trans.*, 15:309–317, 1987.

[7] R. M. Kolpakov and G. Kucherov. Finding maximal repetitions in a word in linear time. In *IEEE Symposium on Foundations of Computer Science*, pages 596–604, 1999.

[8] R. M. Kolpakov and G. Kucherov. Finding approximate repetitions under hamming distance. In *Proceedings of the 9th European Symposium on Algorithms (ESA 2001)*, volume 2161 of *Lecture Notes in Computer Science*, pages 170–181, Aarhus, Denmark, 2001.

[9] S. Kurtz, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. G. Computation and visualization of degenerate repeats in complete genome. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, pages 228–238, La Jolla, California, 2000. The AAAI Press.

[10] S. Kurtz and C. Schleiermacher. Reputer - fast computation of maximal repeats in complete genomes. *Bioinformatics*, 15(5), 1999.

[11] A. Lefebvre and T. Lecroq. Computing repeated factors with a factor oracle. In L. Brankovic and J. Ryan, editors, *Proceedings of the 11th Australasian Workshop On Combinatorial Algorithms*, pages 145–158, Hunter Valley, Australia, 2000.

[12] M. Sagot and E. W. Myers. Identifying satellites in nucleic acid sequences. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Proceedings of the 2nd Annual International Conference on Computational Molecular Biology (RECOMB-98)*, pages 234–242, New York, Mar.22–25 1998. ACM Press.

A An Example of e.t.r. Occurring in *A. thaliana*, chr 4 (17Mb)

We found numerous e.t.r. in chr 4 (17Mb) of *A. thaliana*, here is an example appearing in an exon of the `AT4G38590.1` gene.

```
./evorep  -m11 -e3 -j1 -r4 -f ~/at4.fasta
->
  - number of e.t.r.: 662
  -             time: 0m38.758s

Example of found e.t.r.
#================================================
# Parameters: length=11, error=3, jmin=-1, jmax=1, rMin=4
# Sequence: > at4.seq (17Mb)
# Execution time: 38 sec.

17245698 17245709 17245719 17245731 17245743 17245755 17245767
acaagatgagaagaagaagaaagaagataaagacgaagaggaagaggacgatgaagatgatgatgaagaagaag
                                                                        [ aagaag
17245698        acaagatgaga
17245709        agaagaagaaa
17245719        agaagataaag
17245731        cgaagaggaag
17245743        ggacgatgaag
17245755        tgatgatgaag
17245767        agaagaagaag
#================================================
```

We investigated this sequence using "tandem repeat finder" [2] and "mreps" [7] and obtained:

```
->
Tandem Repeat Finder:
Indices Period Copy   Consensus Percent Percent Score A C G T Entropy(0-2)
        Size   Number Size Matches Indels
No Repeats Found!

->
./mreps  -err 3 -minp 2 -from 1 -exp 3.0
* Processing window [1 : 80] *

from -> to : size <per.> [exp.]  repetition
-----------------------------------------------------
  1  -> 18 : 18   <5>    [3.60]  acaag atgag aagaa gaa
```

```
  5  -> 25 : 21   <6>    [3.50]  gatgag aagaag aagaaa gaa
  8  -> 40 : 33   <4>    [8.25]  gaga agaa gaag aaag aaga taaa gacg aaga g
 10  -> 32 : 23   <7>    [3.29]  gaagaag aagaaag aagataa ag
 11  -> 33 : 23   <5>    [4.60]  aagaa gaaga aagaa gataa aga
 20  -> 80 : 61   <6>    [10.17] aaagaa gataaa gacgaa gaggaa gaggac gatgaa
                                                [ gatgat gatgaa gaagaa gaagaa g
 30  -> 80 : 51   <9>    [5.67]  aagacgaag aggaagagg acgatgaag atgatgatg
                                                [ aagaagaag aagaag
 30  -> 80 : 51   <12>   [4.25]  aagacgaagagg aagaggacgatg aagatgatgatg
                                                [ aagaagaagaag aag
 36  -> 47 : 12   <4>    [3.00]  aaga ggaa gagg
 60  -> 80 : 21   <4>    [5.25]  atga tgaa gaag aaga agaa g
-----------------------------------------------------
RESULTS: There are 10 maximal repetitions in the segment processed
```