# Optimizing Membrane System Implementation with Multisets and Evolution Rules Compression

Abraham Gutiérrez, Luis Fernández, Fernando Arroyo, Ginés Bravo

Natural Computing Group
Universidad Politécnica de Madrid, Spain
{abraham, setillo, farroyo, gines}@eui.upm.es

**Summary.** Nowadays, several research works on implementation of P systems have been focused on the massively parallel character of the model. In particular, it is possible to find out implementation of P system on cluster of processors, networks of processors, FPGA's ad hoc designed, and microcontrollers. Several published time analysis have proved that there is a very strong relationship between communication and evolution rules application time in membranes of the system. This relation determines the number of membranes that can be allocated per processor in order to obtain the minimum evolution time for the P system. This fact presents a problem related to hardware technologies which have a small amount of memory for storing multisets of objects and evolution rules, in particular microcontrollers.

The aim of this work is to present an algorithm for compressing information of multisets and evolution rules stored in membranes. In particular, this algorithm has to compress information, without penalizing evolution rules application and communication times with complex processes for compressing and decompressing such information. At the same time, keeping the same parallelism degree obtained in P systems implementation over distributed architectures presented in previous research works, but storing more membranes per processor.

## 1 Introduction

Membrane computing is a new computational model based on the membrane structure of living cells [14]. It must be stressed that they are not intended to model the functioning of biological membranes. Rather, they explore the computational character of several membranes features that can be used for modelling new computational paradigms inspired in Nature. This model has become, during last years, a powerful framework to develop new ideas in theoretical computation and to connect Biology with Computer Science.

The membrane structure of a P System is a hierarchical arrangement of membranes, embedded in a skin that separates the system from its environment. A membrane with no other membrane inside is called elementary. Each membrane defines a region that contains a multiset of objects, and a set of evolution rules.

The objects are represented by symbols from a given alphabet. The objects can pass through membranes and membranes can change their permeability, they can be dissolved, or they can be divided. These features are used in defining transitions between system configurations, and sequences of transitions are used to define computations. Membrane systems are synchronous, in the sense that a global clock is assumed, i.e., the same clock holds for all regions of the system. At each time unit, a transformation of a system configuration takes place by applying rules in all regions, in a nondeterministic and maximally parallel manner.

Nowadays, membrane systems have been sufficiently characterized from a theoretical point of view. Their computational power has been settled – many variants are computationally complete. Among their most relevant characteristics appears the fact that they can solve non polynomial time problems in polynomial time, but this is achieved by the consumption of an exponential number of resources, in particular, the number of membranes that evolve in parallel.

An overview of membrane computing software can be found in [2] [15]. However, the way in which these models can be implemented is a persistent problem today, because "the next generation of simulators may be oriented to solve (at least partially) the problems of information storage and massive parallelism by using parallel language programming or by using multiprocessor computers" [2]. In this sense, information storage in membrane computation implementation is an example of Parkinson's First Law [13]: "storage and transmission requirements grow double than storage and transmission improvements".

This work focuses upon the problem due to the storage of multisets and rules information for the membranes in P-systems implementation. In particular, we pretend to get the highest compression level for the information without penalizing compression and decompression time with cost-consuming operations. Thus, we intend to reach a compression ratio that complies with the parallelism level reached in previous research works for P-systems implementation.

The structure of this work is as follows: first, related works in P-systems implementation are presented; next two sections present requirements for information compression in membrane systems and the proposed compression schema; then we analyze the obtained results for a set of tests for a well known P-system. Finally, authors present reached conclusions.

## 2 Related Works

First works over massively parallel implementation for P-systems started with Syropoulos [19] and Ciobanu [3] that in their distributed implementations of P systems use Java Remote Method Invocation (RMI) and the Message Passing Interface (MPI) respectively, on a cluster of PC connected by Ethernet. These authors do not carry out a detailed analysis about the importance of the time used during communication phase in the total time of P system evolution; although Ciobanu stated that "the response time of the program has been acceptable. There

are however executions that could take a rather long time due to unexpected network congestion".

Recently, in [20] and [1] they present analysis for distributed architectures that are technology independent, based on: the allocation of several membranes in the same processor; the use of proxies for communication among processors; and, token passing in the communication. These solutions avoid communication collisions, and reduce the number and size of communication among membranes. All this allows to obtain a better step evolution time than in others suggested architectures congested quickly by the network collisions when the number of membranes grows up. Table 1 summarizes minimum times ($T_{min}$), optimal amount of processors needed to implement the P system $P_{opt}$, membranes allocated at each one of them $K_{opt}$ to reach those minimum times, the throughput obtained with corresponding processors $Th_{proc}$ and communications $Th_{com}$ for the architecture. These analysis consider the P-system number of membranes ($M$) that would evolve, the maximum time used by the slowest membrane in applying its rules ($T_{apl}$), and the maximum time used by the slowest membrane for communication ($T_{com}$).

| Distributed Architecture [20] | Distributed Architecture [1] |
|---|---|
| $T_{\min} = 2\sqrt{2\ M\ T_{apl}\ T_{com}} - 2\,T_{com}$ | $T_{\min} = 2\ \sqrt{M\ T_{apl}\ T_{com}} + T_{com}$ |
| $P_{opt} = \sqrt{\frac{M\ T_{apl}}{2\ T_{com}}}$ | $P_{opt} = \sqrt{\frac{M\ T_{apl}}{T_{com}}}$ |
| $K_{opt} = \sqrt{\frac{2\ M\ T_{com}}{T_{apl}}}$ | $K_{opt} = \sqrt{\frac{M\ T_{com}}{T_{apl}}}$ |
| $Th_{proc} \sim 50\%$ | $Th_{proc} \sim 50\%$ |
| $Th_{com} \sim 50\%$ | $Th_{com} \sim 100\%$ |

**Table 1.** Distributed architecture parameters depending on application rules time ($T_{apl}$), communication time ($T_{com}$) and number of membranes ($M$)

It may be concluded that to reach minimum times over distributed architectures, there should be a balance between the time dedicated to evolution rules application and the time used for communication among membranes. So, depending from the existing relation between both times, and from the number of membranes in the P-system, it is possible to determine the number of processors and the number of membranes that can be allocated at each one of them to obtain the evolution minimum time.

The difference between both architectures lies on the different topology for the processors net and the policy for token passing. Thus, [1] reaches a throughput near to a 100% of the communication line. With a 40% of increment in the parallelism level produced by the increment of the number of processors in the architecture, which permit to obtain a reduction over the 70% in the evolution time. Both works conclude that, for a specific number of membranes $M$, if it is possible that:

1. $T_{apl}$ be N faster, the number of membranes that would be hosted in a processor would be multiplied by $\sqrt{N}$, the number of required processors would be

divided by the same factor and the time required to perform an evolution step would improve approximately with the same factor $\sqrt{N}$.
2. $T_{com}$ be N faster, the number of required processors would be multiplied by $\sqrt{N}$, the number of membranes that would be hosted in a processor would be divided by the same factor and the time required to perform an evolution step would improve approximately by the same factor $\sqrt{N}$.

Table 2 summarizes the importance of reducing $T_{apl}$ and $T_{com}$ over the distributed architectures parameters (minimum evolution time, optimum number of processors and optimum number of membranes per processor).

| Conditions | $T_{\min}$ | $P_{opt}$ | $K_{opt}$ |
|---|---|---|---|
| $T_{apl}$ be N faster and $T_{com}$ be equal | $\frac{T_{\min}}{\sqrt{N}}$ | $\frac{P_{opt}}{\sqrt{N}}$ | $K_{opt} \cdot \sqrt{N}$ |
| $T_{apl}$ be equal and $T_{com}$ be N' faster | $\frac{T_{\min}}{\sqrt{N'}}$ | $P_{opt} \cdot \sqrt{N'}$ | $\frac{K_{opt}}{\sqrt{N'}}$ |
| $T_{apl}$ be N faster and $T_{com}$ be N' faster | $\frac{T_{\min}}{\sqrt{N'} \cdot \sqrt{N}}$ | $\frac{P_{opt} \cdot \sqrt{N'}}{\sqrt{N}}$ | $\frac{P_{opt} \cdot \sqrt{N}}{\sqrt{N'}}$ |

**Table 2.** Repercussion on distributed architecture parameters depending on $T_{apl}$ and $T_{com}$

All the previous analysis for distributed architectures is independent of specific technology. In this sense, as it usually happens, implementation of these systems has been addressed from two different approaches: software and hardware models. The main research lines are: simulation over local networks using *cluster of microprocessors*, hardware *ad-hoc*, and generic hardware such as *microcontrollers*:

1. The hardware specifically designed has the possible advantage of being a massively parallel solution [17] [5] [12]. Their weak point resides in the lack of flexibility that presents, because this type of solutions only allows the simulation of a specific kind of membrane systems (for each membrane system a specific hardware is needed). They are also very enclosed solutions because they may be applied only to a very little range of problems (reduced number of objects in the alphabet and small number of evolution rules).
2. The solutions based on cluster of microprocessors and local networks have as main advantage the use of very common and well-known architectures. They are floppy systems also because a change at software level allows the simulation of any kind of membrane systems. Its main problem is that the best simulation times are reached always with few units, so the obtained solutions have a low degree of parallelism.
3. The solution, based on the use of microcontrollers [9] [10], seeks to be an intermediate point between hardware specifically designed and simulation using cluster of microprocessors. Microcontroller architecture is as flexible as a cluster of microprocessors and less enclosed and floppier than the hardware specifically designed. Microcontroller architecture has more level of potential parallelism than cluster of microprocessors but does not have intrinsic parallel

nature of the hardware specifically designed. Therefore, it is the cheapest architecture. But, in return, in [10] it is admitted that it is needed "the definition of more and more efficient data structures and algorithms oriented to minimize the quantity of memory required for the multisets and evolution rules storage that define the membrane".

Thus, in hardware solutions and solutions based upon microprocessors nets, the amount of information that has to be stored and transmitted is very important. In the first case, the main problem is due to their low storage capacity. So, reducing this amount of information needed to represent a membrane, means to be able to extend the variety of problems that can be solved with this technology. In the second case, reducing the amount of information to transmit means to minimize the bottleneck in processor communication and so, increase the parallelism level.

## 3 Compression Requirements

First, unlike other environments, where it is admissible a lossy information system (i.e. multimedia contents transmission), in our environment it is essential to have a lossless information compression system.

Almost all the compression methods require two phases: the first for analysis followed by a second one for conversion. First, an initial analysis of the information is done to identify repeated strings. From this analysis, an equivalences table is created to assign short codes to those strings. In a second phase, information is transformed using equivalent codes for repeated strings. Besides, this table is required with the information for its future compression/decompression. On the other hand, we must realize that a higher compression without any information loss will take more processing time. *Bitrate* is always variable and it is used mainly in text compression [18]. Because all of this, in spite of the fact that there are compression systems that are able to reach entropy limit - highest limit for data compression (i.e. Huffman codes) - they are not the ideal candidates for our system because of the following reasons:

1. Table storage will increase the needs for memory resources and would decrease compression goal.
2. Time for the phase of evolution rules application is penalized with compression/decompression processes when accessing compressed information on the P-system. This reduces parallelism level from distributed systems and increases evolution time.
3. And also, despite of the fact that communication phase time will be reduced because a lowest amount of information is transmitted, this will be counteracted by the time needed for decompression in the destination.

This way, as it is said in the goal of this work, compression schema for information from P-system membranes should accomplish following requirements:

1. there should be no information loss;
2. it should use the lowest amount of space for storage and transmission;
3. it should not penalize time for rules application phase and communication among membranes while processing compressed information. Thus, this means that the system should:
   a) encode information for a direct manipulation in both phases without having to use coding/decoding processes.
   b) do the compression in a previous stage to the P-system evolution
   c) therefore, abandon entropy limit to be able to maintain parallelism level and evolution time reached in previous research works.

## 4 Compression Schema

This work pretends to compress the information from multisets that are present in regions and rules antecedents and consequents from each one of the P-system membranes. But it does not address the compression of another kind of information, such as priorities, membrane targets in rule consequents nor dissolving rule capability.

Representation for multisets information in related literature is Parikh's vector [4]. Data compression is very associated with its representation. Proposed compression schema is presented here in three consecutive steps beginning with Parikh's vector codification over the P-system alphabet. To show it, each of the successive steps will be applied over the following P-System [14]. See figure 1.

$\Pi_1 = (V, \mu, w_1, \ldots, w_4, (R_1, \rho_1), \ldots, (R_4, \rho_4), 4),$
$V = \{a, b, b', c, f\},$
$\mu = [_1[_2[_3\,]_3[_4\,]_4]_2]_1,$
$w_1 = \lambda,\ R_1 = \emptyset,\ \rho_1 = \emptyset,$
$w_2 = \lambda,\ R_2 = \{b' \to b,\ b \to b(c, in_4),\ r_1 : ff \to af,\ r_2 : f \to a\delta\},$
$\rho_2 = \{r_1 > r_2\},$
$w_3 = af,\ R_3 = \{a \to ab',\ a \to b'\delta,\ f \to ff\},\ \rho_3 = \emptyset,$
$w_4 = \emptyset,\ R_4 = \emptyset,\ \rho_4 = \emptyset.$

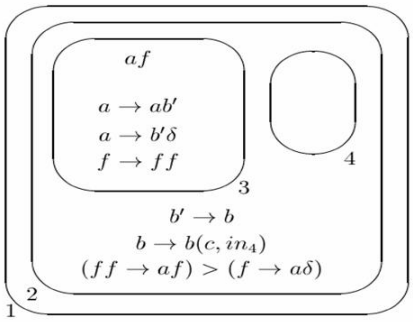


**Fig. 1.** A P System generating $n^2, n \geq 1$

### 4.1 Parikh's Vector over P System Alphabet

Each region of a membrane can potentially host an unlimited number of objects, represented by the symbols from a given alphabet $V$. We use $V^*$ to denote the

set of all strings over the alphabet $V$ (we consider only finite alphabets). For $a \in V$ and $x \in V^*$ we denote by $|x|_a$ the number of occurrences of $a$ in $x$. Then, for $V = \{a_1, ..., a_n\}$, the Parikh vector associated with $V$ is the mapping on $V^*$ denoted by $\psi_V(x) = (|x|_{a_i}, \cdots, |x|_{a_n})$ for each $x \in V^*$. Our representation considers an order in the set of objects, hence the byte's order reflects the order of the objects within the alphabet and consequently, the position directly indicates which symbol's multiplicity is being stored.

Figure 2 shows an example for Parikh's vector associated to a multiset over the alphabet $V = \{a, b, c, d, e, f\}$.

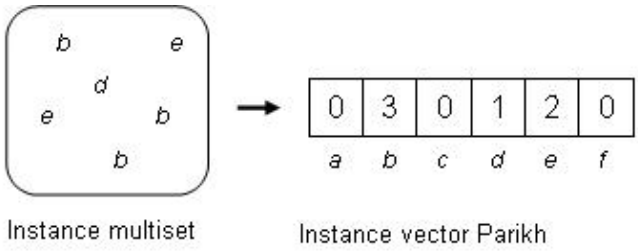


**Fig. 2.** Example for Parikh's vector over the alphabet $V = \{a, b, c, d, e, f\}$

Figure 3 shows previous P-system information by the use of Parikh's vector over the alphabet $V = \{a, b, b', c, f\}$ for all the multisets that are present in each region and evolution rules for each membrane.

As we can see, codification using Parikh's vector over the P system alphabet requires 90 storage units for the multiplicities present in the multisets

### 4.2 Parikh's Vector for each Membrane's Alphabet

First step in compression considers only the alphabet subset for the P system that may exist in each of the regions for the membrane system, whatever are the possible configurations for the P system evolution. This subset may be calculated by a static analysis, previous to P system evolution time. In order to determine membranes alphabets in a given P system it is needed to consider the following facts:

1. Every object present at the region in the P system initial configuration belongs to its membrane's alphabet.
2. Every object present at the consequent in a membrane's evolution rule with target "here" belongs to its membrane's alphabet.
3. Every object present at the consequent in a membrane's evolution rule with target "in" to another membrane, belongs to the target membrane's alphabet.
4. Every object present at the consequent in a membrane's evolution rule with target "out", belongs to it's father alphabet.

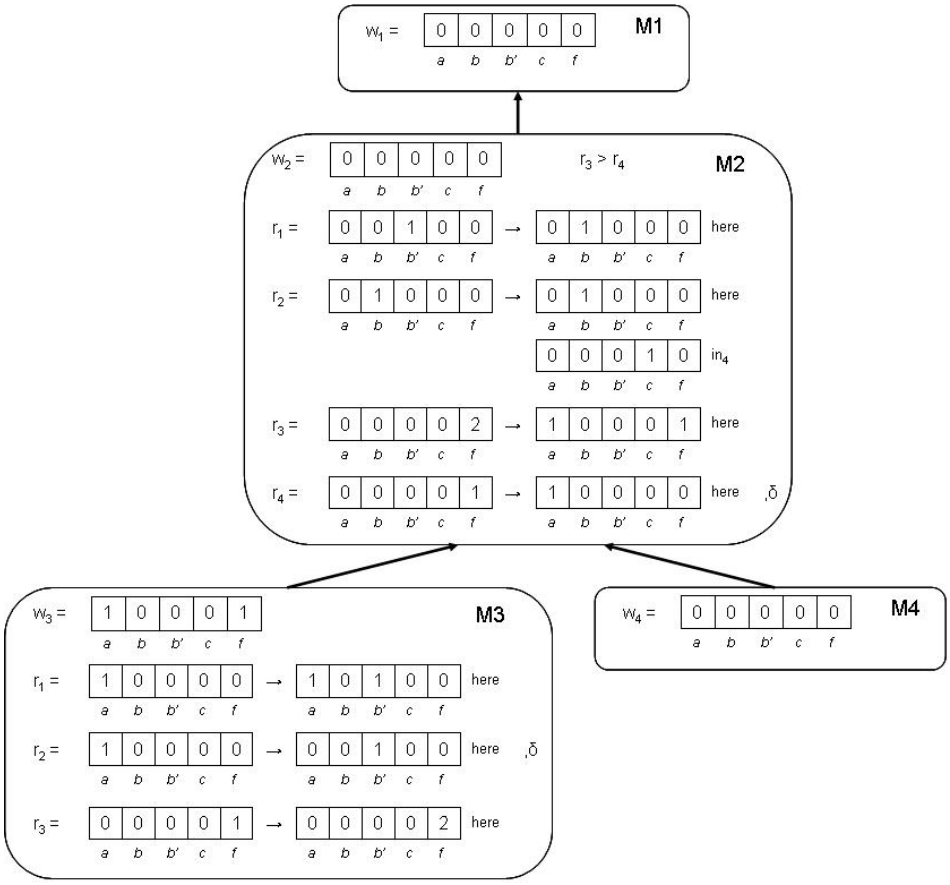


**Fig. 3.** A P System generating $n^2, n \geq 1$ representing multiset and evolution rules with Parikh's vector associated with V alphabet

5. Every object present at any membrane's alphabet with a evolution rule with dissolution capability belongs to it's father alphabet.

We will designate alphabet for a membrane $i$, $V_i \subseteq V$, where $\forall\ V_i\ ,\ , \ |V_i| \leq |V|$. For the example in figure 1, alphabets for the four membranes are: $V_1 = \{a, b, b', f\}$; $V_2 = \{a, b, b', f\}$; $V_3 = \{a, b, f\}$; and $V_4 = \{c\}$. According to these alphabets, each of the multisets in the region and the antecedents for a membrane evolution rules are codified by the Parikh's vector over its membrane alphabet. On the other hand, consequent multisets in each evolution rule are codified by the Parikh's vector over the target membrane alphabet.

Figure 4 shows previous P System information using Parikh's vector for each membrane alphabet.

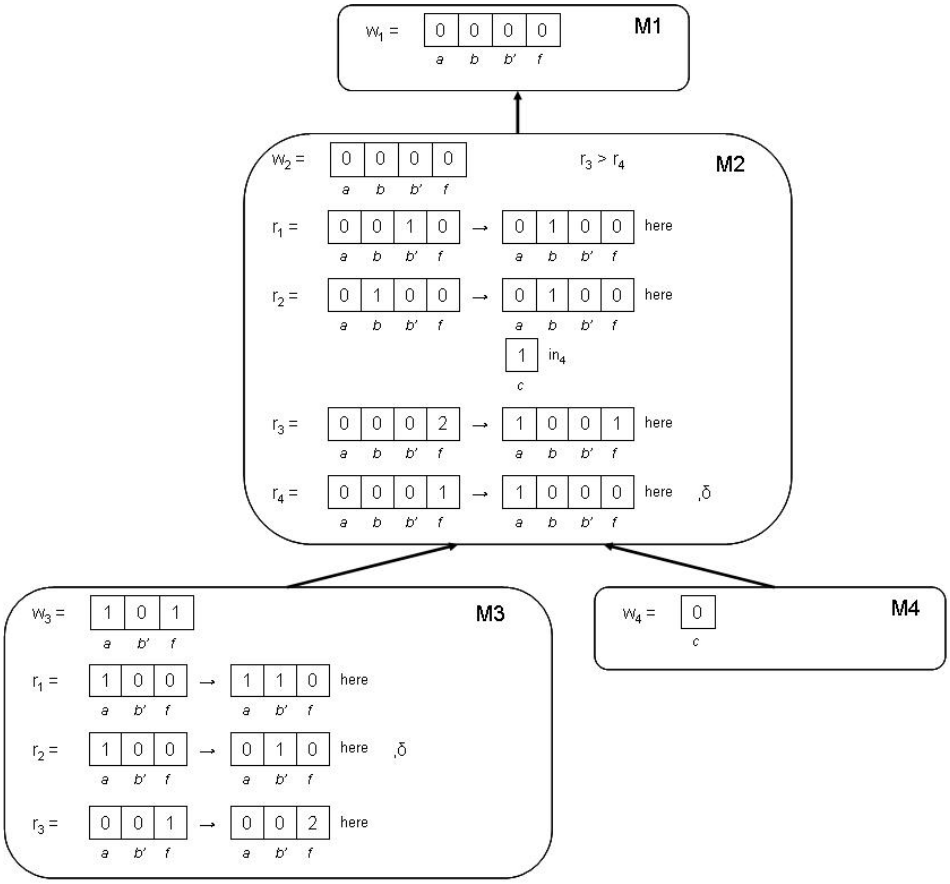


**Fig. 4.** A P System generating $n^2, n \geq 1$ representing multiset and evolution rules with Parikh's vector associated with the membranes' alphabet

Now, P system codification by the use of Parikh's vector over each membrane specific alphabet, requires 63 storage units for present multiplicities in multisets. This codification reduces information size over a 70,0% concerning previous codification.

### 4.3 Parikh's vector without null values

Next compression step is an alteration over the *Run Length Encoding (RLE)* algorithm [11], used mainly to compress FAX transmissions. In this lossless codification, data sequences with same value (usually zeros) are stored as a unique value plus its count. RLE compression factor is, approximately:

$$\frac{E\,(X)}{E\,\{\log_2 x\}}$$

where $X$ is a discrete random variable that represents the number of successive zeros between two ones and $E(X)$ is its expected value (average). Compression value stands between 20% and 30%.

In our case, what we pretend is to eliminate all the null values in Parikh's vector, that is, to eliminate all the references to the alphabet elements in a membrane that do not appear in its multiset. This information may be considered as redundant cause it may be obtained from the new coded information. In a formal way, let $V = \{a_1, a_2, \ldots, a_n\}$ be an ordered finite alphabet, $\forall x \in V*$, the encoded Parikh vector associated with $V$ is defined by $\Psi_V^E(x) = \{(|x|_{a_i}, i) \quad | \quad |x|_{a_i} \neq 0\}$. Figure 5 shows an example for Parikh's vector without null values associated to a multiset over the ordered alphabet $V = \{a, b, c, d, e, f\}$.

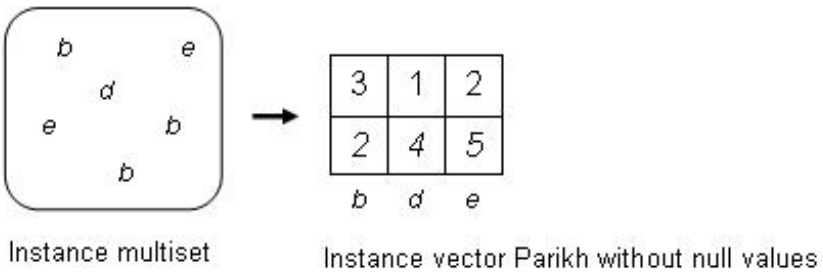


**Fig. 5.** Example for Parikh's vector without null values over the alphabet $V = \{a, b, c, d, e, f\}$

Concerning to multiplicities of objects present in multisets, there is two different situations: for the cases with multisets present at a membrane region, independently from the initial configuration, its multiplicities values are variable depending on the evolution that takes the membrane system in a non deterministic way; on the other hand, for the cases with multisets present at the evolution rules antecedents and consequents, its multiplicities values are constant and known previously to the P system evolution.

According to this situation, the evolution second step encodes without null values just the information that belongs to constant multisets present at evolution rules. Thus, we get a more compressed and lossless representation. The reason that does this representation possible is the fact that the absence of these null values multiplicities does not affect none of the multisets operations (addition, subtraction, applicability, scalar product, ...)

Figure 6 shows previous P system information using Parikh's vector without null values over each membrane's ordered alphabet.

Now, the P system codification using Parikh's vector without null values over each membrane's ordered alphabet requires 46 storage units for the multiplicities

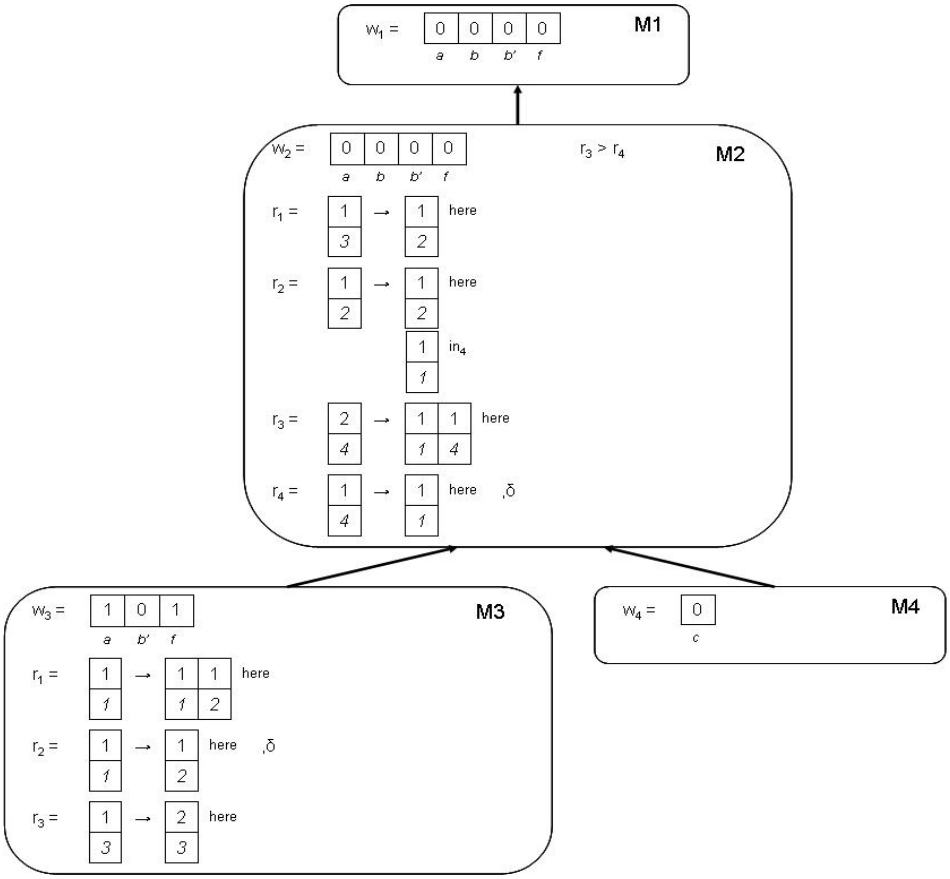


**Fig. 6.** A P System generating $n^2, n \geq 1$ representing multiset and evolution rules with Parikh's vector associated with membranes alphabet and without null values

present at the multisets. This codification, in figure's 1 example, reduces information size until a 51,1% from the initial codification.

### 4.4 Storage Unit Compression

Last compression step concerns storage unit size for each of the P system information values. Depending on the storage unit size (measured in bits), we will be able to codify a greater or smaller range of values. In membrane computing, that does not allow negative values, given a size $t$ bits for the storage unit, the range for possible values will vary from 0 to $2^t - 1$.

In this section, we will have to take into account separately multisets present at the membrane's regions in front of the ones present at evolution rules. For the first, storage unit size depends on the value range we want to reach during

evolution while not having an overflow. Instead of it, for the second ones, we have to take into account, as it was shown in previous sections, that each membrane's ordered alphabet and their multiplicities are constant. Thus, an analysis previous to the P system evolution allows calculating the value ranges that are present in constant multisets for evolution rules and, so, the size that is needed to get their codification:

1. value range for multiplicities present at the antecedents and consecuent for each membrane (from 0 to 2 in example in figure 1; 2 bits needed for its codification).
2. value range for Parikh's vector positions over the ordered alphabet for each membrane (from 1 to 4 in example in figure 1; 2 bits needed for its codification).

Table 3 presents the size, in bits, needed to represent P system information from figure 1, with different storage unit sizes, using two different representations: the original representation (Parikh's vector over P system alphabet) and our compression schema (Parikh's vector without null values over membrane's alphabet). Last row shows compression rates obtained for different storage units sizes. In particular, it has been considered that the minimum storage unit size should be 8 bits according to actual technologies. In this work, we do not address the possibility of encoding several values at the same byte, which would increase more compression rates.

| | *Storage unit size* | | | |
|---|---|---|---|---|
| *Representational Schema* | *8 bits* | *16 bits* | *32 bits* | *64 bits* |
| *Parikh's vector for the P system alphabet* | 720 bits | 1440 bits | 2880 bits | 5760 bits |
| *Parikh's vector whithout null values associated with membrane alphabets* | 368 bits | 464 bits | 656 bits | 1040 bits |
| ***Compression degree*** | **51.1%** | **32.2%** | **22.8%** | **18.1%** |

**Table 3.** Size in bits for representing P system information with different size for storage units and different representations

## 5 Analysis of Results

At this section we present the analysis of results obtained from the compression schema. First we analyze the schema compression itself. Afterwards we analyze the impact that compression has over the times for evolution rules application and communication among membranes. Finally, we analyze the global impact over distributed architectures parameters: evolution minimum time, optimum number of processors and membranes in each processor.

For the analysis of the following sections, we will do a review over a test set composed by the P System published in [14] and [16]. Table 4 describes the considered P System set.

| *P System* | *Task* | *Reference* |
|---|---|---|
| A. | First example | [14] |
| B. | Decidability: n mod k = 0 | [14] |
| C. | Generating: $n^2, n \geq 1 (1^{st}$ version) | [14] |
| D. | Generating: $n^2, n \geq 1$ ($2^{nd}$ version) | [16] |

**Table 4.** P System used for testing

### 5.1 Compression Schema Analysis

Table 5 shows compression rates reached for each P system from table 4, considering different storage unit sizes. Last row presents average compression rates for each storage size.

| | *Storage unit size* | | | |
|---|---|---|---|---|
| *P System* | *8 bits* | *16 bits* | *32 bits* | *64 bits* |
| A. | 59.8% | 37.8% | 26.8% | 21.3% |
| B. | 75.0% | 47.7% | 34.1% | 27.3% |
| C. | 51.1% | 32.1% | 22.8% | 18.1% |
| D. | 52.2% | 33.3% | 23.9% | 19.2% |
| ***Average compression degree*** | **59.5%** | **37.7%** | **26.9%** | **21.5%** |

**Table 5.** Compression degree for P System from Table 4

Considering the worst case for this compression schema (8 bits for all the storage units), at least, we reach a compression rate of 75,0%, which implies a increase of a 33,3% for memory availability to store information. For average compression rate (59,5%), it is reached an increase of 68,0% of memory availability. So we attenuate the storage problem for information in distributed architectures implemented with low storage capacity microcontrollers based technologies. Using this compression schema, it will be possible allocate more membranes in each microcontroller and so, it will be possible to reach minimum times at the same time that we are maximizing resources.

On the other hand, it has to be underlined that the compression process is done by an previous analysis to the P system evolution. Thus, evolution rules application and communication among membranes phases are not penalized with compression/decompression processes.

### 5.2 Impact Analysis for Evolution Rules Application Time

Published parallel and sequential algorithms [6] [7] [21] [22] [8] for evolution rules application are based upon a limited set of multisets primitive operations. These are calculation of: applicability, maximum applicability, antecedent/consequent addition and subtraction over its region multiset and the scalar product of a antecedent/consequent.

Algorithmic complexity of any of these operations is determined by the representation of multiset information present at the evolution rules. In worst case, using representation trough Parikh's vector over the P system alphabet, complexity will be equal to its alphabet cardinal. On the other hand, using representation through the proposed compression schema, complexity in worst case will be equal to the multiset support that is present at the evolution rule antecedent/consequent. Table 6 presents, for each of the P system in table 4, its alphabet support, the average support for multisets present in its evolution rules and a percentage based relation among both cardinals. Last row presents these cardinals average values and their relation.

| *P System* | $\mid V \mid$ | \| support(w) \| | *%* |
|---|---|---|---|
| A. | 4 | 1.05 | 26.3% |
| B. | 4 | 1.50 | 37.5% |
| C. | 5 | 1.13 | 22.6% |
| D. | 5 | 1.13 | 22.6% |
| ***Average*** | **4.5** | **1.20** | **26.7%** |

**Table 6.** Alphabet Cardinality and support average from P systems of Table 4

According to these empirical values, each of the primitive operations previously mentioned will decrease its execution time approximately until a 26,7%. And consequently, evolution rules application time will be approximately 3,75 times faster.

### 5.3 Impact Analysis for Communication among Membranes Time

Communication among membranes addresses submission of multisets present at the applied application rules consequents and, in case of dissolution, the region multiset itself. Depending on information representation, the data packet size to transmit will be smaller or bigger. Table 7 shows, for each of the P systems shown in table 4, information compression rate for its communications for different storage units sizes. Last row presents compression rates average.

According to these empirical values, a reduction until a 55,6% of the information to transmit among membranes may be reached in the worst case. Considering that communication is a linear process that depends upon the amount of information to transmit, communication time among membranes will be approximately 1,80 times faster.

| | *Storage unit size* | | | |
|---|---|---|---|---|
| *P System* | *8 bits* | *16 bits* | *32 bits* | *64 bits* |
| A. | 55.0% | 45.0% | 40.0% | 37.5% |
| B. | 60.0% | 50.0% | 45.0% | 42.5% |
| C. | 54.0% | 44.0% | 39.0% | 36.5% |
| D. | 53.3% | 44.4% | 40.0% | 37.8% |
| ***Average*** | **55.6%** | **45.9%** | **41.0%** | **38.6%** |

**Table 7.** Compression degree for communication units from P systems of Table 4

### 5.4 Global Impact Analysis

Finally, we present a impact analysis of this compression schema over distributed architecture parameters. In particular, we examine, following criteria shown in table 2, implication in optimum number of processors and membranes per processor and minimum evolution time.

On one side, time reduction for evolution rules application increases the number of membranes per processor. It also decreases the number of processors and evolution time. According to the previous empirical data, from a rule application time 3,75 times faster, we get an increment of a 93,5% for membranes per processor, a reduction until a 51,6% for number of processors and for evolution time.

On the other hand, time reduction for communication among membranes increases the number of processors. It also decreases the number of membranes per processor and evolution time. According to the previous empirical data, from a communication time 1,80 times faster, we get, for the worst case, a 34,2% increment for number of processors and a reduction until a 74,5% for the number of membranes per processor and for evolution time.

Taking in account both factors, reduction for application and communication time, counteract their effects over the number of processors and the number of membranes per processor. According to the previous empirical data, we get a reduction of a 69,3% for the number of processors, an increment of a 44,3% for the number of membranes per processor and a reduction until a 38,5% for evolution time.

## 6 Conclusions

In this work has been presented a schema for compressing multisets and evolution rules for P system membranes. This compressing schema is a variant from Run Length Encoding starting form the Parikh's vector considering the specific membrane alphabet. Empirical results over a reduced set of classical P systems show degrees of compression varying from 51.1 % to 18.1% depending on the size in bits needed for storing objects multiplicities for multisisets of objects in membranes of the P systems. This is a way for allocating more membranes per processor in

the implementation of P system on distributed architectures having a low memory capability.

On the other hand, the compression schema does not penalize evolution rule application nor communication times during P system evolution. The schema does not required compression/decompression process during P system evolution. The whole compression process is performed by mean of a static analysis previous to the P system evolution. These facts, thanks to the representation of information established, improve the system performance reducing evolution rule application and communication times, what is very important because it implies a direct implication on reducing the evolution time of the membrane systems.

## References

1. G. Bravo, L. Fernández, F. Arroyo, J. Tejedor, *Master-Slave Parallel Architecture for Implementing P Systems*. The 8th WSEAS International Conference on Mathematics and Computers in Business and Economics (MCBE'07). Vancouver (Canada) June, 2007. (accepted).
2. G. Ciobanu, M. Pérez-Jiménez, Gh. Păun, eds, *Applications of Membrane Computing*, Natural Computing Series, Springer Verlag, October, 2006.
3. G. Ciobanu, G. Wenyuan, *A P System running on a Cluster of Computers*, Membrane Computing. International Workshop, WMC2003, Tarragona, Spain. Lecture Notes in Computer Science, 2933, Springer, Berlin, 2004, 123-150.
4. J. Dassow, *Parikh Mapping and Iteration*. Lecture Notes In Computer Science (Vol. 2235) ISBN:3-540-43063-6. Proceedings of the Workshop on Multiset Processing: Multiset Processing, Mathematical, Computer Science, and Molecular Computing Points of View, 2000, 85 - 102.
5. L. Fernández, V.J. Martínez, F. Arroyo, L.F. Mingo, *A Hardware Circuit for Selecting Active Rules in Transition P Systems*, Workshop on Theory and Applications of P Systems. Timisoara (Romania), September, 2005, 45 - 48.
6. L. Fernández, F. Arroyo, J. Castellanos, J.A. Tejedor, I. García, *New Algorithms for Application of Evolution Rules based on Applicability Benchmarks*. International Conference on Bioinformatics and Computational Biology(BIOCOMP06), Las Vegas (EEUU), July, 2006, 94 - 100.
7. L. Fernández, F. Arroyo, J.A. Tejedor, J. Castellanos, *Massively Parallel Algorithm for Evolution Rules Application in Transition P Systems*. Preproceedings of Membrane Computing, International Workshop (WMC7), Leiden (The Netherlands) July, 2006, 337-343.
8. F.J. Gil, L. Fernández, F. Arroyo, J.A. Tejedor. *Delimited Massively Parallel Algorithm based on Rules Elimination for Application of Active Rules in Transition P Systems*. Fifth International Conference Information Research and Applications (i.TECH-2007). Varna (Bulgary) June, 2007. (accepted).
9. A. Gutiérrez, L. Fernández, F. Arroyo, V. Martínez, *Design of a Hardware Architecture based on Microcontrollers for the Implementation of Membrane Systems*, SYNASC 2006, 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing Timisoara, Romania. September 26-29, 2006, 39-42.

10. A. Gutiérrez, L. Fernández, F. Arroyo, S. Alonso, *Hardware and Software Architecture for Implementing Membrane Systems: A case of study to Transition P Systems*, DNA13 2007, 13th International Meeting on DNA Computing Memphis, EEUU. June 4-8, 2007. (accepted).
11. D.A. Lelewer, D.S. Hirschberg, *Data Compression*. ACM Computing, Springer Verlag, New York, USA. Verlag Surveys, ACM CR 8902-0069, 1987.
12. V. Martínez, L. Fernández, F. Arroyo and A. Gutiérrez, *A Hardware Circuit for the Application of Active Rules in a Transition P Systems Region,* Fourth International Conference Information Research and Applications, Bulgaria, Varna June 20-25, 2006, 45 -48.
13. C.N. Parkinson.*Parkinson's Law, or the Pursuit of Progress*, John Murray, 1957.
14. Gh. Păun, *Computing with Membranes*, Journal of Computer and System Sciences, 61 (2000), and Turku Center of Computer Science-TUCS Report n 208, 1998.
15. Gh. Păun, *Membrane Computing. Basic Ideas, Results, Applications*, Pre-Proceedings of First International Workshop on Theory and Application of P Systems, Timisoara, Romania, September 26-27, 2005, 1-8
16. Gh. Păun, G. Rozenberg, *A Guide to Membrane Computing*, Theoretical Computer Science, vol 287, 2000. 73-100.
17. B. Petreska, C. Teuscher, *A Reconfigurable Hardware Membrane System*. Preproceedings of the Workshop on Membrane Computing (A.Alhazov, C.Martn-Vide and Gh.Paun, eds) Tarragona, July 17-22 2003, 343-355.
18. D. Salomon. *Data Compression: The Complete Reference*. Springer. ISBN 0-387-40697-2. LCCN QA76.9 D33S25. 2004.
19. A. Syropoulos, E.G. Mamatas, P.C. Allilomes, K.T. Sotiriades, *A Distributed Simulation of P Systems*, Preproceedings of the Workshop on Membrane Computing; Tarragona, 2003, 455-460
20. J.A. Tejedor, L. Fernández, F. Arroyo, G. Bravo. *An Architecture for Attacking the Bottleneck Communication in P System*, AROB 12th '07, XII International Symposium on Artificial Life and Robotics, Oita, JAPAN, January 25-27, 2007, 500 - 505.
21. J.A. Tejedor, L. Fernández, F. Arroyo, A. Gutiérrez. *Algorithm of Active Rule Elimination for Application of Evolution Rules*. The 8th WSEAS International Conference on Mathematics and Computers in Business and Economics (MCBE'07). Vancouver (Canada) June, 2007. (accepted).
22. J.A. Tejedor, L. Fernández, F. Arroyo, S. Gómez. *Algorithm of Rules Application based on Competitiveness of Evolution Rules.* Eight Workshop on Membrane Computing, Thessaloniki (Greece), June 25-28, 2007. (submitted)