International Journal of Computer Applications (0975 - 8887)
Volume 70 - No. 4, May 2013

The Weighted Factors Automaton : A Tool for DNA Sequences Analysis

Christiane Hespel
IRISA-INSA
20 avenue des Buttes de Coesmes
35043 Rennes cedex, France

Farida Benmakrouha
IRISA-INSA
20 avenue des Buttes de Coesmes
35043 Rennes cedex, France

Danielle Quichaud
IRISA-INSA
20 avenue des Buttes de Coesmes
35043 Rennes cedex, France

ABSTRACT

A lot of computing tools are often used for analyzing DNA sequences like trees, automata, dictionaries, every one being reserved for a particular problem. A. Blumer and al. have proposed a more general computing tool : the smaller automaton recognizing the subwords of a text (DAWG).
In this paper we propose the concept of "weighted factors automaton" producing every occurrence of any factor. Its transitions are labeled by the read letter and weighted by the set of the indices of the factors beginnings. The factors are obtained by concatenating the read letters and the indices of the factors beginnings are obtained by computing the intersection of the weighting sets, when advancing from the initial state to a final state.
We think that this automaton can be more easily processed than DAWG and we present a comparison between DAWG and our automaton: the set of the factors beginnings indices and the factors frequency are more easily obtained by our automaton and the restriction of our automaton to the factors of length $\leq k$ maintains the automaton structure, when DAWG cannot be easily restricted.
The applications are numerous: By selecting factors of length 1, we obtain the coding regions, factors of length 3, we obtain the expression level of some gene. The "weighted factors automaton" allows us to find matches of pattern, to study homology, FASTA and BLAST algorithms being significantly simplified.

General Terms:

Theory, Algorithms

Keywords:

Algorithms on Strings, Weighted automata, Bioinformatics, DNA Sequences Analysis, DNA Sequences Biasis

1. INTRODUCTION

Several tools are often used for analyzing DNA sequences: suffix table, suffix tree and suffix automaton are efficient for the pattern matching, the dictionary of the k-uples is built by BLAST or FASTA algorithms for the study of homology, the dynamic programming is used for the search of the best exact alignment with gap of 2 sequences [4, 5, 10]. For large sequences like entire chromosomes, heuristic methods are used for analyzing them, particularly for detecting repetitions [13].
The factors automaton, directly derived from the suffix automaton, is described by Blumer [2] and Crochemore [3]. The factors transducer [3, 9] provides the factors with the index of their first occurrence when DAWG [2] provides the index of every occurence. In this paper, we propose an expansion of a small DNA sequence S by its "weighted factors automaton", built on the four letters of the alphabet $X = \{A, C, G, T\}$, providing the factors (subsequences) of the sequence, every factor being given with the set of the indices corresponding to its occurrences beginnings.
After some preliminaries about the theoretic aspects of the words, languages and automata, we present a construction of the "weighted factors automaton", some remarks relating to the size of this automaton and a comparison between DAWG and this automaton. We show that some operations over the automata like mirror and star allow to analyze the reverse subsequences and to study the repetitions in the original sequence. Then we show that some operations over the sequences like concatenation or truncation can induce corresponding operations over the "weighted factors automata".
The main interest of this tool is in the applications to bioinformatics: When analyzing some DNA sequence, it is better to use a full tool containing all the informations instead of several tools, allowing us to obtain the indices, the frequencies of the factors beginnings in order to know the coding regions, the expression level of genes, to study homology.

2. PRELIMINARIES : WORDS AND AUTOMATA

2.1 Alphabet, words, factors (subsequences)

An alphabet A being a finite nonempty set, its elements are called letters. A word over an alphabet A is a finite string built with the letters of A. The empty word denoted by ϵ is made of zero letter. The juxtaposition of two words is called concatenation. A language L being a set of words, $w \in L$, β is a factor (or a subsequence) of w if there are $\alpha, \gamma \in L$ such that $w = \alpha\beta\gamma$.

2.2 Automaton

See [1, 8, 11]. A finite automaton M over an alphabet A is a 4-uple made of

(1) A finite set Q of states
(2) An initial state q_0
(3) A set $T \subseteq Q$ of final states
(4) A set $F \subseteq Q \times A \times Q$ of transitions

A word is recognized by an automaton if there exists a path from the initial state to a final state, labeled by the letters of this word. Knowing that every finite language is recognized by some finite automaton, the next section consists firstly in constructing the finite automaton recognizing the language equal to the finite set of the subsequences (factors) of a given finite DNA sequence built on the alphabet $\{A, C, G, T\}$

3. “THE WEIGHTED FACTORS AUTOMATON”

First we present the automaton recognizing the set of nonempty factors of a given DNA sequence S. It is based on the idea that every nonempty subsequence of S is nonempty prefix of a suffix of S. Knowing the minimal suffixes automaton of S [2], we have just to allow every state different from the initial state, to be final state.

3.1 The suffixes automaton

M. Crochemore [4] present the suffixes automaton $Suf(S)$ as the minimal automaton recognizing the set of the suffixes of a sequence S. A lot of applications like exact online string matching are available [6].
For a sequence S of length n, they prove that the states number and the transitions number are linear in n and are $\leq Sup(2n-1, 3n-4)$. They prove that the time complexity ([7]) of the construction of $Suf(S)$ is $O(n)$.

3.2 The “weighted factors automaton”

The factors automaton recognizes the set of nonempty factors (or subsequences) if and only if it recognizes the nonempty prefixes of the suffixes. Then it is obtained, from the suffixes automaton, by allowing every state, except the initial one, to be final. This automaton has the same size as the previous one. And then, the unexpected property is that the size (sum of the states and the transition numbers) is linear in n when the number of subsequences is generally quadratic [3].
But our aim is to provide the subsequences with the indices of each occurrence appearing in the sequence S. So we propose that this automaton should be more informed: The transitions are labeled by the read letter and are weighted by the set of the indices of the recognized subsequences beginnings. By starting at the initial state, by reading the letter labeling the transition and by maintaining the same subsequence index when advancing up to a final state, every subsequence can be provided with its index.
Example 1 : For $S = ACACGT$, we obtain the automaton given by Figure 1.

3.2.1 Construction of the “weighted factors automaton”. First we build the whole sequence automaton, we enrich it by adding some transitions and states in order to recognize the sequence suffixes and we end by allowing every state, except the initial one, to be final.
Two informations are provided for every transition : the label, expressed by the read symbol (letter) and the weight, expressed by the set of the occurrences beginnings indices.
Example 1 :
For $S = ACACGT$, let us denote by i the index of the suffix beginning.

(1) $i = 1$: $ACACGT$ and its proper nonempty prefixes $\{A, AC, ACA, ACAC, ACACG\}$
(2) $i = 2$: $CACGT$ and its proper nonempty prefixes $\{C, CA, CAC, CACG\}$
(3) $i = 3$: $ACGT$ and its proper nonempty prefixes $\{A, AC, ACG\}$
(4) $i = 4$: CGT and its proper nonempty prefixes $\{C, CG\}$
(5) $i = 5$: GT and its proper nonempty prefixes $\{G\}$
(6) $i = 6$: T

The Algorithm is the following for $n = length(S)$

(1) For every transition, the indices set is equal to $\emptyset$ and we set $i = 0$
(2) For $i = 1$ to n we add i to the indices set, on every transition composing the path corresponding to the factor beginning at index i.

If we are processing $S = ACACGT$ for $i = 4$, we add 4, index of the beginning of C, CG, CGT, to the sets weighting the 3 transitions of the path CGT.

3.2.2 This automaton provides the subsequences with their indices set

(1) by starting at the initial state,
(2) by following a successful path (from the initial state to a final state) compounded of the letters labeling the transitions, while
   (a) concatenating the letters in order to build the corresponding subsequence and while
   (b) calculating the nonempty intersection of the sets weighting these transitions.

Example 1 : For $S = ACACGT$, AC is a subsequence of $ACACGT$ and the set of its occurrence beginnings indices is $\{1, 3\} = \{1, 3\} \cap \{1, 3\}$.
CG is a subsequence of $ACACGT$ and the set of its occurrence beginnings indices is $\{4\} = \{2, 4\} \cap \{3, 4\}$.

3.3 Comparison between DAWG and the “weighted factors automaton”

The two automata structures are the same but the informations are different: in DAWG, the states are denoted by the set of the letters indices composing the factors, when in the “weighted factors automaton”, the transitions are weighted by the set of the factors beginnings indices.
And then, for providing some factor of DAWG, we have to start at the initial state, to follow a path by incrementing the state index (corresponding to the current letter index) from this state to the next state, up to a final state. On the contrary, in the “weighted factor automaton”, we have just to follow a path by maintaining the same transition index.

Example 2 : $S = ACCCCCG$. We present the two automata in Figure 2, Figure 3.
For instance, the beginning index of factor $CCCG$ is obtained by computing the intersection of the intervals 2..6 and 2..5 and 2..4 and 4..4 = 4..4, weighting the corresponding path in the second automaton (Fig 3) when the computing is more hard in the first automaton (Fig 2) since we have to compute an indices incrementation (from the current letter to the next letter) and an intervals intersection.

3.4 Some operations over the “weighted factors automata” inducing some operations on the sequences

The operations over automata such as mirror and star are particularly interesting. The mirror automaton allows us to provide the palindromes of a DNA strand and to study the DNA strand in the opposite direction. The star automaton is useful for modeling the repetition of motifs constituting a DNA strand.
3.4.1 The mirror automaton. An automaton recognizing a language $L = \{w_i = z_{i_1} \cdots z_{i_l}\}_{i \in I}$ where $z_{i_k} \in A$, its mirror recognizes the reverse language $L' = \{{w_i}^R = z_{i_l} \cdots z_{i_1}\}_{i \in I}$.
The construction of M', mirror automaton of an automaton M consists in inverting the direction of the transitions and in exchanging initial state for final state. The original automaton recognizing the prefixes of the suffixes of the original sequence S, the mirror automaton recognizes the suffixes of the prefixes of the reverse sequence S^R.
Instead of calculating the indices of the subsequences beginnings, the construction of the mirror automaton incites us to calculate the indices j of the subsequences ends, either directly or by using the indices i of the subsequences beginnings of the original automaton in terms of n: $j = n + 1 - i$

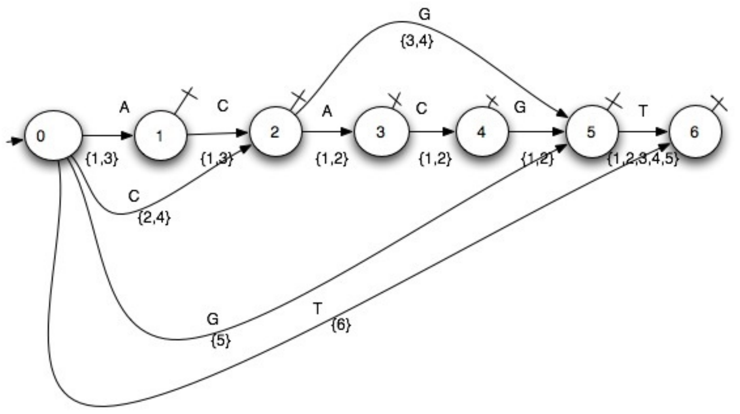

Fig. 1. "Weighted factors automaton" of ACACGT

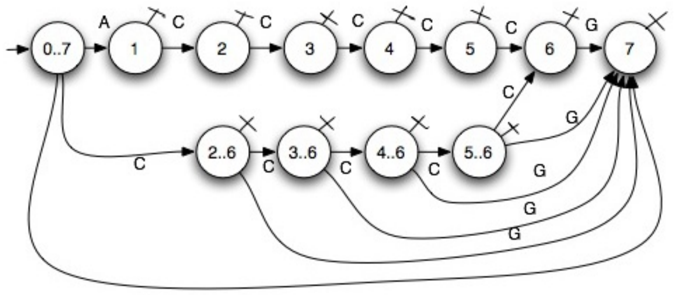

Fig. 2. DAWG of AC^5G

Example 1 : For $S = ACACGT$, let us denote by j the index of the prefix end of $S^R = TGCACA$.

(1) $j = 6$: $TGCACA$ and its proper nonempty suffixes $\{A, CA, ACA, CACA, GCACA\}$
(2) $j = 5$: $TGCAC$ and its proper nonempty suffixes $\{C, AC, CAC, GCAC\}$
(3) $j = 4$: $TGCA$ and its proper nonempty suffixes $\{A, CA, GCA\}$
(4) $j = 3 : TGC$ and its proper nonempty suffixes $\{C, GC\}$
(5) $j = 2 : TG$ and its proper nonempty suffix $\{G\}$
(6) $j = 1 : T$

By computing $j = 7 - i$, the subsequences are provided with their indices set according to the steps:

(1) by starting at an initial state,
(2) by following a successful path (from an initial state to a final state), while
    (a) concatenating the letters in order to build the corresponding subsequence
    (b) calculating the intersection of the sets (of the indices j) weighting these transitions.

GC is a subsequence whose weight is $\{3,4\} \cap \{3,5\} = \{3\}$ and CA is a subsequence whose weight is $\{4,6\} \cap \{4,6\} = \{4,6\}$. The palindromes (recognized by the automaton and its mirror) are:A, C, G, T, ACA, CAC. See Figure 4.

3.4.2 The star automaton. An automaton recognizing a language L, its star recognizes the star language $L^* = \Sigma_{k=0}^{\infty} L^k$
The construction of M^*, star automaton of an automaton M consists in adding a transition from the last final state (associated with the last letter of the whole sequence) to the state (associated with the first letter of the whole sequence) successor of the initial state and in allowing the initial state to be final state. See Figure 5.
For providing the non empty subsequences of a finite sequence, we bound the number of repetitions and prohibit the initial state from being final. So we use a weight of the form $kl + r$ where l is the length of the motif, $k \in N$ is bounded by the number of repetitions and r is the index in the first occurrence of the motif. For $S = (ACACGT)^5$, the weight indices have to be added to $6k$ for $0 \leq k \leq 4$.

3.4.3 The sub-automaton of k-subsequences. For a lot of problems, the sub-automaton recognizing the subsequences of length $\leq k$ is the right tool. It is obtained by selecting the successful paths of length $\leq k$. In **Algorithm *3.2.1***, we restrict the length of the paths to k. And then the automaton structure is maintained but some indices sets are restricted.
See Figure 6 for $ACACGT, k \leq 3$.
Remark 1 : Such a sub-automaton of DAWG cannot be provided, this restriction of the indices sets weighting some transitions in the sub-automaton being impossible with DAWG.
Remark 2 : The automaton recognizing subsequences of length $\leq k+1$ is obtained by growing the indices sets of the automaton recognizing subsequences of length $\leq k$.

3.5 Some operations over the "weighted factors automata" induced by some operations on sequences

Concatenation, truncation on sequences can be transfered to the corresponding "weighted factors automata".

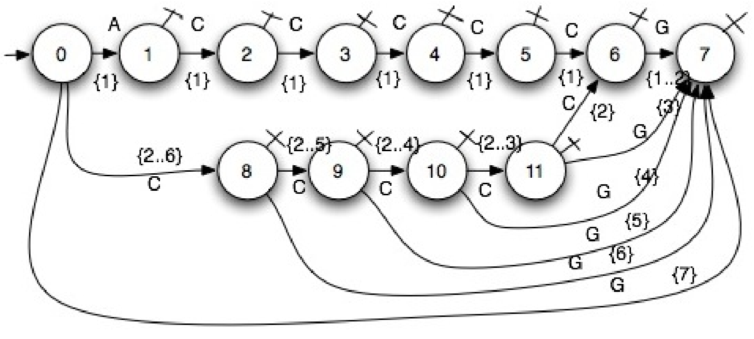

Fig. 3. Weighted factors automaton of AC^5G

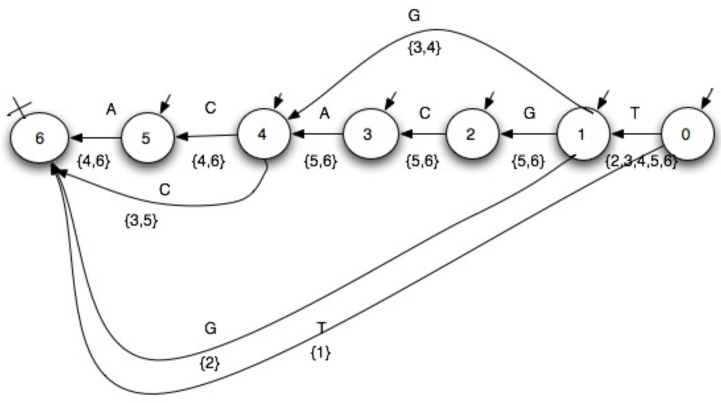

Fig. 4. Mirror weighted factors automaton of ACACGT

3.5.1 Concatenation of 2 sequences S_1 et S_2. Let be $Subseq(S)$ the "weighted factors automaton" of S. $Subseq(S_1.S_2)$ is obtained from $Subseq(S_1)$, $Subseq(S_2)$:

(1) In the "weighted factors automaton" $Subseq(S_1)$ of S_1, we add the sequence automaton of S_2 by identifying the initial state of S_2 to the final state of S_1 associated with the end of the reading of S_1.
(2) We add the transitions starting from the initial state of $Subseq(S_2)$ to the initial state of $Subseq(S_1)$.
(3) Transitions appearing in $Subseq(S_2)$ or overlapping $Subseq(S_1)$ and $Subseq(S_2)$ have to be studied.

3.5.2 Suppression of a subsequence S_1 of S. Let be S', the new sequence $S - S_1$, made of a left remainder S_l and a right remainder S_r. $Subseq(S')$ is obtained:

(1) In the "weighted factors automaton" $Subseq(S)$, we suppress the transitions labelled by the letters of S_1.
(2) We reconstruct the sequence automaton of S'.
(3) The transitions appearing between S_l and S_r in $Subseq(S')$ have to be studied.

Two particular cases occur when S_l is empty (suppression of the already studied beginning) or when S_r is empty (suppression of the non relevant end).

4. ADVANTAGES OF THE "WEIGHTED FACTORS AUTOMATON" FOR DNA SEQUENCES ANALYSIS

We present direct applications of this representation to some problems of bioinformatics, this automaton providing for every subsequence, its beginnings indices and then its frequency. By using this automaton for subsequences of length ≤ 3 , the search of the coding regions, of the reading frame, of the gene expression rate becomes elementary. The sequences homology is easier by using this automaton. Moreover, this automaton is an adaptive tool: It is possible to advance from the initial state to a final state, up to a suitable length for some object and to extend it to an upper length for an other object.
Remark 3 : In oder to simplify the explanations, we will assume that every subsequence is weighted by a pair made of the set $\{i_1, \cdots, i_{occ}\}$ of the subsequence indices in S and of its cardinal occ instead of the only set of the subsequences indices. For instance, for $S = ACA$, the recognized subsequences are, with their weight w:

$$w(A) = (\{1,3\},2) \qquad w(C) = (\{2\},1) \qquad w(AC) = (\{1\},1)$$

4.1 Search for the 3 DNA reading frame

By using the sub-automaton of 3-subsequences of the "weighted factors automaton", the reading frame number $k \in \{0,1,2\}$ is obtained by selecting in the set $\{i_1, \cdots, i_{occ}\}$ of the indices of the subsequence, the indices i_j such as $(i_j - 1) \ mod \ 3 = k$ (or such as $i_j = 3p + k + 1$ for $p \in N$) .
Let be S_1 = ATGAGTAAGCTGAAAGAGTACAGAGT GAACAGACAGATAAGGGCAAAGGAGTGCA
Its "weighted factors automaton" contains the subsequence AAG appearing with the weight $(\{7, 14, 39, 46\}, 4)$. The codon AAG, corresponding to the $0-$reading frame, appears twice at the indices $\{7, 46\}$.

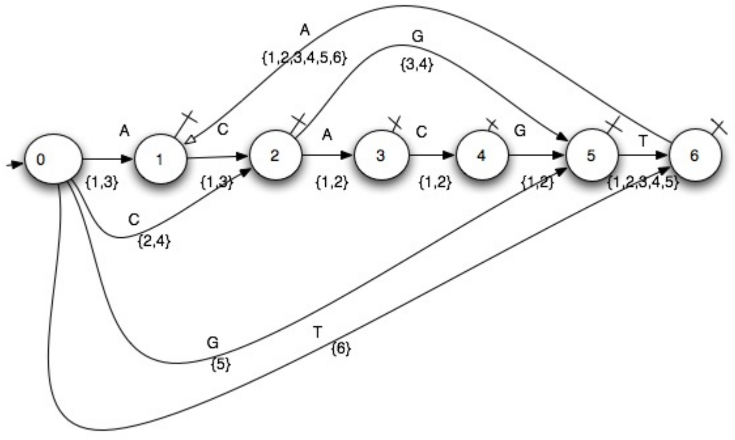

Fig. 5. Star weighted factors automaton of ACACGT

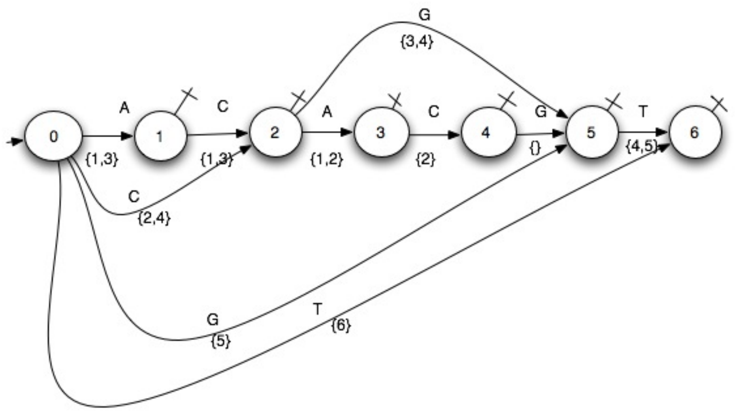

Fig. 6. Weighted factors sub-automaton of ACACGT ($k \leq 3$)

4.2 Period-3 component: detection of nucleotide bias and coding regions

The more important biases are found in the coding regions. They are detected by computing the following bias:

$$\Delta = \sum_{N=A,C,G,T} \sum_{k=0}^{2} |freq_N - freq_{N,frame\ k}|$$

where $freq_{N,frame\ k}$ indicates the frequency of the nucleotide N in the frame k and $freq_N$ indicates the frequency of the nucleotide N in the sequence. By computing this score Δ along a window of P codons, P going from 10 to 100, we detect the coding regions, corresponding to the more important biases.
By using the sub-automaton of 1-subsequences of the "weighted factors automaton", the computing of $freq_N$ is made by dividing the number of occurrences of N by the length of the sequence. The computing of $freq_{N,frame\ k}$ is made in the same way, by restricting it to $k-$reading frame.
For instance, for the sequence S_1, of length 55, the bias Δ is easily computed:
S_1 =ATGAGTAAGCTGAAAGAGTACAGAGTGAACA GACA-
GATAAGGGCAAAGGAGTGCA
Its "weighted factors automaton" contains the subsequence A of lenght 1 with the weight
({1,4,7,8,13,14,15,17,20,22,24,28,29,31,
33,35,37,39,40,45,46,47,50,55 }, 24)

So $freq_A = 24/55$ and the nucleotide A appearing at the $1-$reading frame at indices $\{8, 14, 17, 20, 29, 35, 47, 50\}$, 8 times, then $freq_{A,frame\ 1} = 8/18$. (There are $18 = 55/3$ nucleotides at $1-$reading frame).

4.3 Expression level of genes

We consider all its codons ($1 \leq i \leq L$), and we define for every codon i, the ratio $w_i = freq_{codon\ i}/freq_{major\ codon}$
The codon adaptation index (CAI) used to predict gene expression level is then $Index = (\Pi_{i=1}^{i=L} w_i)^{1/L}$
With our representation, the acquisition of every codon i is made by extracting the subsequences of length 3 in the $0-$reading frame, i.e. such as $(i-1) \quad mod \quad 3 = 0$. Their frequency is computed by dividing the number of occurrences of this codon at $0-$reading frame, by the whole number of codons at $0-$reading frame.
For instance, for the sequence S_2 of length 54:
S_2 = ATGAGTAAGCTGAAAGAGTACAGAGTGAACA-
GACAGATAAGGGCAAAGGAGGAG
the sub-automaton of 3-subsequences of the "weighted factors automaton", contains the subsequence AAG with the weight $(\{7, 14, 39, 46\}, 4)$. This codon appears twice (see 4.1), at $0-$reading, in the indices $\{7, 46\}$. Its appearance orders, as a codon, are $\{3 = (7 + 2)/3, 16 = (46 + 2)/3\}$. And then, it is the third codon of the sequence and its frequency is $freq_{codon\ 3} = 2/18$.
The major codon is the subsequence of length 3 at $0-$reading frame whose the number of occurrences is maximum. The

codon GAG appearing three times in the sequence at $0-$reading frame, it is major and

$$w_3 = freq_{codon\ 3}/freq_{major\ codon} = (2/18)/(3/18)$$

Then the codon adaptation index is easily computed.

4.4 Repetitions and palindromes

The search of repetitions of the same subsequence in a sequence is given by looking at the weight of this subsequence in the "weighted factors automaton" or by using the star automaton.
The search of the palindromes of a sequence is simplified by knowing the subsequences of this sequence and by using the mirror automaton.

4.5 Common longest subsequence

In order to simplify the search of the longest common subsequence to 2 sequences, we propose to add some informations to the "weighted factors automaton". We have to extract subsequences, from one of the automata, by decreasing length, knowing only their beginning index i. So we number every state of the whole sequence by the index of the last read letter and we add a link from every other state towards the corresponding numbered state. Then every state contains the index (or the link towards the state number) e of the subsequence end and the length l of the subsequence is $l = e - i + 1$. So, for the sequence $ACCCCCG$, the beginning index of the subsequence CCC belongs to the set $\{2, 3, 4\}$ and its length is given by the $1+$ the difference between the state number (4) linked with the final state (10) and the minimal beginning index 2 of the set. See Figure 7.

4.6 Homology of sequences, search of the longest common subsequence

Two sequences being given, the FASTA or BLAST methods [5] consist in extracting from one of these sequences, the dictionary of its $k-$tuples and the set of their indices, and in searching their occurrence in the other sequence.
By using our tool, the $k-$tuples dictionaries of the 2 sequences are already in their "weighted factors automata", when we restrict the automata to order k: FASTA or BLAST algorithms are significantly simplified.
The search of the longest common subsequence to 2 sequences is also significantly simplified since we have just to move along one of the "weighted factors automata" by decreasing length of the subsequence, looking for this subsequence in the other "weighted factors automaton".

5. DISADVANTAGES

Our method is based on the extraction and the utilization of the subsequences. Some algorithms do not lend themselves to this representation, particularly the algorithms based on the chain of the successive characters, like dynamic programming [4, 12]. Our method is not convenient to large sequences that need heuristic methods.

6. CONCLUSION

The advantages of the "weighted factors automaton" providing index and frequency of every subsequence, are numerous. It is a global computer tool: The same "weighted factors automaton" can be used by truncating the indices sets weighting the transitions, to the suitable order, according to the problem. For searching coding regions, the "weighted factors automaton" truncated at order 1 is suitable. For computing the gene expression level, the "weighted factors automaton" truncated at order 3 is suitable. In order to improve the homology search algorithms (BLAST, FASTA), the "weighted factors automaton" truncated at order k is suitable. And for searching for the longest common sequence, we can use the expanded "weighted factors automaton" with links.
Moreover, by using the "weighted factors automaton", the operations over the automata like mirror and star induce naturally some applications such as the search of palindromes, the studies of reverse strand and repetitions.
For these reasons, the "weighted factors automaton" is a right tool for the analysis of small DNA sequences.

7. REFERENCES

[1] J. Berstel, C. Reutenauer, *Rational series and their languages*. Springer-Verlag, 1988.

[2] A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M.T. Chen, J. Deiferas, *The smallest automaton recognizing the subwords of a text*. Theor. Comput. Sci. 40, 1985, 31–35.

[3] M. Crochemore, *Transducers and repetitions*. Theor. Comput. Sci. 45, 1986, 63–86.

[4] M. Crochemore, C. Hancart, T. Lecroq, *Algorithms on strings*. Cambridge University Press, 2007.

[5] F. Dardel, F. Képès, *Bioinformatique, Génomique et post-génomique*. Editions de l'Ecole Polytechnique, 2002.

[6] S. Faro, T. Lecroq, A Fast Suffix Automata Based Algorithm for Exact Online String Matching. LNCS 7276, Springer-Verlag, to appear, 149–158.

[7] P. Flajolet, R. Sedgewick, *Introduction à l'analyse des algorithmes*. International Thomson Publishing, 1998.

[8] M. Fliess, Un outil algébrique : les séries formelles non commutatives. In: G. Marchesini, S.K. Mitter (ed.), *Mathematical System Theory*. LNEMS 131, Springer-Verlag, 1976, 122–148.

[9] D. Gusfield, *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.

[10] D.L. Hartl, E.W. Jones, *Génétique, Les grands principes*. Dunod, 2003.

[11] C. Hespel, Une étude des séries formelles non commutatives pour l'Approximation et l'Identification des systèmes dynamiques. Thèse d'état, Univ. of Lille 1, 1998.

[12] C. N. Jones, A. P. Pevzner, *An introduction to Bioinformatics Algorithms*. MIT Press, 2004.

[13] A. Lefebvre, T. Lecroq, H. Dauchel, J. Alexandre, FORRepeats *Bioinformatics* (2003).

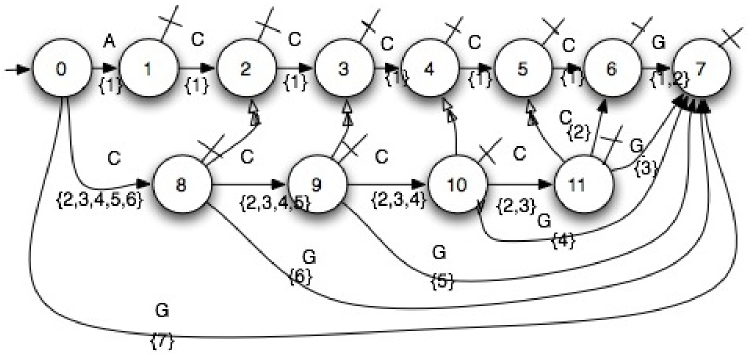

Fig. 7. "Weighted factors automaton" of ACCCCCG with links (double arrow)