J Glob Optim (2007) 39:623–645
DOI 10.1007/s10898-007-9158-9

Management and analysis of DNA microarray data by using weighted trees

Trang Tran · Cam Chi Nguyen · Ngoc Minh Hoang

Received: 5 July 2006 / Accepted: 23 March 2007 / Published online: 7 July 2007

Abstract We investigate discrete structures and combinatoric modeling of *weighted* prefix trees for managing and analyzing DNA microarray data. We describe the algorithms to construct the weighted trees for these data. Using these weighted trees with our algorithms, we propose methods to compute the *appearance probability* of a DNA microarray, to compare the informational distances in the expression of genes between the DNA microarrays, to search the *characteristic microarrays* and the group of *candidate genes* suggestive of a pathology.

Keywords Combinatorics on words · Weighted automata · Weighted trees · Classification · DNA microarrays.

1 Introduction

DNA microarray, or DNA chip, is a technology used to measure simultaneously the expression levels of thousands of genes under various conditions and then provide genome-wide insight. It is used to collect information from tissue and cell samples to observe differences in gene expression. The interpretation of the results requires new methods and software programs to capture the biological or medical information and to extract new knowledge from the microarray data.

There exist already several statistical methods and software programs to analyze the microarrays. Examples include (1) Unsupervised learning methods: hierarchical

T. Tran (✉) · C. C. Nguyen · N. M. Hoang
EA-2694 – Centre Intégré de BioInformatique – Université de Lille 2, 1 Place Verdun, Lille Cedex 59045, France
e-mail: ttran@univ-lille2.fr

C. C. Nguyen
e-mail: cnguyen@univ-lille2.fr

N. M. Hoang
e-mail: hoang@univ-lille2.fr

clustering and k-means clustering algorithms based on distance similarity metrics [7,11]; these are used to search for genes with maximum similarity. (2) Supervised learning methods: support vector machine approaches based on kernel function [9], machine learning classifiers and the Bayesian naive approach based on maximum likelihood method [3,16]; these are used to classify the gene expression into a training set. These methods are based essentially on *numerical* algorithms and do not really require structured data.

In this paper, we introduce *combinatoric* methods and *nonnumerical* algorithms, based on *weighted* prefix trees, to analyze microarray data, *i.e*:

– defining the informational distances to compare different microarrays,
– classifying the microarrays into the different categories,
– searching the characteristic microarrays,
– determining the group of candidate genes suggestive of a pathology.

The key to understanding our approach is that the information on gene expression in a microarray may be represented by a finite *symbolic* sequence, and a collection of microarrays may be viewed as a finite *multiset of words*, which can be implemented by using weighted prefix trees.

Prefix trees have been used to manipulate enormous masses of strings as a dictionary. These trees are used to distinguish all words of a multiset of words according to their prefixes, to search the frequency of patterns in a random text, to identify a sub-string in a word, etc... [10,14,15]. These prefix trees are viewed also as acyclic automata, and thus they can be represented by a matrix representation with coefficients in a Boolean ring [1,2,12].

To facilitate the probability and statistical computations, we enriched the data structure of prefix trees by introducing trees of *longest prefixes*, and by using *counting* trees labelled by an integer, and *probabilistic* trees labelled by *maximum likelihood* probability. This probability is, in fact, a frequency. Thus, to limit the propagation of numerical errors, the probability and statistical computations may be performed simply using only arithmetic operations on integers. When weighted trees are viewed as weighted automata without loops and this time, they are represented by a matrix representation with integer coefficients or with maximum likelihood probability coefficients. Using this representation of weighted automata, computations are done simply by using linear algebra. Further, following this representation, we could exploit other advantages of weighted automata as the reduction algorithm [2,4] to compress the data in our future work.

The weighted prefix trees provide a visual of a set of data and also are tools to classify the mass of data according to the weight of their prefixes. Thus, these structures open a new direction to examine the data-mining step in the process of knowledge discovery in databases. In other words, they permit the extraction of new information and hidden information from the mass of data. They also address the automatic learning and pattern recognition problems. More precisely, in this work we examine the following questions related to DNA microarray data.

1. Given some observed output strings $\mathcal{L}$ on an alphabet $\mathfrak{X}$, how do we construct the weighted prefix trees over $\mathcal{L}$?
2. How do we calculate the *appearance probability* of a string of $\mathcal{L}$?
3. How do we compare the information between different strings? This is most efficiently done by classifying the degree of overlap between different strings.

4. What are the sequences of nodes constituting the strings having the highest probability? This is most efficiently done by finding the sequence of nodes constituting the *characteristic strings* with the maximum frequency.
5. How do we find a transformation of $\mathcal{L}$ to obtain a multiset of words having *longest* prefixes and maximal probabilistic prefix? This is most efficiently done by finding the permutations of sequences giving the longest prefix.

This paper is organized as follows. The next section presents the elements of words, the concepts of multiset of words and permutations. Section 3 introduces the algorithm to construct a weighted tree and its matrix representation. Section 4 describes the application of weighted trees for the management and analysis of DNA microarray data.

2 Words and multiset of words

2.1 Words and precoding of words

Let $\mathfrak{X} = \{x_1, \ldots, x_m\}$ be an alphabet of the size m. A *word* w is a sequence of elements of $\mathfrak{X}$. The length of w is $|w|$. A particular case is the empty word that contains no letter, denoted by ε, $|\varepsilon| = 0$. The set of all words over $\mathfrak{X}$ is denoted by $\mathfrak{X}^*$. We can then express also (see [4,14])

$$\mathfrak{X}^+ = \mathfrak{X}^* \setminus \{\varepsilon\} \text{ and for any } h \geq 0 \;\; \mathfrak{X}^h = \{w \in \mathfrak{X}^*, \text{ s.t. } |w| = h\}. \tag{1}$$

Example 1 Let $\mathfrak{X}_1 = \{+, \circ, -\}$ be an alphabet. Words constructed over $\mathfrak{X}_1$ are $\{+ - \circ\circ +, + \circ + \circ\circ, + \circ + - - + \circ, ...\}$. Let $\mathfrak{X}_2 = \{A, C, G, T\}$, words constructed over $\mathfrak{X}_2$ are $\{ACGT, AATTACG, ATCTTTGACC, ...\}$.

The *concatenation* of word $u = x_{i_1} \ldots x_{i_k}$ and $v = x_{j_1} \ldots x_{j_l}$ is the word $w = uv = x_{i_1} \ldots x_{i_k} x_{j_1} \ldots x_{j_l}$. Equipped with the concatenation product, $\mathfrak{X}^*$ is a monoid whose neutral element is the empty word ε.

A word u (resp. v) is called *prefix*, or *left factor* (resp. *suffix*, or *right factor*) of w if $w = uv$. For any $u, v \in \mathfrak{X}^*$, let a be the *longest prefix* (*longest left factor*) of u and v, *i.e* $u = au'$ and $v = av'$. Let[1]

$$d(u, v) = |a| = d(v, u) \text{ and } d(u, v) = 0 \text{ if } a = \varepsilon. \tag{2}$$

Example 2 $d(+ - \circ\circ +, - \circ + \circ\circ) = |\varepsilon| = 0, d(+ \circ + \circ\circ, + \circ + - - + \circ) = |+ \circ +| = 3.$

For $x_i \in \mathfrak{X}$, let $\text{precod}(x_i) = i$ be the precoding of x_i, for $i = 1, .., m$. The *precoding* $\text{precod}(w)$ of w in base m (the cardinality of $\mathfrak{X}$) is defined as

$$\text{precod}(w) = \begin{cases} 0 & \text{if } w = \varepsilon, \\ m\,\text{precod}(u) + \text{precod}(x), & \text{if } w = ux, \forall u \in \mathfrak{X}^*, x \in \mathfrak{X}. \end{cases} \tag{3}$$

Example 3 Let $\mathfrak{X} = \{+, \circ, -\}$ be an alphabet and let

$$\text{precod}(+) = 1, \text{precod}(\circ) = 2, \text{precod}(-) = 3,$$

thus one has $\text{precod}(++) = 3\,\text{precod}(+) + \text{precod}(+) = 4$ and $\text{precod}(+ + -) = 3\,\text{precod}(++) + \text{precod}(-) = 15$.

[1] With this definition d is not a distance.

2.2 Languages

Let $\mathcal{L}$ be a finite multiset of words containing N words over $\mathfrak{X}^*$, for $N \ggg 0$. For $u \in \mathfrak{X}^*$, let us consider

$$N_u = \text{Card}\{w \in \mathcal{L} | \exists v \in \mathfrak{X}^*, w = uv\}, \quad \text{in particular } N_\varepsilon = N. \tag{4}$$

The N_u is the number of words beginning with u. Thus, $\forall u \in \mathfrak{X}^h$, one has

$$N_u = \sum_{x \in \mathfrak{X}} N_{ux} \text{ and for any } 0 \le h \le L,\ N = \sum_{u \in \mathfrak{X}^h} N_u. \tag{5}$$

The *mass function* μ over $\mathcal{L}$ is defined as follows

$$\begin{gathered} \mu : \mathcal{L} \longrightarrow \mathbb{N}, \\ u = x_{i_1} \dots x_{i_h} \longmapsto \mu(u) = \prod_{l=1}^{h} N_{x_{i_l}}. \end{gathered} \tag{6}$$

For $u \in \mathfrak{X}^*$, let us consider also the following ratios

$$P_u = \frac{N_u}{N}, \text{ in particular } P_\varepsilon = 1. \tag{7}$$

For $u \in \mathfrak{X}^*, x_i \in \mathfrak{X}$, to simplify the notation, let

$$p = \text{precod}(u), \quad q_i = \text{precod}(ux_i) = mp + \text{precod}(x_i). \tag{8}$$

and we consider the ratios

$$P_{p,q_i} = \frac{N_{ux_i}}{N_u}, \quad \text{for } i = 1, .., m. \tag{9}$$

By the formula (5), since $\sum_{x_i \in \mathfrak{X}} N_{ux_i} = N_u$ then the ratios P_{p,q_i} define the *discrete probability* over $\mathfrak{X}^*$:

$$0 \le P_{p,q_i} \le 1 \quad \text{and for all } u \in \mathfrak{X}^*,\ \sum_{x_i \in \mathfrak{X}} P_{p,q_i} = 1. \tag{10}$$

For $u = x_{i_1} \dots x_{i_h} \in \mathfrak{X}^h$, the *appearance probability* of u is computed by

$$P_u = \prod_{l=1}^{h} P_{q_{i_{l-1}}, q_{i_l}}. \tag{11}$$

Note that

$$\sum_{w \in \mathcal{L}} P_w = 1 \text{ and for any } 0 \le h \le L,\ \sum_{u \in \mathfrak{X}^h} P_u = 1. \tag{12}$$

Using the notation (9), for $u = x_{i_1} \dots x_{i_h}$, the *entropy* of u is computed by

$$H(u) = -\sum_{l=1}^{h} P_{q_{i_{l-1}}, q_{i_l}} \log_2 P_{q_{i_{l-1}}, q_{i_l}}. \tag{13}$$

Example 4 Let

$$\mathcal{L} = \{+-+, ++\circ\circ-, ++\circ--, ++\circ, +-+\circ, +-+\circ, ++\circ\circ, ++\}$$

be the multiset of 8 words over $\mathfrak{X} = \{+, \circ, -\}$. By using the notation (4,5), we have $N_+ = 8, N_{++} = 5$ and $N_{++\circ} = 4$. Thus by (6), $\mu(++) = N_+ \times N_{++} = 40$ and $\mu(++\circ) = N_+ \times N_{++} \times N_{++\circ} = 160$. By using the notation (9), we have $P_{0,1} = \frac{N_+}{N_\varepsilon} = 1, P_{1,4} = \frac{N_{++}}{N_+} = \frac{5}{8}, P_{1,6} = \frac{N_{+-}}{N_+} = \frac{3}{8}$ and $P_{4,14} = \frac{N_{++\circ}}{N_{++}} = \frac{4}{5}$, thus $P_{1,4} + P_{1,6} = 1$. By (11), $P_{++\circ} = P_{0,1}P_{1,4}P_{4,14} = \frac{1}{2}$. And by using the notation (13), $H(++\circ) = -(P_{0,1}\log_2 P_{0,1} + P_{1,4}\log_2 P_{1,4} + P_{4,14}\log_2 P_{4,14}) \approx 0.68$.

2.3 Rearrangement of multiset of words

Let $\mathcal{L} = \{w_1, \ldots, w_N\}$ be the finite multiset of words such that $|w_i| = L$, for $i = 1, .., N$. Let $\mathfrak{S}_L$ denote the set of permutations over $[1, \ldots, L]$. Let $\sigma \in \mathfrak{S}_L$ and let $w = x_{i_1} \ldots x_{i_L}$. Then

$$\sigma w = x_{\sigma(i_1)} \ldots x_{\sigma(i_L)}. \tag{14}$$

We extend this definition over $\mathcal{L}$ as follows

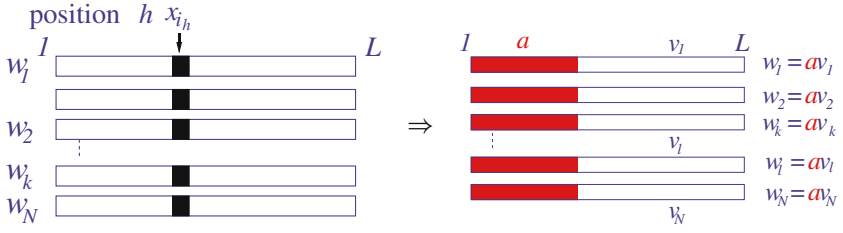
$$\sigma\mathcal{L} = \{\sigma w\}_{w\in\mathcal{L}}. \tag{15}$$

There always exists a permutation $\sigma \in \mathfrak{S}_L$ such that $\sigma w_1 = av_1, \ldots, \sigma w_N = av_N$, where $v_1, \ldots, v_N \in \mathfrak{X}^*$ and a is the *left longest factor* of $\mathcal{L}$. This permutation is not unique.

Example 5 Let $\mathcal{L} = \begin{Bmatrix} ++-\circ \\ \circ++\circ \\ -++\circ \end{Bmatrix}$ and $\sigma = \begin{pmatrix} 1&2&3&4 \\ 3&2&1&4 \end{pmatrix}$. Then $\sigma\mathcal{L} = \begin{Bmatrix} -++\circ \\ ++\circ\circ \\ +++-\circ \end{Bmatrix}$. Let $\sigma_1 = \begin{pmatrix} 1&2&3&4 \\ 4&2&3&1 \end{pmatrix}$ and $\sigma_2 = \begin{pmatrix} 1&2&3&4 \\ 2&4&1&3 \end{pmatrix}$. Then $\sigma_1\mathcal{L}$ and $\sigma_2\mathcal{L}$ have the same longest prefix $|\circ+| = |+\circ| = 2$.

We propose the REARRANGEMENT $(\mathcal{L})$ algorithm as follows: for $h \le L$ and for $x \in \mathfrak{X}$, let $n_h(x)$ be the number of letters x in the position h of the words in $\mathcal{L}$ and let n_h be the maximum of $n_h(x), x \in \mathfrak{X}$,

$$n_h = \max_{x\in\mathfrak{X}} n_h(x), \text{ and } \sum_{x\in\mathfrak{X}} n_h(x) = N. \tag{16}$$



One rearranges, then, the numbers $n_1, \ldots, n_L$ such that they appear in descending order $n_{\sigma(1)} \ge \cdots \ge n_{\sigma(L)}$ by *sorting* algorithm [15]. We return the permutation $\sigma = [\sigma(1), \ldots, \sigma(L)]$.

Example 6 With the multiset of words $\mathcal{L}$ in Example 5, one has

$$n_2(+) = n_4(\circ) = 3 > n_3(+) = 2 > n_1(+) = n_1(\circ) = n_1(-) = 1,$$

so one permutes the position 1 and the position 4 of multiset of words $\mathcal{L}$. One obtains $\sigma = \begin{pmatrix} 1\,2\,3\,4 \\ 4\,2\,3\,1 \end{pmatrix}$ and $\sigma\mathcal{L} = \begin{bmatrix} \circ + -+ \\ \circ + + \circ \\ \circ + +- \end{bmatrix}$. One has $a = \circ+$ as a longest common prefix of $\mathcal{L}$.

Algorithm 1 REARRANGEMENT ($\mathcal{L}$)

Input: a multiset of words $\mathcal{L}$;
for $h = 1$ to L **do**
  count $n_h(x), x \in \mathfrak{X}$;
  determine $n_h = \max_{x \in \mathfrak{X}} n_h(x)$;
  store $\{n_h\}_{1 \le h \le L}$;
end for
rearrangement $n_{\sigma(1)} \ge \cdots \ge n_{\sigma(L)}$;
return $\sigma = [\sigma(1), \ldots, \sigma(L)]$;
$\mathcal{L} \leftarrow \sigma\mathcal{L}$;
$a \leftarrow \varepsilon$;
for $h = 1$ to L **do**
  if exist $x \in \mathfrak{X}$ s.t $n_h(x) = N$ **then**
    $a \leftarrow ax$;
  end if
end for
Output: return $\sigma = [\sigma(1), \ldots, \sigma(L)]$ with longest prefix a;

The advantage of the permutation of a multiset of words is that it increases the length of common prefix between words but the appearance probability of words does not change, where

Theorem 1 *Let us suppose that there exists a permutation σ such that $\sigma(\mathcal{L})$ has the longest prefix. Thus,*

$$\forall w \in \mathcal{L}, \quad P_w = P_{\sigma w}.$$

Property 1 *Let σ be the permutation given in* Algorithm 1. *For any $w \in \mathcal{L}$, if there exists $u, \bar{u} \in \mathfrak{X}^h$ such that $w = uv$ and $\sigma w = \bar{u}\bar{v}$. Then*

1. $\mu(\bar{u}) \ge \mu(u)$,
2. $P_{\bar{u}} \ge P_u$,
3. $d(\sigma u, \sigma v) \ge d(u, v), \forall u, v \in \mathcal{L}$.

To evaluate the change between the multiset of words $\mathcal{L}$ and $\sigma\mathcal{L}$ according to their prefixes, we propose to use the following informational distance.

Proposition 1 *With the notations (6,9,13), the semi-distances between multiset of words $\mathcal{L}$ and $\sigma\mathcal{L}$ are defined on l^1 space as*

$$d_\mu(\sigma\mathcal{L}, \mathcal{L}) = \|\mu_{\sigma\mathcal{L}} - \mu_{\mathcal{L}}\|_1 = \sum_{\bar{u}, u \in \mathfrak{X}^h} |\mu(\bar{u}) - \mu(u)|,$$
$$d_P(\sigma\mathcal{L}, \mathcal{L}) = \|P_{\sigma\mathcal{L}} - P_{\mathcal{L}}\|_1 = \sum_{\bar{u}, u \in \mathfrak{X}^h} |P_{\bar{u}} - P_u|,$$
$$d_H(\sigma\mathcal{L}, \mathcal{L}) = \|H_{\sigma\mathcal{L}} - H_{\mathcal{L}}\|_1 = \sum_{\bar{u}, u \in \mathfrak{X}^h} |H(\bar{u}) - H(u)|.$$

3 Weighted prefix trees

This section introduces the weighted prefix trees which include counting trees and probabilistic trees. A counting tree is a data structure that classifies a set of words according to their successive symbols and their occurrence number. By using the maximum likelihood method, one obtains a probabilistic tree.

3.1 Prefix trees

Let $\mathcal{L} \subseteq \mathfrak{X}^*$ be a nonempty multiset of words and not containing the empty word ε. The *prefix tree* $\mathcal{A}(\mathcal{L})$ associated to the finite multiset of words $\mathcal{L}$ is usually used to optimize the storage of multiset of words $\mathcal{L}$ and defined as follows [10]

- the *root* is the initial node which contains the empty word ε,
- the set of the nodes corresponds to the prefixes of $\mathcal{L}$,
- the set of the terminal nodes represents $\mathcal{L}$,
- the transitions have the form (p, x, q) i.e: for $x \in \mathfrak{X}, u \in \mathfrak{X}^*$,

$$p = \text{precod}(u) \xrightarrow{x} q = \text{precod}(ux). \tag{17}$$

To enumerate the nodes of trees, we use the precoding of words defined in (8). The *internal* nodes $p = \text{precod}(u)$ associated to prefix u are nodes such that $N_u \geq 2$ and the *simple internal* nodes q are nodes such that $N_u = 1$. Thus an internal node p corresponds to a common prefix of all words stored in sub-tree of root p (see Fig. 1a). The prefix tree structure has been used to store enormous masses of words such as in a dictionary, to differentiate the words according to their prefixes, to search the frequency of patterns in a random text, to identify a sub-string in a text, etc.. [6,10,14,15,21]. There are several parameters which permit the analysis of the behavior of algorithms using this type of structure:

1. The *size* $S(\mathcal{A}(\mathcal{L}))$ is the sum of the number of internal nodes and the number of simple internal nodes. This parameter is the size of the necessary memory storage,

$$S(\mathcal{A}(\mathcal{L})) = \sum_{u \in \mathfrak{X}^*} \mathbb{1}_{\{N_u \geq 2\}} + \sum_{u \in \mathfrak{X}^*} \mathbb{1}_{\{N_u = 1\}}, \tag{18}$$

where $\mathbb{1}_{\{N_u \geq 2\}}$ and $\mathbb{1}_{\{N_u = 1\}}$ are the indication functions over set of nodes.

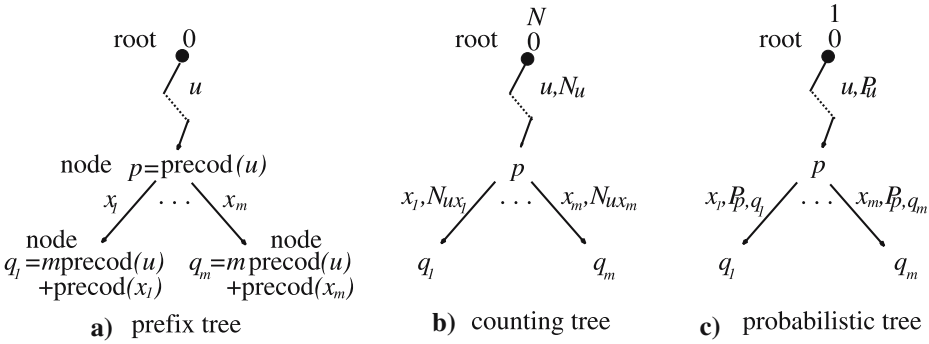


Fig. 1 Structure of the trees associated to a multiset of words $\mathcal{L}$

2. The *height* of the prefix tree, denoted $H(\mathcal{L})$, is the maximum longest prefix $d(u,v)$ among all longest prefixes of $\mathcal{L}$. This is the maximum number of necessary comparisons to separate two words,

$$H(\mathcal{L}) = \max_{u,v\in\mathcal{L}} d(u,v). \tag{19}$$

3. The *length of path* $LP(\mathcal{L})$ is the sum of all the longest prefixes $d(u,v)$. This parameter measures the number of necessary comparisons to construct the prefix trees and analyze the cost of the search algorithm,

$$LP(\mathcal{L}) = \sum_{u,v\in\mathcal{L}} d(u,v). \tag{20}$$

3.2 Counting trees

Let $\mathcal{L}$ be a finite multiset of words. Equipped with the number N_{ux} defined in (4,5), the prefix tree $\mathcal{A}(\mathcal{L})$ becomes a *counting tree* (see Fig. 1b). According to the notation (8) and the definition of the prefix tree, from an internal node $p = \text{precod}(u)$ associated with a sub-tree with p as a root which composes N_u words starting with the same prefix u. Then, the counting tree of $\mathcal{L}$ is constructed by INSERT $(\mathcal{L}, \mathcal{A})$ algorithm as follows: Denote Succ[p] the implementation of the set of the labeled successors of the node p. The construction algorithm of a counting tree is presented below; it is a modified version of that cited by Crochemore [10]. It considers successively each word of $\mathcal{L}$ in the loop of the lines 4 – 20, inserts it within the structure in sequential construction, letter by letter during the execution, and increases the number of labeled occurrences calculated on the lines 9 – 12. This algorithm permits one to manipulate any multiset of words to obtain a counting tree. By this construction, the transitions between the nodes on a counting tree have the form $(p,(x,N_{ux}),q)$, i.e:

$$p = \text{precod}(u) \xrightarrow{x,N_{ux}} q = \text{precod}(ux). \tag{21}$$

Proposition 2 *The construction time of a counting tree of N words is* $\mathcal{O}(N)$.

The advantage of the structure of prefix trees is that they distinguish all words of a multiset of words uniquely according to their prefixes. When we use counting trees, the weight of the common prefixes will be computed by (6). It permits the comparison of all words of $\mathcal{L}$ according to the weight of their prefixes. We propose the CHARACTERISTIC-WORDS $(\mathcal{A}(\mathcal{L}))$ algorithm, which returns the words having the maximum number of occurences, to extract *characteristic words*. By starting from the root of the tree and by descending sequentially, at each node p, the maximum product of N_{ux} will be used to determine the paths of nodes which gives the characteristic words. In other words, the choice of symbol x corresponding to the maximum weight $N_u N_{ux}$ determines the paths to follow,

$$\widehat{x} = \arg\max_x (N_u N_{ux}), \tag{22}$$

where $\arg\max_x N_x$ return a value x (not unique) which maximizes N_x.

Proposition 3 *The algorithm* CHARACTERISTIC-WORDS *has a complexity in* $\mathcal{O}(L)$ *with L is the height of counting tree.*

Algorithm 2 INSERT $(\mathcal{L}, \mathcal{A})$

Input: a list of words $\mathcal{L}$;
$\mathcal{A} \leftarrow$ new-tree();
$N_\varepsilon \leftarrow 0$;
while $\mathcal{L} \neq \emptyset$ **do**
  $w \leftarrow$ first($\mathcal{L}$);
  $\mathcal{L} \leftarrow$remainder($\mathcal{L}$);
  $p \leftarrow$ root;
  $N_\varepsilon \leftarrow N_\varepsilon + 1$;
  for each letter x of w sequentially **do**
    if exists a node q such that $(x, q) \in \text{Succ}[p]$ **then**
      $N_{ux} \leftarrow N_{ux} + 1$;
      $p \leftarrow q$;
    else
      $q \leftarrow$ new-node();
      $\text{Succ}[p] \leftarrow \text{Succ}[p] \cup \{(x, q)\}$;
      $N_{ux} \leftarrow 1$;
      $p \leftarrow q$;
    end if
  end for
end while
Output: a counting tree $\mathcal{A}(\mathcal{L})$;

Algorithm 3 CHARACTERISTIC-WORDS $(\mathcal{A}(\mathcal{L}))$

Input: $\mathcal{A}(\mathcal{L})$;
$p \leftarrow$ root; $u \leftarrow \varepsilon$;
while $p \neq NULL$ **do**
  for x_1 to x_m outgoing of the node p **do**
    $x \leftarrow \underset{x_k}{\text{argmax}}(N_u N_{ux_k}), 1 \le k \le m$;
    $u \leftarrow ux$;
  end for
  $p \leftarrow \text{precod}(u)$;
end while
Output: the words u with maximal occurrence numbers;

3.3 Probabilistic trees

To compute the appearance probability of an output word over an alphabet $\mathfrak{X}$, we introduce the *probabilistic tree*. A probabilistic tree is a modified counting tree. Augmented with the probability $P_{p,q}$ defined in (9), the counting tree $\mathcal{A}(\mathcal{L})$ becomes a probabilistic tree (see Fig. 1c). The labeled probability is estimated by the maximum likelihood method (see the notations (7,9)). It is the conditional probability that the word w accepts the common prefix ux knowing the common prefix u. The transitions between the nodes on a probabilistic tree have the form $(p, (x, P_{p,q}), q)$, i.e: for any $x \in \mathfrak{X}, u \in \mathfrak{X}^*$,

$$p = \text{precod}(u) \xrightarrow{x, P_{p,q}} q = \text{precod}(ux). \tag{23}$$

According to the construction of probabilistic trees, the appearance probability of a word is computed by (11).

Example 7 Let $\mathcal{L} = \{+-+\circ, ++\circ\circ, ++\circ\circ, ++\circ\circ, +-+\circ, +-+\circ, ++\circ\circ, ++-\circ\}$ be the multiset of words over $\{+, \circ, -\}$, we have the weighted trees as below:

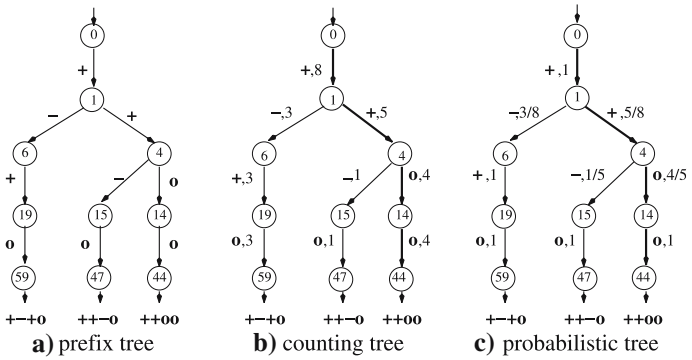


Fig. 2 Weighted trees associated to $\mathcal{L}$. The sequence of nodes $0 \to 1 \to 4 \to 14 \to 44$ represents the characteristic word

3.4 Matrix representation of weighted trees

Let $\mathcal{L} = \{w_1, \ldots, w_N\}$ be a finite multiset of words such that $|w_i| = L$, for any $i = 1, \ldots, N$. The prefix tree associated to $\mathcal{L}$ can be viewed as the finite acyclic automaton of size $n \leq m^L$. Thus, it can be represented by a matrix representation with coefficients in a Boolean ring [1,2,8,12]. Using the notations (4,5,9), we introduced the methods to construct weighted trees (counting trees and probabilistic trees) to enrich the structure of prefix trees. Weighted trees are viewed as finite weighted automata. In this manner, we get a weighted tree with a modified algebraic structure of a weighted automaton that can be represented as follows.

Definition 1 (Matrix representation of prefix trees)
Let K be the Boolean ring. The prefix tree $\mathcal{A}(\mathcal{L})$ associated with the finite multiset of words $\mathcal{L}$ over $\mathfrak{X}$ is given by the triplet (λ, M, γ), where

i. $\lambda = (1, 0, \ldots, 0) \in \mathcal{M}_{1,n}(K)$. The row vector λ represents the initial node (the root of the prefix tree),
ii. $\gamma = (\gamma_i)_{1 \leq i \leq n} \in \mathcal{M}_{n,1}(K)$ such that,

$$\gamma_i = \begin{cases} 1 & \text{if } \exists w \in \mathcal{L} \text{ such that } i = \text{precod}(w), \\ 0 & \text{otherwise.} \end{cases}$$

This column vector γ represents the set of the final nodes of the tree,
iii. M is a morphism of monoids representing of $\mathfrak{X}$ in the monoid of the square matrices of dimension n over K,

$$\begin{aligned} M : \mathfrak{X} &\longrightarrow \mathcal{M}_{n,n}(K) \\ x &\longmapsto M(x). \end{aligned}$$

For any $x \in \mathfrak{X}$, the coefficient of matrix $M(x)$ is given by

$$M_{i,j}(x) = \begin{cases} 1 & \text{if } \exists u \in \mathfrak{X}^* \text{ s.t. } i = \text{precod}(u) \text{ and } j = mi + \text{precod}(x), \\ 0 & \text{otherwise.} \end{cases}$$

The matrix $M(x)$ represents the transition of letter x on the prefix tree.

The triplet (λ, M, γ) is called the *K-matrix representation* of dimension n of prefix tree $\mathcal{A}(\mathcal{L})$.

Property 2 *The matrices $M(x), x \in \mathfrak{X}$, are triangular with diagonal elements of zero.*

Property 3 *For any $w = x_{i_1} \dots x_{x_L} \in \mathfrak{X}^*$, one has $M(w) = M(x_{i_1}) \dots M(x_{x_L})$.*

Property 4 *For any $u, v \in \mathfrak{X}^*$, one has $M(uv) = M(u)M(v)$.*

Property 5 *Let (λ, M, γ) be the matrix representation of the prefix tree $\mathcal{A}(\mathcal{L})$ associated to the finite multiset of words $\mathcal{L}$. Then*

$$\forall w \in \mathfrak{X}^*, \lambda M(w)\gamma = \begin{cases} 1 & \text{if and only if } w \in \mathcal{L}, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 2 (Matrix representation of counting trees)
Let $K = \mathbb{N}$. The counting tree $\mathcal{A}(\mathcal{L})$ associated with the finite multiset of words $\mathcal{L}$ is the given of the triplet (λ, C, γ), where λ and γ are already given in Definition 1 and by the notations (4,5), the morphism C is given by

$$C_{i,j}(x) = \begin{cases} N_{ux} & \text{if } \exists u \in \mathfrak{X}^* \text{ s.t. } i = \text{precod}(u) \text{ and } j = mi + \text{precod}(x), \\ 0 & \text{otherwise.} \end{cases}$$

The triplet (λ, C, γ) is called the $\mathbb{N}$-matrix representation of dimension n of counting tree $\mathcal{A}(\mathcal{L})$.

Example 8 The $\mathbb{N}$-matrix representation of the counting tree in Example 7 (Fig. 2b) is the following: $\lambda = (\mathbf{1}\,0\,0\,0\,0\,0\,0\,0\,0\,0), \gamma = (0\,0\,0\,0\,0\,0\,0\,\mathbf{1}\,\mathbf{1}\,\mathbf{1})^T$,

$$C(+) = \begin{pmatrix} 0 & \mathbf{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, C(\circ) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, C(-) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Definition 3 (Matrix representation of probabilistic trees)
Let $K = [0, 1]$. The probabilistic tree $\mathcal{A}(\mathcal{L})$ associated to the finite multiset of words $\mathcal{L}$ is the given of the triplet $(\lambda, \mathbb{P}, \gamma)$, where λ and γ are already given in Definition 1 and by notation (9), the morphism $\mathbb{P}$ is given by

$$\mathbb{P}_{i,j}(x) = \begin{cases} \dfrac{N_{ux}}{N_u} & \text{if } \exists u \in \mathfrak{X}^* \text{ s.t. } i = \text{precod}(u) \text{ and } j = mi + \text{precod}(x), \\ 0 & \text{otherwise.} \end{cases}$$

The triplet $(\lambda, \mathbb{P}, \gamma)$ is called the K-matrix representation of dimension n of probabilistic tree $\mathcal{A}(\mathcal{L})$.

With the definition of the matrices $\{C(x)\}_{x\in\mathfrak{X}}$, the coefficients of the matrices $\{\mathbb{P}(x)\}_{x\in\mathfrak{X}}$ can be performed as follows

$$\mathbb{P}_{i,j}(x) = \begin{cases} C_{i,j}(x)\Big(\sum_{j=1}^{m} C_{i,mi+j}(x_j)\Big)^{-1} & \text{if } C_{i,j}(x) \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

A word w is recognized by this tree if and only if it labels a path going from the initial node (root) to a final node. Therefore,

Theorem 2 *Let (λ, C, γ) be the matrix representation of the weighted tree. Then*

$$\forall w \in \mathfrak{X}^*, \lambda C(w)\gamma \neq 0 \ \text{ if and only if } \ w \in \mathcal{L}.$$

In particular $K = [0,1]$, the probability of appearance of word w is

$$P(w \in \mathcal{L}) = \lambda \mathbb{P}(w)\gamma.$$

Thus, the matrix representation (λ, M, γ) of a weighted tree does the computations in a simple way.

Example 9 From the matrices $\{C(x)\}_{x\in\{+,\circ,-\}}$ in Example 8, the matrices $\{\mathbb{P}(x)\}_{x\in\{+,\circ,-\}}$ are determined as follows

$$\mathbb{P}(+) = \begin{pmatrix} 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\mathbf{5}}{\mathbf{8}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \mathbb{P}(\circ) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\mathbf{4}}{\mathbf{5}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \mathbb{P}(-) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\mathbf{3}}{\mathbf{8}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\mathbf{1}}{\mathbf{5}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

By using Theorem 2, the appearance probability of the word $++\circ\circ$ is

$$P(++\circ\circ) = \lambda\mathbb{P}(++\circ\circ)\gamma = \lambda\mathbb{P}(+)\mathbb{P}(+)\mathbb{P}(\circ)\mathbb{P}(\circ)\gamma = \frac{1}{2}.$$

3.5 Trees having longest prefix

Consider the following schema

$$\mathcal{A}(\mathcal{L}) \longleftrightarrow \mathcal{L} \quad \underset{\sigma}{\rightarrow} \quad \sigma\mathcal{L} \longleftrightarrow \mathcal{A}(\sigma\mathcal{L}), \tag{24}$$

where $\mathcal{A}(\mathcal{L})$ (resp. $\mathcal{A}(\sigma\mathcal{L})$) is the tree associated with $\mathcal{L}$ (resp. $\sigma\mathcal{L}$). Where the tree $\mathcal{A}(\sigma\mathcal{L})$ represents the longest prefix of $\sigma\mathcal{L}$ (see Sect. 2.3). We call $\mathcal{A}(\sigma\mathcal{L})$ a Weighted Longest Prefix Tree (WLPT).

Example 10 Let $\mathcal{L}$ given in Example 7 and let $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix}$. The tree $\mathcal{A}(\mathcal{L})$ (resp. $\mathcal{A}(\sigma\mathcal{L})$) associated to $\mathcal{L}$ (resp. $\sigma\mathcal{L}$) is given in Fig. 3.

We propose the LONGEST-PREFIX-TREE $(\mathcal{L} \cup w)$ algorithm, as below. This permits the insertion of a word into a longest prefix tree, to give a new longest prefix tree. Suppose that at instant t one has the tree having the longest prefix $u \in \mathfrak{X}^*$ of $\mathcal{L}$

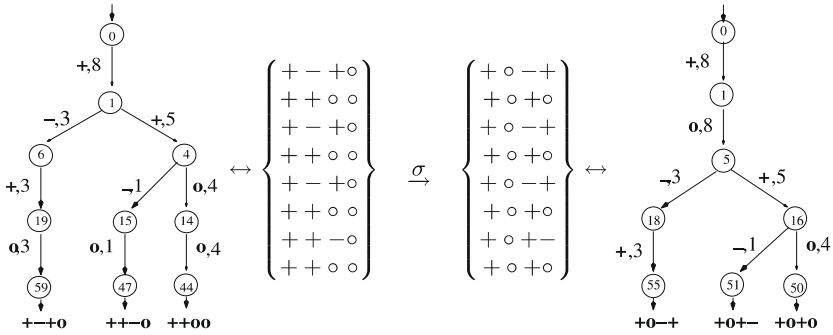


Fig. 3 Schema obtaining a WLPT $\mathcal{A}(\sigma\mathcal{L})$

with the permutation $\sigma = [\sigma(1), \ldots, \sigma(L)]$. To insert a word $w \in \mathfrak{X}^*$ in this tree, it is first necessary to permute this word by σ, to obtain $w' = \sigma w$. Hence, by notation (2), we calculate $l = d(w', u)$. In the case $l \neq 0$, one determines the longest prefix $u' \in \mathfrak{X}^*$ between u and w' such that $|u'| = l$ (note that $|u'| \leq |u|$). The search for a new permutation is as follows: by regarding l first components of σ, let us put

$$\sigma_1 = [\sigma(1), \ldots, \sigma(l)]. \tag{25}$$

By leaving from the depth $h = l + 1$ to depth L of the counting tree, let

$$n_h = \max_{x \in \mathfrak{X}} \sum_{u \in \mathfrak{X}^{h-1}} N_{ux}. \tag{26}$$

If there exists $x \in \mathfrak{X}$ such that $n_h = N_\varepsilon$, then one returns the new longest prefix $u'x$ by concatenating product u' with x. Hence, we then rearrange the numbers $n_{l+1}, \ldots, n_L$ such that they appear in descending order $n_{\sigma'(l+1)} \geq \cdots \geq n_{\sigma'(L)}$ using the sorting algorithm [15]. Let us put

$$\sigma_2 = [\sigma'(l+1), \ldots, \sigma'(L)]. \tag{27}$$

We return the new permutation by the concatenation of σ_1 and σ_2, *i.e*

$$\sigma = [\sigma_1, \sigma_2] = [\sigma(1), \ldots, \sigma(l), \sigma'(l+1), \ldots, \sigma'(L)]. \tag{28}$$

In the case $l = 0$, thus $u' = \varepsilon$. One implements (26) with $h = 1, \ldots, L$. And if there exists $x \in \mathfrak{X}$ such that $n_h = N_\varepsilon$, then one returns the new longest prefix $u'x$ by concatenating the product u' with x and a new permutation $\sigma = \sigma_2$.

Example 11 Let $\mathcal{L} = \{++-\circ, \circ++\circ, \circ++\circ\}$ be the multiset (Fig. 4).

Theorem 3 *Let $\mathcal{L}$ be the finite multiset of words over $\mathfrak{X}$. Let $\mathcal{A}(\mathcal{L})$ (resp. $\mathcal{A}(\sigma\mathcal{L})$) the counting tree associated to $\mathcal{L}$ (resp. $\sigma\mathcal{L}$). Then*

$$LP(\sigma\mathcal{L}) \geq LP(\mathcal{L}) \text{ and } S(\mathcal{A}(\sigma\mathcal{L})) \leq S(\mathcal{A}(\mathcal{L})).$$

Proof By the permutation $\sigma, \sigma\mathcal{L}$ having the longest prefix. That means, for any $u, v \in \mathcal{L}$, by Proposition 1, one has $d(\sigma u, \sigma v) \geq d(u, v)$. Thus

$$LP(\sigma\mathcal{L}) = \sum_{\sigma u, \sigma v \in \sigma\mathcal{L}} d(\sigma u, \sigma v) \geq LP(\mathcal{L}) = \sum_{u,v \in \mathcal{L}} d(u, v).$$

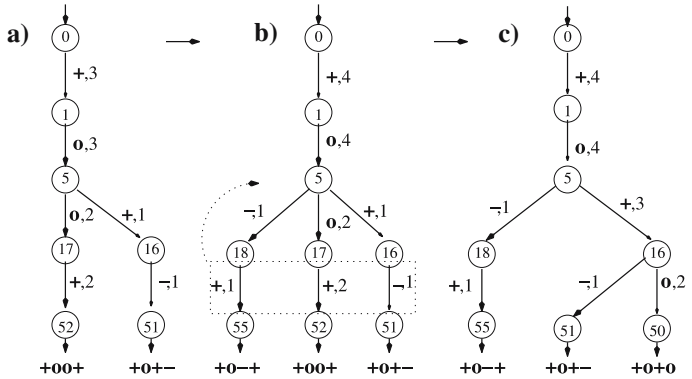


Fig. 4 The longest prefix trees. Tree (**a**) is longest prefix tree constructed on $\mathcal{L}$ by $\sigma = [2\ 4\ 1\ 3]$: the longest prefix is $u = +\circ$. After inserting the last word $\sigma(-++\circ)$, one obtained tree (**b**) with longest prefix is $u' = +\circ$ and observed that $n_3(+) = 3 > n_1(\circ) = 2$, then tree (**c**) is constructed from the tree (**c**) by $\sigma = [2\ 4\ 3\ 1]$

Algorithm 4 LONGEST-PREFIX-TREE ($\mathcal{L} \cup w$)

Input: tree $\mathcal{A}(\mathcal{L})$, longest prefix u, permutation σ, a word w;
Insert$(\sigma w, \mathcal{A})$;
$l \leftarrow d(\sigma w, u)$;
if $l \neq 0$ **then**
  determine left factor u' between σw and u such that $|u'| = l$;
  determine σ_1 according to (25);
  for depth $h = l+1$ to depth L of tree **do**
    determine $n_h = \max_{x \in \mathfrak{X}} \sum_{u \in \mathfrak{X}^{h-1}} N_{ux}$;
    if there exists $x \in \mathfrak{X}$ such that $n_h = N_\varepsilon$ **then**
      $u' \leftarrow u'x$;
    end if
  end for
  rearrangement $n_{\sigma'(l+1)} \geq \cdots \geq n_{\sigma'(L)}$;
  return σ_2 according to (27);
  $\sigma \leftarrow [\sigma_1, \sigma_2]$ according to (28);
else
  $u' \leftarrow \varepsilon$;
  realize the loop 7-14 with $l = 0$;
  $\sigma \leftarrow \sigma_2$;
end if
return u';
Output: tree $\mathcal{A}(\mathcal{L} \cup w)$, new longest prefix u', new σ;

Suppose that at the depth h of $\mathcal{A}(\mathcal{L})$ exists $x \in \mathfrak{X}$ verifying (26). Hence, the tree $\mathcal{A}(\sigma\mathcal{L})$ obtained is rebuilt from $\mathcal{A}(\mathcal{L})$ by regrouping the sum $\sum_{ux \in \mathfrak{X}^h} \mathbb{1}_{\{N_{ux} \geq 1\}}$ of nodes by only one node. And one deduces then

$$\sum_{\sigma u \in \mathfrak{X}^*} \mathbb{1}_{\{N_{\sigma u} \geq 2\}} + \sum_{\sigma u \in \mathfrak{X}^*} \mathbb{1}_{\{N_{\sigma u} = 1\}} \leq \sum_{u \in \mathfrak{X}^*} \mathbb{1}_{\{N_u \geq 2\}} + \sum_{u \in \mathfrak{X}^*} \mathbb{1}_{\{N_u = 1\}}.$$

From the formula (18), one has $S(\mathcal{A}(\sigma\mathcal{L})) \leq S(\mathcal{A}(\mathcal{L}))$.

4 Application: analysis of the DNA microarray data

In Sect. 3 we proposed the methods and algorithms of weighted prefix trees. In this section, we apply these methods to manage and to analyze DNA microarray data.

4.1 Brief representation of DNA microarray technology

DNA microarray is a multidisciplinary technology used to measure simultaneously the expression levels of thousands of genes under various conditions and provide genome-wide insight. A DNA microarray (slide) is a microscope slide to which the thousands of DNA fragments are attached. The DNA microarrays are hybridized with fluorescently labelled cDNA prepared from total mRNA of studied cells. The cDNA of the first cell sample is labelled with a green-fluorescent dye and the second with a red-fluorescent dye [7, 13]. After hybridization, the DNA microarrays are placed in a scanner to create a digital image of the arrays. The intensity of fluorescent light varies with the strength of the hybridization. The measure of expression level of a gene is determined by the logarithm of the ratio of the luminous intensity of the red fluorescence I_R to the luminous intensity of the green fluorescence I_G, $E = \log_2(\frac{I_R}{I_G})$. In this study, a change of differential expression of a gene between two cell samples by a factor of greater than $\sqrt{2}$ was considered significant. Thus:

- If $E \geq 0.5$, the gene is considered *up-regulated* in the second cell sample.
- If $-0.5 < E < 0.5$, the gene is considered *no-regulated*.
- If $E \leq -0.5$, the gene is considered *down-regulated*.

Since some microarray experiments can contain up to 30,000 target spots, the data generated from a single array mounts up quickly. For extraction of biological and medical informations from these data, the use of mathematics and software programs is essential to aid the interpretation of the results. The interpretation requires *discrete* and *combinatoric modeling*. This work used counting trees and probabilistic trees as the tools to solve the following problems:

P1. *storing* and *visualizing* the DNA microarray data,
P2. computing the appearance probability of a slide,
P3. comparing the gene expression level of different slides in order to classify the slides,
P4. identifying the *characteristic slides*: slides having the highest appearance probability,
P5. searching the *lexicographic order* of genes spotted on slides to obtain the common profile of DNA microarray data.

4.2 Solving the problems P1–P5

Consider a collection of L genes across in N different measure experiments, the gene expression profiles is the matrix $E = \left(E_{ij}\right)_{1\leq j\leq N}^{1\leq i\leq L}$, where $E_{ij} = \log_2(I_{R_{ij}}/I_{V_{ij}})$ and $I_{R_{ij}}$ (resp. $I_{G_{ij}}$) is the luminous intensity of the red (resp. green) fluorescence dye of spot (gene) i in experiment j. The key idea is modeling the DNA microarray (*slide profile*) as a sequence of symbolic information. In fact, the expression levels of a gene are modeled as a finite alphabet whose size is the number of expression levels, in this instance, three. Thus, a DNA microarray is viewed as a finite word over this alphabet, and a collection of microarrays forms a multiset of words is called a *DNA microarray*

language. In this study, we chose three symbols $+, \circ$ and $-$ to represent the three expression levels of a gene according to

- if $E_{ij} \geq 0.5$, the spot represents an *up-regulated* gene and symbolically encoded by the symbol $+$,
- if $-0.5 < E_{ij} < 0.5$, the spot represents a *no-regulated* gene and encoded by the symbol $\circ$,
- if $E_{ij} \leq -0.5$, the spot represents a *down-regulated* gene and encoded by the symbol $-$.

From this modeling, we obtain an alphabet $\mathfrak{X} = \{+, \circ, -\}$ that describes three expression levels of a gene. A slide profile is then represented by a *ternary* word of length L over $\mathfrak{X}$. And a collection of N slide profiles is represented as the multiset, noted $\mathcal{L}$, of N ternary words of length L (Fig. 5).

In general, the encoding by symbol sequences permits to optimize the storage (problem P1) by the use of the structure of prefix trees. The structure of WLPT (ternary tree) is a essential tool for analyzing a collection of slide profiles (problem P2–P5) by exploiting the weights (i.e. occurrence numbers) on the branches of trees. In fact, by *Algorithm 2* and *Algorithm 4*, we construct a WLPT associated to collection of slide profiles by classifying all words of the multiset of words $\mathcal{L}$ in a structure of weighted ternary tree according to their successive letters and their frequency labeled on the adjacent transitions (see Example 7 and 12). With this structure, we classify the informational expression between the slides by grouping the slides which have a common expression profiles as follows: let $x \in \mathfrak{X}$ be the labeled letter representing the expression of a gene at the node $q = \text{precod}(ux)$, where u is a prefix starting from the root of tree. Let N_{ux} be the number of occurrences observed for the letter x that labelled on the transition from the node $p = \text{precod}(u)$ to the node q. That means, for each node and each depth of the weighted tree, we know that there are N_{ux} slides having the common expression profile described by the prefix ux. The prefix ux is also called a *motif* of N_{ux} slides. For understanding our approach and how the theoretical tools in the preceding sections can really apply to extract biological information, we give the following example.

Example 12 Let $\mathcal{L} = \left\{ \begin{array}{c|cccccccc} & g_1 & g_2 & g_3 & g_4 & g_5 & g_6 & g_7 & g_8 \\ s_1 & + & + & + & + & + & + & + & + \\ s_2 & + & + & + & - & - & + & + & - \\ s_3 & - & \circ & - & \circ & \circ & \circ & - & \circ \\ s_4 & - & - & - & + & + & - & - & + \\ s_5 & + & + & + & + & + & + & + & + \end{array} \right\}^T$, be a symbolic matrix representing the expression profile of 8 genes $(g_1, \ldots, g_8)$ in 5 conditions $(s_1, \ldots, s_5)$. With the action of permutation $\sigma = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 3 & 7 & 2 & 6 & 4 & 5 & 8 \end{smallmatrix}\right)$ to $\mathcal{L}$, the gene expression data is structured by the WLPT given in Fig. 6.

$$L \text{ genes} \begin{pmatrix} E_{1,1} & E_{1,2} & \ldots & E_{1,N} \\ E_{2,1} & E_{2,2} & \ldots & E_{2,N} \\ \vdots & \vdots & & \vdots \\ E_{L,1} & E_{L,2} & \ldots & E_{L,N} \end{pmatrix} \overset{N \text{ slides (conditions)}}{\xrightarrow{\text{coding}}} \mathcal{L} = \begin{Bmatrix} + & + & - & \ldots & - \\ + & + & \circ & \ldots & - \\ \vdots & & \vdots & & \vdots \\ + & \circ & - & \ldots & \circ \end{Bmatrix}$$

Fig. 5 Symbolic representation of DNA microarray data

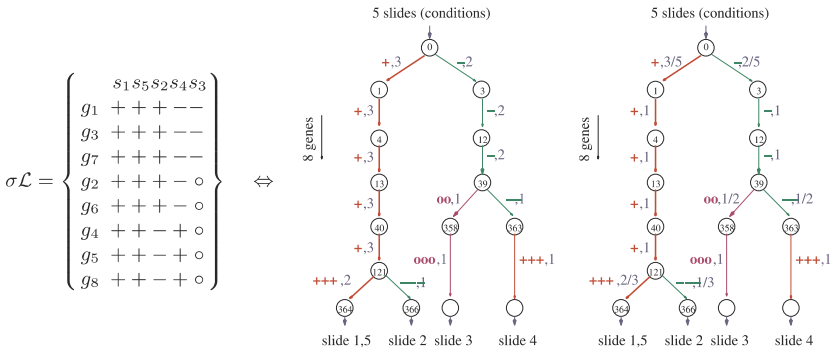


Fig. 6 WLPT of DNA microarray data

The left figure is the counting tree of 5 slide profiles. It represents the multiset $\mathcal{L}$ of 5 slide profiles. At the depth 5, the node 121 labels the word $u = +++++$ having the occurrence number $N_u = 3$ which indicates that a group of 3 slides (s_1, s_2, s_5) has the commun profile on 5 *up-regulated* genes; at the depth 3, the node 39 labels the word $v = ---$ having the occurrence number $N_v = 2$ which represents a group of 2 slides (s_3, s_4) with the common profile is 3 *down-regulated* genes. The right figure is the associated probabilistic tree. The slides 1 and 5 have the same appearance probability of 2/5 that are considered as the slides having the highest appearance probability. And then, these trees divide the data in various groups. For example, at the depth 5, the WLPT gives three groups of slides.

Using the WLPT, the four analytic problems (P2–P5) proposed are solved as below:

1. For the problem P2, we calculate directly the appearance probability of a word in two ways. The first way is to apply Theorem 2, the second is the use of (9):

$$\forall u = x_{i_1} \dots x_{i_h} \in \mathfrak{X}^h, P_u = \prod_{l=1}^{h} P_{q_{i_{l-1}}, q_{i_l}} = \frac{N_u}{N}. \tag{29}$$

   By using a logarithmic scale for (29), we obtain

$$\log P_u = \sum_{l=1}^{h} \log P_{q_{i_{l-1}}, q_{i_l}} = \log N_u - \log N. \tag{30}$$

2. For the problem P3, we measure the contained information in the words according to their prefixes as follows.

   Proposition 4 *By the notations (6,9,13), the informational semi-distance between different words according to the mass μ is*

$$\forall u, v \in \mathfrak{X}^h, \quad d_\mu(u, v) = |\mu(u) - \mu(v)|. \tag{31}$$

   The semi-distance between different words according to probability is

$$\forall u, v \in \mathfrak{X}^h, \quad d_P(u, v) = |P_u - P_v|. \tag{32}$$

   And the informational entropy semi-distance according to classic method,

$$\forall u, v \in \mathfrak{X}^h, \quad d_H(u, v) = |H(u) - H(v)|. \tag{33}$$

Table 1 Parameters of weighted trees

Tree\parameters	S	LP	$\overline{LP}$	H
$\mathcal{A}(\sigma\mathcal{L})$	93383	3004	3004/50	243

Proposition 5 *Let $u, v \in \mathfrak{X}^*$. The prefix distance between u and v, note $d_{pref}(u, v)$, is defined by*

$$\begin{aligned} d_{pref} : \mathfrak{X}^* \times \mathfrak{X}^* &\longrightarrow \mathbb{N} \\ (u, v) &\longmapsto d_{pref}(u, v) := |u| + |v| - 2d(u, v), \end{aligned}$$

where $d(u, v)$ is the length of longest prefix of u and v given by (2). For all $u, v \in \mathfrak{X}^*$, with this definition, the prefix distance $d_{pref}(u, v)$ is a metric.

Example 13 Let $w_1 = ++\circ--+, w_2 = ++\circ++-, w_2 = +-\circ-+-$ be three words representing three slides. We have $d_{pref}(w_1, w_2) = 6$ and $d_{pref}(w_1, w_3) = d_{pref}(w_2, w_3) = 10$. Thus w_1 and w_2 are more similar than w_1 and w_3. Then, two words w_1 and w_2 may be classified in the same group.

3. For the problem P4, we search the expression sequence w (slides w) which have the maximum appearance probability $P(w)$ by *Algorithm 3*; we thus obtain *characteristic slides*.
4. Problem P5 is solved by the *Algorithm 4* of Sect. 3.5. This permutation method permits our searching the order of genes spoted on a slide to establish the characteristic slides which have the longest common expressions.

4.3 Experimental results

We applied the weighted tree for the *Apoptosis* microarray collection provided by the Functional Genomic Platform laboratory of Lille 2. In this experiment, a collection of 50 Apoptosis microarrays of *colon cancer cells* issuing from three series of patients is studied. Each Apoptosis microarray contains 1920 spots representing 1920 genes. As noted above, a ternary word obtained by our modeling has the length $L = 1920$. The interpretation of these experimental results is illustrated in Figs. 7–9.

4.4 Some discussions

This paper presents the theoretical concepts from discrete mathematics based on algorithmics and combinatorics aspect to the extremely important problem for contemporary applied research—the transcriptomic data analysis. The objective of this original contribution is to develop the informatic tools including the novel data structure and novel algorithms for analyzing the DNA microarray data.

The first part of this work consists of develop the data structure that permits to store and manage efficiently an enormous mass of microarray data. Recently, the microarray data is usually stored by a matrix $E \in \mathcal{M}_{L,N}$ containing L gene profiles across N slides. Nowadays this type of storage is not suitable. Because the capacity of memory is very considerable if $N, L \ggg 1$ and the computing time is not optimal for mining the complex data. In fact, using matric storage, we need $N \times L$ memory space to store the data and the research of an object of size L in this matrix is in

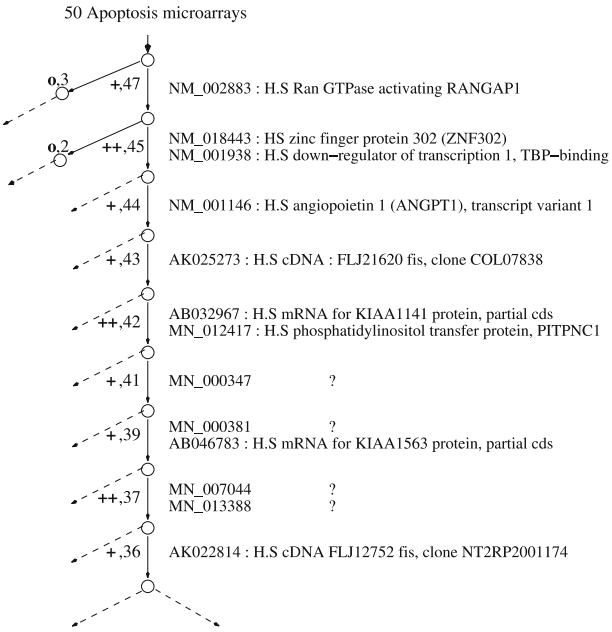


Fig. 7 Visualization of the first depths of counting tree. The counting tree gives the following information: each depth of the tree represents the expression of a gene. At the first depth, the gene NM-002883 is up-regulated on 47 (94%) slides and no-regulated on 3 (6%)slides. At the depth 2 and 3 there are two genes NM-018443 and NM-001938, that are up-regulated on 45 (90%) slides, etc....These genes may be viewed as the group of significant candidate genes of 50 Apoptosis microarrays

$\mathcal{O}(NL)$. In addition, the visualization by a matrix does not make possible to release any analysable result, for example, the common informations of a maximum of the slide profiles. To overcome these difficulties, we introduced the structure of WLPT. Using the WLPT, the complex data of DNA microarrays is structured and indexed such that the common informations between slides will be represented by an internal node associated to a prefix. The research of an object of size L in WLPT is in $\mathcal{O}(L)$. By this way, the WLPT is considered as an engine of the request and the research of informations.

The next essential part is to propose the novel algorithms in order to identify a set of differential regulated genes. Using the WLPT given in Fig. 7, we observe that there are 12 up-regulated genes on least 36 (72%) slides: the first depth gives the gene *NM-002883* up-regulated on 47 slides; the $2^n d$ and $32d$ depth gives the genes *NM-018443* and *NM-001938* up-regulated on 45 slides, *etc.*... We have tried to identify the function of these genes using the genbank tool of site web *http://www.ncbi.nlm.nih.gov/Genbank*: 8 genes are identified and described in the figure; there are 4 genes unknown ("?"). These 12 genes form then a group of co-up-regulated genes along over 36 conditions. This group of genes may be viewed as a group of significant candidate genes of 50 Apoptosis microarrays (may be viewed as the important genes of colon cancer cell). Therefore, the WLPT provides a new tool for identifying groups of *co-regulated* genes across certain conditions known well as the *bi-clustering* problem. The research of a *maximal bi-cluster* is known as

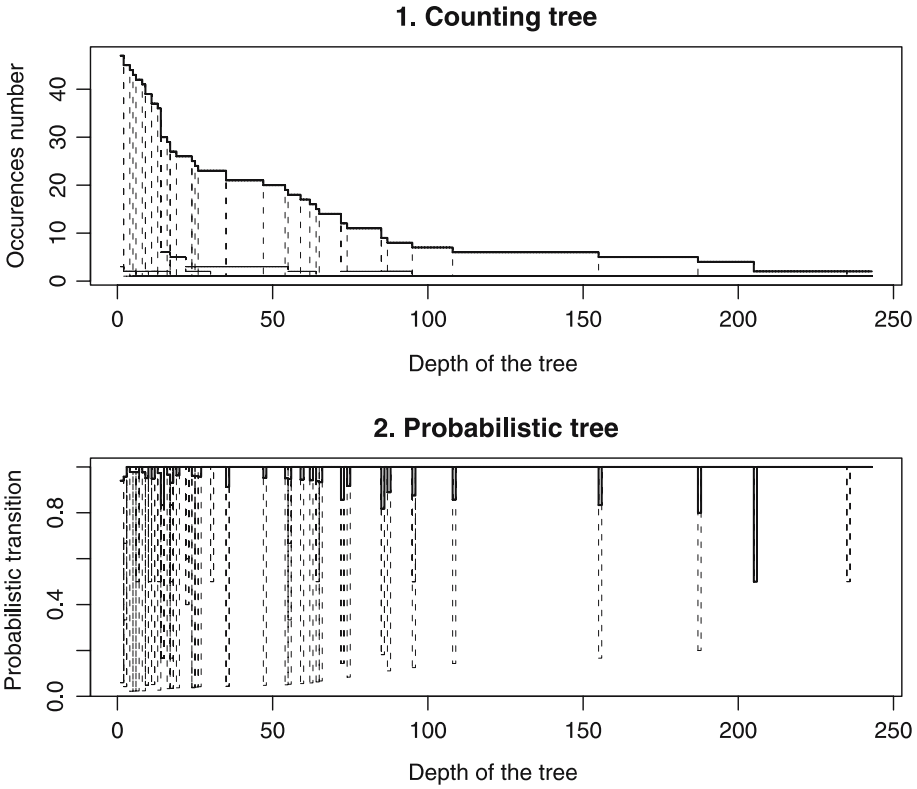


Fig. 8 Construction result of weighted tree. The first graph represents the weight of 50 leaves of the counting tree. Each depth of the tree gives the number of microarrays having a common expression. The x-axis is the depths of tree and the y-axis is the occurrence number between the slides: at the depth 100, for example, there are 7 similar slides on the 100 first gene expressions. The second graph represents the frequency of 50 leaves of the corresponding probabilistic tree. The slides having a maximal probabilistic prefix are represented by a dark line. These results solved the problem P2

NP-complete problem [17]. Once the WLPT constructed, a maximal bi-clusters may be identified exactly in linear time [20].

The results described in Figs. 8 and 9 allow us extract the characteristics by using the appearance probability. By descending to a fixed depth, we compute the probabilistic prefix of the slides and classify those into the different groups by the use of the informational semi-distance (Proposition 4). This method returns then the hierarchical clustering using WLPT which is an unsupervised learning technique and it does not require a priori knowledge of cluster number. This criterion is very important in DNA microarray data analysis since the characteristics of the data are often unknown. Thus, WLPT open a new direction to examine the data-mining step in the process of KDD to understand the characteristics of DNA microarray data. In other words, they permit the extraction of hidden biological and medical informations from the mass of data.

In the present work, we performed the analysis of slide profiles to extract medical informations—the determination of characteristic slides (*characteristic molecular portraits*) of a pathology. With a given small dataset, the results described in this paper introduced thus only a novel methodology and a theoretical aspect. Our ambition in

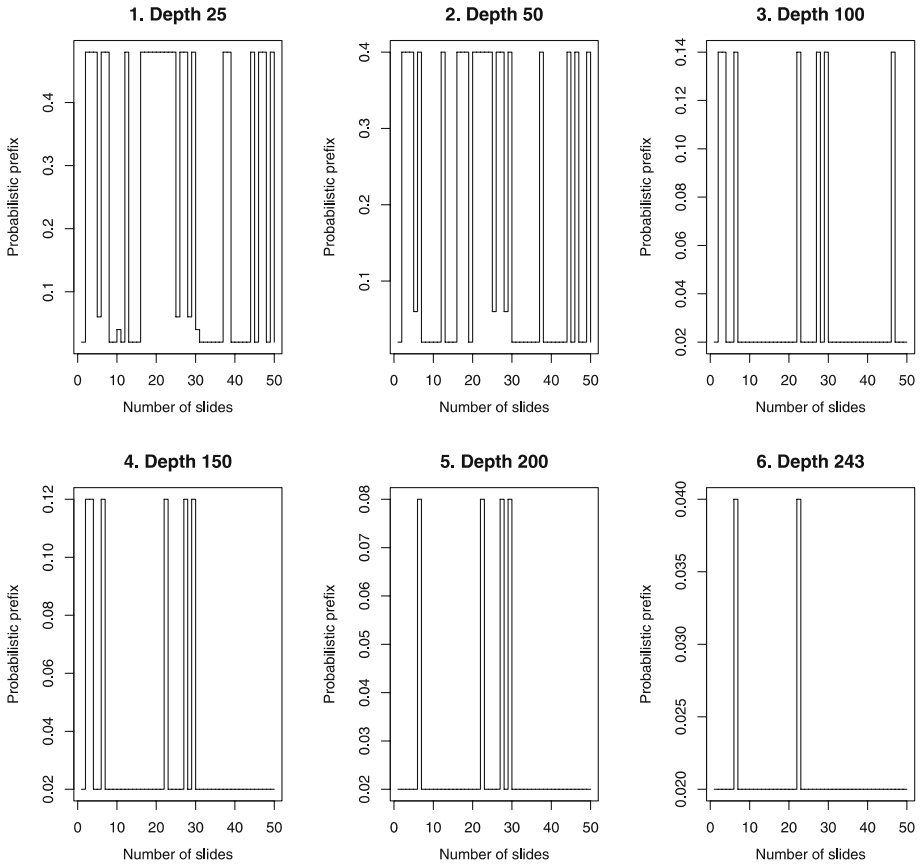


Fig. 9 Classification results of DNA microarrays. The graphs represent the appearance probability of prefixes at each depth of the tree. For example, at the depth 100 of tree, we observe there are 7 slides which are similar on 100 genes with the probability of prefixe of 0.14; at the depth 150 of tree, we observe there are 6 similar slides with the probability of prefixe of 0.12. At the depth 243 there are 2 slides which are similar on 243 gene expression with a probability of 0.04. They are considered as the characteristic slides of the sample studied. Among 243 first genes, there are 232 up-regulated genes, 11 no-regulated genes. This result solved two problems P3 and P4

the future is that the combinatoric method based on WLPT will offer an effective tool for the analysis of DNA microarray data and, in particular, for the diagnosis assisted by the computer. Nevertheless, we would need, for all that, to have the large quantities of samples and to make the reliable and reproducible quantitative measurements. That is not nowadays, because the technique of DNA microarray is still expensive and several technical points must to be improved. In addition, the databases of DNA microarray are not validated (homogenized) between the various platforms of biochip yet, as the link with the clinical data for example.

Using the same structure of WLPT by considering L gene profiles as L words of length N, the gene expression profiles will be analyzed to identify biological informations [19,20] in the transcriptomic analysis process. In fact, the three problems of the transcriptomic analysis are solved by the use of WLPT. The first is the identification of differential expressed genes: identifying the genes whose their expressions is modified

during a normal or pathological cellular process. With their differential expressions, one makes a hypothesis that these genes are implied in this process; The second is the studies of expression profiles. Starting from various conditions, it is possible to establish a common expression profile of which relates to a set of genes of the genome. One makes a hypothesis that the genes presented by the same profile have the same function; The last problem is to determine genes co-regulated under certain specific conditions. It is an essential step for constructing a regulation networks or a biology system [5].

Finally, the algorithmics and combinatorics approaches presented in this paper lead to implant an operational software entiled WLPT@DNA-ARRAY where its kernel is the WLPT and the algorithms of extraction on this tree. The software will answer some problems of the management, the storage and the visualization. It provides an effective tool to compare and classify the DNA microarray data. This software permits thus to make an evidence of the analysis results (see *http://www.genopole-lille.fr/spip/spip.php?article44*).

5 Conclusions and perspectives

The structure of weighted trees is introduced to analyze DNA microarray data. The collection of DNA microarrays was modelled by ternary weighted trees, which are finite weighted acyclic automata. These trees are used to store microarray data. Once the number of slides is large enough, we can deduce the characteristic slides of a microarray collection. The probabilistic tree enables the computation of the appearance probability of a microarray. The longest prefix tree is used to establish the characteristic slides that have the longest common expression and also the determine the groups of genes that are candidates of a pathologic condition. Encouraged by these preliminary results, we intend to construct further diagnostic trees, using these methods. We anticipate that with further refinement these methods will be extremely valuable in analyzing the mass of DNA microarray data, with possible significant clinical applications.

Acknowledgements We would like to thank the Functional Genomic Platform of University of Lille 2 for providing the DNA microarray data in this study.

References

1. Salomaa, A.: Theory of automata. Pergamon Press (1969)
2. Jacob, G.: Langages, Automates, Séries Formelles. Publications no 107, Laboratoire d'Informatique, USTL (1978)
3. Inza, I. et al: Filter versus wrapper gene selection approaches in DNA microarray domains. Artif. Intell. Med. **31**, 91–103 (2004)
4. Berstel J., Reutenauer, C.: Les séries rationnelles et leurs langage. Masson (1984)
5. Boulicaut, J.F., Gandrillon, O.: Informatique pour l'analyse du transcriptome. Hermee-Lavoisier, (2004)
6. Bourdon, J.: Size and path length of Patricia tries: dynamical soures context. Random Struct Algorithms **19**(3–4), 289–315 (2001)
7. Quackenbush, J.: Computational analysis of microarray data. Nat Genet **32**, 509–514 (2001)
8. Sakarovitch, J.: Elements de theorie des automates. Vuibert Informatique (2003)
9. Brown, M., Grundy, W., et al.: Knowledge-based analysis of microarray gene expression data by using support vector machines. University of California (1999)

10. Crochemore, M., et al.: Algorithmique du texte. Vuibert Informatique (2001)
11. Eisen, M., Botstein, D. et al.: Cluster analysis and display genome-wide expression patterns. Pro. Natl. Acad. Sci **95**(25), 14863–14868 (1998)
12. Fliess, M.: Automates stochastiques et séries rationnelles non commutatives. In: Nivat, M. (ed.), Proceedings International Colloquium on Automata. Languages and Programming (ICALP), pp. 397–411 (1972)
13. Kerr, M., Churchill, G.: Experimental design for gene expression microarrays. Biostatistics **2**, 183–201 (2001)
14. Lothaire, M.: Combinatorics on Words. Addison-Wesley Publishing (1983)
15. Flajolet, P., Sedgewick, R.: An Introduction to the Analysis of Algorithms. Addison-Wesley (1996)
16. Walker, P., Famili, A. et al.: Data mining of gene expression changes in Alzheimer brain. Artif. Intell. Med **31**, 137–154 (2004)
17. Peeter, R.: The maximum edge biclique problem is NP-complete. Discrete Appl. Math. **131**(3), 651–654 (2000)
18. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley Interscience Publication (1991)
19. Tran, T., Nguyen, C.C., Minh, H.N.: Data mining of gene expression microarray via weighted prefix trees. In: Proceedings PAKDD'05, LNAI 3518, pp. 21–31. Springer Verlag (2005)
20. Tran, T., Nguyen, C.C., Minh, H.N.: Bi-clustering des donns de biopuces par les arbres pondérés de plus long préfixe, accepté. TSI. vol. Modisation et Simulation pour la Post-Gomique, Hemes-Lavoisier (2006)
21. Szpankowski, W.: Average Case Analysis of Algorithms on Sequences. Wiley (2001)