New Generation Computing (2019) 37:325–337
https://doi.org/10.1007/s00354-019-00052-4

Modeling of Decision Trees Through P Systems

José M. Sempere[1]

Received: 12 October 2018 / Accepted: 31 January 2019 / Published online: 14 February 2019

Abstract
In this paper, we propose a decision-tree modeling in the framework of membrane computing. We propose an algorithm to obtain a P system that is equivalent to any decision tree taken as input. In our case, and unlike previous proposals, we formulate the concepts of decision trees endogenously, since there is no external agent involved in the modeling. The tree structure can be defined naturally by the topology of the regions in the P system and the decision rules are defined by communication rules of the P system.

Keywords Decision trees · Membrane computing · P systems with communicating rules · Classification methods

Introduction

Decision trees are tree-structured classification models that have been widely used in different application domains such as bioinformatics, pattern recognition and data mining, among others [14]. This is one of the most used classifiers in the field of machine learning. Furthermore, it is the base classifier used to produce random forest in the framework of ensemble classification methods [16]. Several learning algorithms have been proposed that allow the inference of representations of decision trees from classification examples [9]. In this paper, we propose a modeling of decision trees using P systems. P systems are the models that support the computing paradigm known as membrane computing [11]. One of the biggest advantages of using this paradigm to model decision trees lies in its massive parallelism that allows efficient implementations to work with a large amount of data that can be classified into complex decision trees. In fact, it has been possible to show the plausibility of the implementation of the P systems through technology based on GPUs which makes it an affordable technology in most cases [3, 8, 17].

✉ José M. Sempere
jsempere@dsic.upv.es

1 Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain

The definition of P systems to model decision trees has been previously approached in different works. For example, Díaz-Pernil et al. [4] proposed recognizer P systems to define decision trees. Their proposal is based on a non-deterministic search for structures that are compatible with examples of the classification that the decision tree must carry out. Wang et al. [15] proposed the use of tissue-like P systems with tree-like objects. They applied evolutionary strategies to explore a searching space using non-deterministic P systems. In our approach, we model decision trees using basic concepts provided by the P systems: on one hand, the tree structure is defined immediately through the tree-like structure of the regions of a cell-like P system, and on the other hand, the definition of the rules of the decision tree can be defined by communicating rules in the P system in a very simple way. Therefore, we believe that our proposal is better adapted to the use of P systems in a more natural way than the proposals referred to above. In addition, in our approach, the implementation in parallel hardware platforms is achieved more easily by eliminating the non-determinism and using a more simple and comprehensible encoding of the data to be classified.

The structure of this work is as follows: first, we introduce basic concepts about decision trees, then, we define the main components of the P systems, and we provide a description of how these systems work. We propose an algorithmic scheme to translate decision trees to cell-like P systems with communicating rules, and we solve the classification task with the proposed system. Finally, we describe some works in progress related to this topic.

Basic Concepts

In this section, we introduce basic concepts of decision trees from [9, 14] and basic concepts about P systems and membrane computing from [10, 11].

Decision Trees

In the following, we consider objects with a finite set of discrete value attributes $\mathcal{A} = \{a_1, a_2, \dots, a_m, c\}$, where c is a special attribute that designates the class of the object. Every attribute a_i or c can take a value from a finite set. The set of values that can be assignated for the attribute a_i is $\{v_{i_1}, \dots, v_{i_j}\}$, while the set of values for the attribute c is defined by $\{v_{c_1}, \dots, v_{c_p}\}$. For the case of continuous-valued attributes, it is an additional task to define thresholds for the intervals that allow the discretization of numerical values. Therefore, for example, if we define two thresholds c_1 and c_2 with $c_1 < c_2$ and we assign the label 'low' for the values lower than c_1, 'medium' for the values greater than c_1 and lower than c_2 and ' high' for the values greater than c_2, then, we can assign discrete values ('low', 'medium', or 'high') to the integer values to be considered in the task. The choice of thresholds should favor the attribute selection criterion in the construction of decision trees from a machine-learning point of view [5].

For every set of attributes $\mathcal{A} = \{a_1, a_2, \dots, a_m, c\}$, we can define the regular expression[1] $\mathcal{A}_{reg}$ as follows:

$$(v_{11} + v_{12} + \cdots + v_{1i_1})(v_{21} + v_{22} + \cdots + v_{2i_2}) \cdots (v_{m1} + v_{m2} + \cdots + v_{mi_m}).$$

A decision tree over $\mathcal{A}$ is a tree, where every node is either a leaf (with a value for the attribute c) or an internal node with a label from $\{a_1, \dots, a_m\}$. Every internal node, with label a_i, denotes a test over the attribute, and every descending branch from node a_i means that the attribute fulfills a logical statement constructed by the relational operators over the attribute values. We consider the set of relational operators $\{=, >, <, \neq, \geq, \leq\}$. Let us see the following example that illustrates this concept.

Example 1 Let us consider the following set of attributes with their respective values that corresponds to an adaptation of the *protein–protein interaction prediction problem* taken from [6]. The attribute Interaction is the class attribute.

Attribute	Values
Expression correlation (EC)	{0.1, 0.3, 0.7, 0.9}
Shared location (SL)	{Yes, No}
Genomic distance (GD)	{Yes, No}
Shared function (SF)	{Yes, No}
Interaction	{Yes, No}

Let us consider the threshold value {0.7} for the attribute EC. In this case, this threshold produces the discrete values $\{\mathrm{EC}_{\leq 0.7}, \mathrm{EC}_{>0.7}\}$. The regular expression defined from the set of attributes and values is the following one:

$$(\mathrm{EC}_{\leq 0.7} + \mathrm{EC}_{>0.7})(\mathrm{SL}_\mathrm{Y} + \mathrm{SL}_\mathrm{N})(\mathrm{GD}_\mathrm{Y} + \mathrm{GD}_\mathrm{N})(\mathrm{SF}_\mathrm{Y} + \mathrm{SF}_\mathrm{N}).$$

In Fig. 1, we show a decision tree over the set of attributes previously defined. We have drawn the nodes of the class attribute as ellipses and the rest of attributes as boxes

Any decision tree for a given classification object receives as input a tuple of values for the attributes, and outputs a value for the class attribute. The main problem from the point of view of machine learning is to find the best decision tree that fits the input data according to a pre-established criterion (which is usually established in terms of information measures or benefits in the classification of new examples). Over time, various learning algorithms on decision trees have been proposed. From our point of view, the best known algorithms are Quinlan's ID3 and C4.5 [12] and the CART algorithm for classification and regression trees [1].

1 A regular expression is a classical concept from formal language theory that, in this case, represents a sequence of values obtained by selecting a value of each set defined for the sum of values within a pair of parentheses.

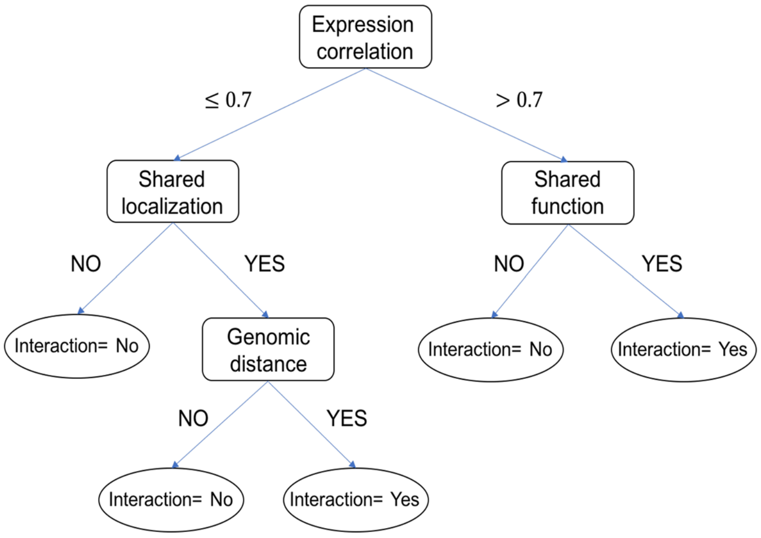

Fig. 1 Decision tree for the set of attributes adapted from [6]

P Systems with Evolution and Communication Rules

In the following, we introduce basic concepts about P systems from [10] and [11], and we define cell-like P systems with communication rules.

First, we define multisets as follows: let D be a set. A multiset over D is a pair $\langle D,f\rangle$, where $f : D \longrightarrow \mathbb{N}$ is a function. The size of a multiset M is the number of elements that it contains and it is denoted by $|M|$, that is

$$|M| = \sum_{a\in D} f(a).$$

Any multiset $\langle D,f\rangle$ where $D = \{a_1, \dots, a_n\}$ can be represented by the finite sequence $a_1^{f(a_1)} \cdots a_n^{f(a_N)}$ that is the representative string of the multiset.

Basically, a P system is defined as a finite set of regions separated by membranes and organized hierarchically, so that it can be defined by a tree-like structure. Within each region, there is a multiset of objects over a previously defined alphabet. In each region, there is a finite set of rules that transform objects (evolution rules) or rules that send objects from one region to an adjacent region (communication rules). The whole system is encompassed in a special region delimited by a skin membrane and the outside environment provides new objects to the system and picks up the objects that the system expels outside the region of the skin.

We provide a formal definition of the P systems as follows.

Definition 1 A P system with evolution and communication rules of degree $m \geq 1$ is defined by the tuple $\Pi = (O, H, \mu, w_1, w_2, \dots, w_m, R, i_0)$, where

1. O is the alphabet of objects. O^* denotes the set of all the strings defined over O.

2. H is the alphabet of labels for membranes.
3. μ is the membrane structure, of degree m, with all membrane labels from H. A membrane with label h is represented by $[\ ]_h$.
4. $w_1, w_2, \dots, w_m$ are strings over O that define the multisets of objects in every region of μ.
5. R is a finite set of rules of the following types:

    (a) $[v \to w]_h$ with $v, w \in O^*$ (evolution rules).
    The objects denoted by the multiset v are transformed into the objects denoted by the multiset w in the region delimited by the membrane with label h.

    (b) $v[\ ]_h \to [w]_h$ with $v, w \in O^*$ (*'in' communication rules*).
    The objects denoted by the multiset v are transformed into the objects denoted by the multiset w and they are sent into the internal region bounded by the membrane with label h.

    (c) $[v]_h \to w[\ ]_h$ with $v, w \in O^*$ (*'out' communication rules*).
    The objects denoted by the multiset v are transformed into the objects denoted by the multiset w and they are sent out the region bounded by the membrane with label h. If the membrane h is the *skin* membrane, the objects denoted by w are sent out to the environment; otherwise, the objects are sent out to the region that contains the membrane with label h.

6. $i_0 \in \{0, 1, \dots, m\}$ is the region where the result of a computation is obtained (0 represents the environment).

The rules of the P system are applied in a non-deterministic maximally parallel manner. Maximal parallelism means that the rules should be used in parallel to the maximum degree possible. That is, for any rule that can be applied more than once simultaneously, the rule is applied the maximum number of times that allows the objects that enable the execution. Non-determinism is a classic concept in computability theory, which, in this case, means that if several rules can be applied over the same objects, the selection of the rule to apply is non-deterministic. That is, in several identical situations, the rules that are applied may be different. The computation of the system finishes whenever no rule can be applied.

A system configuration at time t is defined by the multisets in each region of the structure defined by μ. The initial configuration of the system is defined by the multisets $w_1, w_2, \dots, w_m$. The configuration during a computation at time t is defined by the multisets $w_1^t, w_2^t, \dots, w_m^t$.

From Decision Trees to P Systems

In this section, we propose an algorithm that obtains P systems with communication rules from decision trees.

```
              CreateInternalRules($T, \mathcal{A}$)

              /* Let $a_i \in \mathcal{A}$ be the (internal) attribute of the root of T */

(1)           For every branch with value $v_j$ that goes to (internal) attribute $b$ in
              preorder traversal
(1.1)             For every string $x \in (\widehat{\mathcal{A}} - \{a_i\})_{reg}$
(1.1.1)               add the rule $[a_{ij}x[\,]_b]_{a_i} \rightarrow [\,[x]_b]_{a_i}$

              /* Let $T_b$ be the subtree of $T$ rooted in the attribute $b$ */

(1.2)             CreateRules($T_b$,$\mathcal{A} - \{a_i\}$)

(2)           For every branch with value $v_j$ that goes to class attribute $c$ with
              value $c_k$
(2.1)             For every string $x \in (\widehat{\mathcal{A}} - \{a_i\})_{reg}$
(2.1.1)               add the rule $[a_{ij}x]_{a_i} \rightarrow [\,]_{a_i}c_k$
```

```
              CreateClassificationRules($\mathcal{A}$)

(1)           For every attribute $a \in \mathcal{A}$
(1.1)             For every value $c_k$ of the class attribute $c$
(1.1.1)               add the rule $[c_k]_{a_i} \rightarrow [\,]_{a_i}c_k$
```

Fig. 2 Two algorithms to obtain P system rules from the decision trees and the set of attributes

Let T be a decision tree defined for a set of attributes, and let $\mathcal{A} = \{a_1, a_2, \dots, a_n, c\}$ be the set of attributes in a preorder enumeration according to the tree structure, with c as the class attribute. Let $\widehat{\mathcal{A}} = \mathcal{A} - \{c\}$, and $\widehat{\mathcal{A}}_{reg}$ denotes the regular expression associated to $\widehat{\mathcal{A}}$.

We propose a P system $\Pi_T = (O_T, H_T, \mu_T, w_1, w_2, \dots, w_{|\mathcal{A}|-1}, R, 0)$, where

1. The alphabet of objects is defined from the set of values for every attribute in $\widehat{\mathcal{A}}$. Therefore, $a_{ij} \in O_T$ if the attribute a_i can take the value v_j. Observe that, for any numerical attribute, we define a set of values according to a previously defined set of intervals according to the decision tree. Every value for the class attribute defines an object in O.
2. The set of membrane labels is the set of attributes with the exception of the class attribute, that is $H_T = \widehat{\mathcal{A}}$.
3. The membrane structure μ_T is defined to be equivalent to the decision-tree structure. If the attribute a_j is a son of the attribute a_i in the tree T, then the substructure $[\,[\,]_j\,]_i$ is defined in μ_T.
4. Initially, all the multisets of objects are empty, $w_i = \lambda$ for $1 \leq i \leq m$.
5. The set of rules R are defined from the decision tree T using the algorithms of Fig. 2.

   The algorithm **CreateInternalRules(T,A)** deals with the creation of rules corresponding to the branches of the decision tree. In this way, based on the fact that the set of values to be evaluated are those of the attribute a_i, which are found in the region with membrane a_i, we replicate the behavior of the decision tree by moving the remains of attribute values to the region b which will correspond to

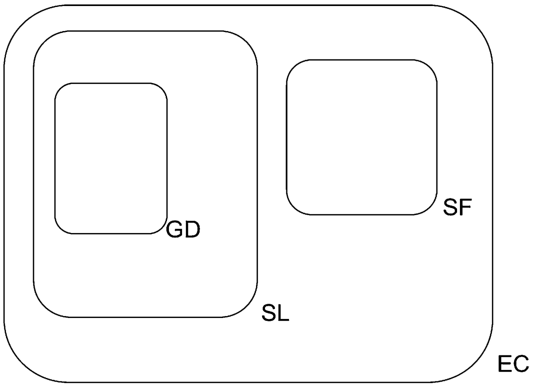

Fig. 3 Membrane structure for the decision tree of Fig. 1

the next attribute to be evaluated according to the decision tree. Note that the rules defined in point (2.1.1) allow to obtain the value of the class attribute according to the decision tree. The algorithm **CreateClassificationRules(A)** deals with the creation of the rules that move the class attribute through the system regions and, finally, send it to the environment.

Example 2 Let us consider the classification task of Example 1 and the decision tree of Fig. 1. Initially, $\widehat{\mathcal{A}}_{reg}$ is defined as follows

$$(\mathrm{EC}_{\leq 0.7} + \mathrm{EC}_{>0.7})(\mathrm{SL}_{\mathrm{Y}} + \mathrm{SL}_{\mathrm{N}})(\mathrm{GD}_{\mathrm{Y}} + \mathrm{GD}_{\mathrm{N}})(\mathrm{SF}_{\mathrm{Y}} + \mathrm{SF}_{\mathrm{N}}).$$

We define the following P system

$$\Pi_{\mathrm{T}} = (O_{\mathrm{T}}, H_{\mathrm{T}}, \mu_{\mathrm{T}}, w_1, w_2, \dots, w_{|\mathcal{A}|-1}, R, 0)$$

that is equivalent to the referred decision tree, where

- $O_{\mathrm{T}} = \{\mathrm{EC}_{\leq 0.7}, \mathrm{EC}_{>0.7}, \mathrm{SL}_{\mathrm{Y}}, \mathrm{SL}_{\mathrm{N}}, \mathrm{GD}_{\mathrm{Y}}, \mathrm{GD}_{\mathrm{N}}, \mathrm{SF}_{\mathrm{Y}}, \mathrm{SF}_{\mathrm{N}}, \mathrm{YES}, \mathrm{NO}\}$
- $H_{\mathrm{T}} = \{\mathrm{EC}, \mathrm{SL}, \mathrm{GD}, \mathrm{SF}\}$
- $\mu_{\mathrm{T}} = [\ [\ [\ ]_{\mathrm{GD}}\ ]_{\mathrm{SL}}\ [\ ]_{\mathrm{SF}}\ ]_{\mathrm{EC}}$. The structure is showed in Fig. 3.
- $w_{\mathrm{EC}} = w_{\mathrm{SL}} = w_{\mathrm{SF}} = w_{\mathrm{GD}} = \lambda$ (the empty multiset)
- R is defined as follows.

  For the case that EC is less than or equals to 0.7, we must evaluate the SL attribute first, and then the rest of attributes. It is carried out using the following rules:

  1. $[\mathrm{EC}_{\leq 0.7}\mathrm{SL}_{\mathrm{Y}}\mathrm{GD}_{\mathrm{Y}}\mathrm{SF}_{\mathrm{Y}}[\ ]_{\mathrm{SL}}\ ]_{\mathrm{EC}} \rightarrow [\ [\mathrm{SL}_{\mathrm{Y}}\mathrm{GD}_{\mathrm{Y}}\mathrm{SF}_{\mathrm{Y}}]_{\mathrm{SL}}\ ]_{\mathrm{EC}}.$
  2. $[\mathrm{EC}_{\leq 0.7}\mathrm{SL}_{\mathrm{Y}}\mathrm{GD}_{\mathrm{Y}}\mathrm{SF}_{\mathrm{N}}[\ ]_{\mathrm{SL}}\ ]_{\mathrm{EC}} \rightarrow [\ [\mathrm{SL}_{\mathrm{Y}}\mathrm{GD}_{\mathrm{Y}}\mathrm{SF}_{\mathrm{N}}]_{\mathrm{SL}}\ ]_{\mathrm{EC}}.$
  3. $[\mathrm{EC}_{\leq 0.7}\mathrm{SL}_{\mathrm{Y}}\mathrm{GD}_{\mathrm{N}}\mathrm{SF}_{\mathrm{Y}}[\ ]_{\mathrm{SL}}\ ]_{\mathrm{EC}} \rightarrow [\ [\mathrm{SL}_{\mathrm{Y}}\mathrm{GD}_{\mathrm{N}}\mathrm{SF}_{\mathrm{Y}}]_{\mathrm{SL}}\ ]_{\mathrm{EC}}.$
  4. $[\mathrm{EC}_{\leq 0.7}\mathrm{SL}_{\mathrm{Y}}\mathrm{GD}_{\mathrm{N}}\mathrm{SF}_{\mathrm{N}}[\ ]_{\mathrm{SL}}\ ]_{\mathrm{EC}} \rightarrow [\ [\mathrm{SL}_{\mathrm{Y}}\mathrm{GD}_{\mathrm{N}}\mathrm{SF}_{\mathrm{N}}]_{\mathrm{SL}}\ ]_{\mathrm{EC}}.$
  5. $[\mathrm{EC}_{\leq 0.7}\mathrm{SL}_{\mathrm{N}}\mathrm{GD}_{\mathrm{Y}}\mathrm{SF}_{\mathrm{Y}}[\ ]_{\mathrm{SL}}\ ]_{\mathrm{EC}} \rightarrow [\ [\mathrm{SL}_{\mathrm{N}}\mathrm{GD}_{\mathrm{Y}}\mathrm{SF}_{\mathrm{Y}}]_{\mathrm{SL}}\ ]_{\mathrm{EC}}.$
  6. $[\mathrm{EC}_{\leq 0.7}\mathrm{SL}_{\mathrm{N}}\mathrm{GD}_{\mathrm{Y}}\mathrm{SF}_{\mathrm{N}}[\ ]_{\mathrm{SL}}\ ]_{\mathrm{EC}} \rightarrow [\ [\mathrm{SL}_{\mathrm{N}}\mathrm{GD}_{\mathrm{Y}}\mathrm{SF}_{\mathrm{N}}]_{\mathrm{SL}}\ ]_{\mathrm{EC}}.$

7. $[EC_{\leq 0.7}SL_NGD_NSF_Y[\,]_{SL}\,]_{EC} \rightarrow [\,[SL_NGD_NSF_Y]_{SL}\,]_{EC}$.
8. $[EC_{\leq 0.7}SL_NGD_NSF_N[\,]_{SL}\,]_{EC} \rightarrow [\,[SL_NGD_NSF_N]_{SL}\,]_{EC}$.

Once we have evaluated the attribute EC, for the case that the attribute SL is positive we must evaluate the GD attribute. It is carried out using the following rules:

9. $[SL_YSF_YGD_Y[\,]_{GD}\,]_{SL} \rightarrow [\,[SF_YGD_Y]_{GD}\,]_{SL}$.
10. $[SL_YSF_YGD_N[\,]_{GD}\,]_{SL} \rightarrow [\,[SF_YGD_N]_{GD}\,]_{SL}$.
11. $[SL_YSF_NGD_Y[\,]_{GD}\,]_{SL} \rightarrow [\,[SF_NGD_Y]_{GD}\,]_{SL}$.
12. $[SL_YSF_NGD_N[\,]_{GD}\,]_{SL} \rightarrow [\,[SF_NGD_N]_{GD}\,]_{SL}$.

The rules needed to evaluate the attribute GD, and set the value for the class of the object are the following, according to the decision tree:

13. $[SF_YGD_Y\,]_{GD} \rightarrow [\,]_{GD}YES$.
14. $[SF_NGD_Y\,]_{GD} \rightarrow [\,]_{GD}YES$.
15. $[SF_YGD_N\,]_{GD} \rightarrow [\,]_{GD}NO$.
16. $[SF_NGD_N\,]_{GD} \rightarrow [\,]_{GD}NO$.

For the case that EC is greater than 0.7, we must evaluate the SF attribute first, and then the rest of attributes. It is carried out using the following rules:

17. $[EC_{>0.7}SL_YSF_YGD_Y[\,]_{SF}\,]_{EC} \rightarrow [\,[SL_YSF_YGD_Y]_{SF}\,]_{EC}$.
18. $[EC_{>0.7}SL_YSF_YGD_N[\,]_{SF}\,]_{EC} \rightarrow [\,[SL_YSF_YGD_N]_{SF}\,]_{EC}$.
19. $[EC_{>0.7}SL_YSF_NGD_Y[\,]_{SF}\,]_{EC} \rightarrow [\,[SL_YSF_NGD_Y]_{SF}\,]_{EC}$.
20. $[EC_{>0.7}SL_YSF_NGD_N[\,]_{SF}\,]_{EC} \rightarrow [\,[SL_YSF_NGD_N]_{SF}\,]_{EC}$.
21. $[EC_{>0.7}SL_NSF_YGD_Y[\,]_{SF}\,]_{EC} \rightarrow [\,[SL_NSF_YGD_Y]_{SF}\,]_{EC}$.
22. $[EC_{>0.7}SL_NSF_YGD_N[\,]_{SF}\,]_{EC} \rightarrow [\,[SL_NSF_YGD_N]_{SF}\,]_{EC}$.
23. $[EC_{>0.7}SL_NSF_NGD_Y[\,]_{SF}\,]_{EC} \rightarrow [\,[SL_NSF_NGD_Y]_{SF}\,]_{EC}$.
24. $[EC_{>0.7}SL_NSF_NGD_N[\,]_{SF}\,]_{EC} \rightarrow [\,[SL_NSF_NGD_N]_{SF}\,]_{EC}$.

The rules needed to evaluate the attribute SF, and to set the value for the class of the object are the following, according to the decision tree:

25. $[SL_YSF_NGD_N\,]_{SF} \rightarrow [\,]_{SF}NO$.
26. $[SL_YSF_NGD_Y\,]_{SF} \rightarrow [\,]_{SF}NO$.
27. $[SL_YSF_YGD_N\,]_{SF} \rightarrow [\,]_{SF}YES$.
28. $[SL_YSF_YGD_Y\,]_{SF} \rightarrow [\,]_{SF}YES$.
29. $[SL_NSF_NGD_N\,]_{SF} \rightarrow [\,]_{SF}NO$.
30. $[SL_NSF_NGD_Y\,]_{SF} \rightarrow [\,]_{SF}NO$.
31. $[SL_NSF_YGD_N\,]_{SF} \rightarrow [\,]_{SF}YES$.
32. $[SL_NSF_YGD_Y\,]_{SF} \rightarrow [\,]_{SF}YES$.

Finally, the necessary rules to expel to the environment the value of the classification attribute are showed. Observe that rules (37) to (40) are use-less given that the object configuration required to execute the rule will never appear during the computation time:

33. $[\mathrm{NO}]_{\mathrm{EC}} \rightarrow [\,]_{\mathrm{EC}}\mathrm{NO}$.
34. $[\mathrm{YES}]_{\mathrm{EC}} \rightarrow [\,]_{\mathrm{EC}}\mathrm{YES}$.
35. $[\mathrm{NO}]_{\mathrm{SL}} \rightarrow [\,]_{\mathrm{SL}}\mathrm{NO}$.
36. $[\mathrm{YES}]_{\mathrm{SL}} \rightarrow [\,]_{\mathrm{SL}}\mathrm{YES}$.
37. $[\mathrm{NO}]_{\mathrm{GD}} \rightarrow [\,]_{\mathrm{GD}}\mathrm{NO}$.
38. $[\mathrm{YES}]_{\mathrm{GD}} \rightarrow [\,]_{\mathrm{GD}}\mathrm{YES}$.
39. $[\mathrm{NO}]_{\mathrm{SF}} \rightarrow [\,]_{\mathrm{SF}}\mathrm{NO}$.
40. $[\mathrm{YES}]_{\mathrm{SF}} \rightarrow [\,]_{\mathrm{SF}}\mathrm{YES}$.

Objects' Classification

Suppose that we have a finite set of objects to be classified using a decision tree. For each object, we have the values of its attributes and we want to obtain the value of the class attribute that is the only one that is unknown. The classification of objects according to a decision tree specified in a P system can be carried out in two ways: first, we can carry out a sequential classification, that is, we provide the P system with only one example each time, and the P system returns the value for the class attribute through the environment. Alternatively, we can carry out a classification in parallel, where we provide the P system with all the objects to be classified, and the P system returns a value of the class attribute for each object through the environment.

We will describe each of these possibilities separately.

Sequential Classification

We consider the classification of only one object every time. In this case, we have a value for each attribute of the object, and initially, we place them in the skin region of the P system. In this way, we consider a set of attributes $\mathcal{A}$, and a set of attribute values $X = \{x_1, x_2, \dots, x_{|\mathcal{A}|-1}\}$, where we do not include a value for the class attribute. Then, the P system is defined as $\Pi_{\mathrm{T}} = (O_{\mathrm{T}}, H_{\mathrm{T}}, \mu_{\mathrm{T}}, w_1, w_2, \dots, w_{|\mathcal{A}|-1}, R, 0)$, where $w_1 = x_1 x_2 \cdots x_{|\mathcal{A}|-1}$, and the rest of the elements are defined according to the proposal explained in the previous section.

The classification is carried out by means of the execution of the rules of the P system. Note that, in this case, only one rule is executed in each computation step, and we can consider that the system is sequential. The object that is received in the environment of the system is the classification value for the object whose attributes we have initially placed in the skin region.

Next, we will see an example that illustrates the mechanism of action that we have just described.

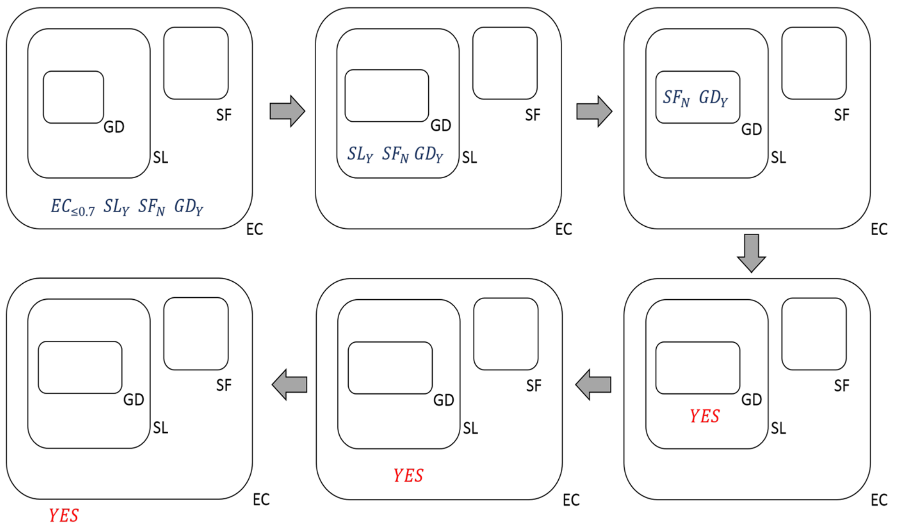

Fig. 4 Sequential classification of objects

Example 3 Let us consider the P system defined in Example 2 for the classification task of Example 1, according to the decision tree of Fig. 1. Let us classify an object with the set of attribute values $\{\mathrm{EC}_{\le 0.7}, \mathrm{SL}_\mathrm{Y}, \mathrm{SF}_\mathrm{N}, \mathrm{GD}_\mathrm{Y}\}$.

The sequence of rules to be applied in the P system is the following: let us consider the rule enumeration of Example 2. First, we apply rule number (3), and the objects SL_Y, SF_N, and GD_Y are sent to the region SL. Then, inside region SL, rule number (11) is applied and the objects SF_N and GD_Y are sent to the region GD. In region GD, the rule number (14) is applied, and the classification of the object is carried out as the object YES. This object is moved through the system using rules (36) and (34), and finally, it is expelled from the system to the environment.

In Fig. 4, we can see graphically the computation steps that are carried out in the P system and that produce as output the classification object YES that is collected in the environment. It means that, for the object with the initial attributes, there is a protein–protein interaction.

Parallel Classification

In this case, by starting from a set of attributes $\mathcal{A}$, we consider that we have a set of objects $X = \{x^1, x^2, \dots, x^p\}$, where each object has assigned values for each attribute (except for the class attribute). For example, given the object x^i, we denote the set of its attributes as $\{x^i_1, x^i_2, \dots, x^i_{|\mathcal{A}|-1}\}$. The rules of the P system are replicated by renaming their objects according to the object-value notation that we have just established. For example, if we have two objects x_1 and x_2 and the rule $a_{11}a_{21} \cdots a_{p1}[\ \ ]_h \rightarrow [a_{21} \cdots a_{p1}]_h$, then the pair of rules $a^1_{11}a^1_{21} \cdots a^1_{p1}[$

$]_h \rightarrow [a^1_{21} \cdots a^1_{p1}]_h$ and $a^2_{11}a^2_{21} \cdots a^2_{p1}[\,]_h \rightarrow [a^2_{21} \cdots a^2_{p1}]_h$ are produced. This allows us to process the attributes of each object in parallel, although independently, even though the attributes of different objects are placed in the same region during any computing step.

The P system is defined as $\Pi_T = (O_T, H_T, \mu_T, w_1, w_2, \dots, w_{|\mathcal{A}|-1}, R, 0)$, where $w_1 = x^1_1 x^1_2 \cdots x^1_{|\mathcal{A}|-1} \cdots x^p_1 x^p_2 \cdots x^p_{|\mathcal{A}|-1}$, the objects and the rules of the system are replicated according to our previous explanation, while the set of membrane labels and the membrane structure is defined as in the sequential case. Note that, in this case, different rules can be executed in each computation step, and the system runs in parallel. The objects that are received in the environment during the computation time are the classification values for every object, whose attributes we have initially placed in the skin region. Observe that we can collect the classification results at different computation steps (depending on the number of rules that are needed to carry out the classification), and every class attribute is labeled with the object identification.

We will see an example that illustrates the mechanism of action that we have just described.

Example 4 Let us consider the P system defined in Example 2 for the classification task of Example 1, according to the decision tree of Fig. 1. Let us classify two objects with the set of attribute values $x_1 = \{\mathrm{EC}^1_{\leq 0.7}, \mathrm{SL}^1_{\mathrm{Y}}, \mathrm{SF}^1_{\mathrm{N}}, \mathrm{GD}^1_{\mathrm{Y}}\}$, and $x_2 = \{\mathrm{EC}^2_{>0.7}, \mathrm{SL}^2_{\mathrm{Y}}, \mathrm{SF}^2_{\mathrm{N}}, \mathrm{GD}^2_{\mathrm{N}}\}$.

In this case, we have replicated all the rules that have defined in Example 2. The replicated rules that are used for the classification task are the following:

- $[\mathrm{EC}^1_{\leq 0.7}\mathrm{SL}^1_{\mathrm{Y}}\mathrm{GD}^1_{\mathrm{Y}}\mathrm{SF}^1_{\mathrm{N}}[\,]_{\mathrm{SL}}\,]_{\mathrm{EC}} \rightarrow [\,[\mathrm{SL}^1_{\mathrm{Y}}\mathrm{GD}^1_{\mathrm{Y}}\mathrm{SF}^1_{\mathrm{N}}]_{\mathrm{SL}}\,]_{\mathrm{EC}}$
- $[\mathrm{SL}^1_{\mathrm{Y}}\mathrm{SF}^1_{\mathrm{N}}\mathrm{GD}^1_{\mathrm{Y}}[\,]_{\mathrm{GD}}\,]_{\mathrm{SL}} \rightarrow [\,[\mathrm{SF}^1_{\mathrm{N}}\mathrm{GD}^1_{\mathrm{Y}}]_{\mathrm{GD}}\,]_{\mathrm{SL}}$
- $[SF^1_{\mathrm{N}}\mathrm{GD}^1_{\mathrm{Y}}\,]_{\mathrm{GD}} \rightarrow [\,]_{\mathrm{GD}}\mathrm{YES}^1$
- $[\mathrm{YES}^1]_{\mathrm{SL}} \rightarrow [\,]_{\mathrm{SL}}\mathrm{YES}^1$
- $[\mathrm{YES}^1]_{\mathrm{EC}} \rightarrow [\,]_{\mathrm{EC}}\mathrm{YES}^1$
- $[\mathrm{EC}^2_{>0.7}\mathrm{SL}^2_{\mathrm{Y}}\mathrm{SF}^2_{\mathrm{N}}\mathrm{GD}^2_{\mathrm{N}}[\,]_{\mathrm{SF}}\,]_{\mathrm{EC}} \rightarrow [\,[SL^2_{\mathrm{Y}}\mathrm{SF}^2_{\mathrm{N}}\mathrm{GD}^2_{\mathrm{N}}]_{\mathrm{SF}}\,]_{\mathrm{EC}}$
- $[\mathrm{SL}^2_{\mathrm{Y}}\mathrm{SF}^2_{\mathrm{N}}\mathrm{GD}^2_{\mathrm{N}}\,]_{\mathrm{SF}} \rightarrow [\,]_{\mathrm{SF}}\mathrm{NO}^2$
- $[\mathrm{NO}^2]_{\mathrm{EC}} \rightarrow [\,]_{\mathrm{EC}}\mathrm{NO}^2$.

The application of the rules is similar, as we have described in Example 3. Observe that, in this case, at every computation step, more than one rule is applied simultaneously, to manage each sequence of objects independently. In Fig. 5, we can see graphically the computation steps that are carried out in the P system and that produce as output the classification objects YES^1 for the object x_1, and NO^2 for the object x_2. They are collected in the environment at computation steps 3 and 5. It means that, for the object x_1 with the initial attributes, there is a protein–protein interaction, while for the object x_2, the interaction does not exist.

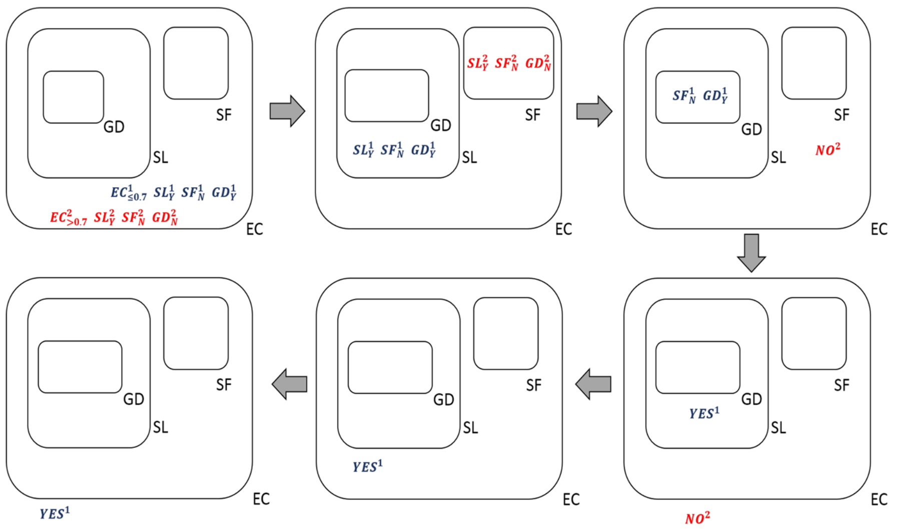
GD
SF
SL
EC
$EC^1_{\leq 0.7}$ SL^1_Y SF^1_N GD^1_Y
$EC^2_{>0.7}$ SL^2_Y SF^2_N GD^2_N
SL^2_Y SF^2_N GD^2_N
SL^1_Y SF^1_N GD^1_Y
SF^1_N GD^1_Y
NO^2
YES^1

Fig. 5 Parallel classification of objects

Final Comments and Future Research

In this paper, we have proposed a modeling of decision trees through P systems. The advantage of using this computation model lies basically in its high parallelism that allows the processing of large volumes of data efficiently. In addition, its implementation with GPU-based technology makes us believe that it is a plausible solution for solving problems that are of great strategic interest in the big data processing framework.

One of the most important aspects when working with decision trees is how to build them. Usually, this task is approached through machine-learning techniques (i.e., [12]). Although this aspect is outside the scope of this paper, we would like to point to a solution based on machine learning of decision trees integrated within the framework of the P systems. Our approach is based on the design of rules for membranes creation and object communication that would be executed according to an information theory criterion, such as the decrease in entropy of the system [13]. This, without a doubt, will be one of the aspects that we will approach in our future works.

The parallel processing version that we have proposed opens up new scenarios to be considered. In this case, we could enable different decision trees that operate simultaneously. The decisions of each tree (in this case, of each P system) would be sent to an additional P system that would evaluate the classification objects and make a final decision, as it is done in ensemble methods based in decision trees. The formalization of this type of systems would be done in a framework of tissue P systems [7] or multi-environment P systems [2]. Actually, this is part of our work in progress.

References

1. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Chapman & Hall, Boca Raton (1984)
2. Cardona, M., Colomer, M.A., Margalida, A., Palau, A., Pérez-Hurtado, I., Pérez-Jiménez, M.J., Sanuy, D.: A computational modeling for real ecosystems based on P systems. Nat. Comput. **10**(1), 39–53 (2011)
3. Cecilia, J.M., García, J.M., Guerrero, G.D., Martínez-del-Amor, M.A., Pérez-Hurtado, I., Pérez-Jiménez, M.J.: Simulation of P systems with active membranes on CUDA. Brief. Bioinform. **11**(3), 313–322 (2010)
4. Díaz-Pernil, D., Peña-Cantillana, F., Gutiérrez-Naranjo, M.A.: Self-constructing Recognizer P Systems. In: Proceedings of the Thirteenth Brainstorming Week on Membrane Computing. Fénix Editora, pp. 137–154 (2014)
5. Fayyad, U.M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. Mach. Learn. **8**, 87–102 (1992)
6. Kingsford, C., Salzberg, S.L.: What are decision trees ? Nat. Biotechnol. **26**(9), 1011–1013 (2008)
7. Martín-Vide, C., Păun, Gh, Pazos, J., Rodríguez-Patón, A.: Tissue P systems. Theor. Comput. Sci. **296**, 295–326 (2003)
8. Martínez-del-Amor, M.A., García-Quismondo, M., Macías-Ramos, L.F., Valencia-Cabrera, L., Riscos-Núñez, A., Pérez-Jiménez, M.J.: Simulating P systems on GPU devices: a survey. Fund. Inf. **136**(3), 269–284 (2015)
9. Mitchell, T.: Machine Learning. McGraw-Hill, New York City (1997)
10. Păun, Gh: Membrane Computing, An Introduction. Springer, Berlin (2002)
11. Păun, Gh, Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford (2010)
12. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, Burlington (1993)
13. Sempere, J.M.: A View of P systems from information theory. In: Proceedings of the 17th international conference on membrane computing (CMC 2016) LNCS vol. 10105. Springer, pp. 352–362 (2017)
14. Sammut, C., Webb, G.I. (eds.): Encyclopedia of Machine Learning. Springer, Berlin (2011)
15. Wang, J., Hu, J., Peng, H., Pérez-Jiménez, M.J., Riscos-Núñez, A.: Decision tree models induced by membrane systems. Rom. J. Inf. Sci. Technol. **18**(3), 228–239 (2015)
16. Zhang, C., Ma, Y. (eds.): Ensemble Machine Learning, Methods and Applications. Springer, Berlin (2012)
17. Zhang, X., Wang, B., Ding, Z., Tang, J., He, J.: Implementation of membrane algorithms on GPU. J. Appl. Math. **2014**, 7 (2014)

José M. Sempere is an Associate Professor at the Universitat Politècnica de València (UPV), where he teaches and makes scientific research since 1989. He holds a Doctorate in Computer Science from the same university. His areas of interest include membrane computing, biomolecular computing, machine learning, and bioinformatics. He has published numerous international works on the subjects referred to above and he is a member of several scientific organizations such as the European Association for Theoretical Computer Science (EATCS) and the ACM Special Interest Group on Bioinformatics, Computational Biology, and Biomedical Informatics (SIGBio). He currently coordinates the Spanish Thematic Network in Biomolecular and Biocellular Computing (REDBIOCOM) and the research group in formal language theory, computability, and cryptography at UPV.