ALGORITHM OF ACTIVE RULES ELIMINATION FOR APPLICATION OF EVOLUTION RULES

JORGE A. TEJEDOR
Universidad Politécnica de Madrid
Natural Computing Group
Crta. de Valencia km. 7 Madrid 28031
SPAIN

LUIS FERNÁNDEZ
Universidad Politécnica de Madrid
Natural Computing Group
Crta. de Valencia km. 7 Madrid 28031
SPAIN

FERNANDO ARROYO
Universidad Politécnica de Madrid
Natural Computing Group
Crta. de Valencia km. 7 Madrid 28031
SPAIN

ABRAHAM GUTIÉRREZ
Universidad Politécnica de Madrid
Natural Computing Group
Crta. de Valencia km. 7 Madrid 28031
SPAIN

Abstract: This paper presents a new evolution rules application algorithm to a multiset of objects to use in the P system implementation in digital devices. In each step of this algorithm two main actions are carried out eliminating, at least, an evolution rule to the set of active rules. Therefore, the number of operations executed is limited and it can be known a priori which is its execution time at worst. This is very important as it allows for determination of the number of membranes to be located in each processor in the distributed implementation architectures of P systems to obtain optimal times with minimal resources. Although the algorithm is sequential, it reaches a certain degree of parallelism due to a rule that can be applied several times in a single step. In addition to this, this algorithm has shown in the experimental tests that the execution times is better than the ones previously published.

Key–Words: Natural Computing, Transition P System, Evolution Rules Application, Algorithm

1 Introduction

Computation with membranes was introduced by Gheorghe Păun in 1998 [**?**] through a definition of transition P systems. This new computational paradigm is based on the observation of biochemical processes. The region defined by a membrane contains chemical elements (multisets) which are subject to chemical reactions (evolution rules) to produce other elements. Transition P systems are hierarchical, as the region defined by a membrane may contain other membranes. Multisets generated by evolution rules can be moved towards adjacent membranes (parent and children). This multiset transfer feeds back into the system so that new products are consumed by further chemical reactions in the membranes.

These systems perform computations through transition between two consecutive configurations. Each transition or evolution step goes through two sequential steps: application of rules and communication. First, the evolution rules are applied simultaneously to the multiset in each membrane. This process is performed by all membranes at the same time. Then, also simultaneously, all membranes communicate with their destinations, the multisets generated.

The objective of this paper is to present a new sequential algorithm for rules application to a multiset to obtain a certain degree of parallelism. This algorithm improves upon those published previously in two ways: it is faster and it allows for prior determination of the time needed to execute it.

The paper is structured as follows: first, related works are presented; then, the problem is discussed formally; next, the algorithm is presented, its accuracy shown and its complexity calculated. Finally, experimental data is presented, followed by the conclusions.

2 Related works

At present, three distinct lines of P system implementation are being followed in electronic devices (Personal Computers [**?**], microcontrollers [**?**] and hardware chips [**?**]). None of these has managed to achieve broadly parallel execution, either in applying evolution rules or in communication. Implementation with

Personal Conmputers and microcontrollers face the problem that these devices are sequential by nature. Implementation in Hardware is the closest to maximal parallelism, although it has not yet achieved it. Moreover, it is not easily scalable.

In literature on P systems, we find that Ciobanu [**?**] proposes an architecture in which rules application achieves a certain level of parallelism while communications are sequential. Ciobanu himself has acknowledged that the problem of this architecture lies in network congestion: "There are, however, executions that could take a rather a long time due to unexpected network congestion".

The SW architecture for P Systems implementation proposed in [**?**] eradicates network congestion and obtains an evolution step time proportional to the square root of the number of membranes. This solution requires renouncing the maximal parallelism in favor of achieving a degree of parallelism that would be dependent on both the speed of communications and the quicker application of evolution rules. Thus, quicker algorithms must be developed to adapt to both sequential and parallel technologies.

Moreover, architecture [**?**] must know, a priori, the execution time of evolution rules application algorithm in order to make a balanced distribution of membranes in system processors. This information cannot be obtained with algorithms published to date [**?**][**?**][**?**] because their computational cost depends on the multiset cardinals to which evolution rules are applied.

3 Formal definitions related to rules application in a P system

Firstly, this section formally defines the concepts of multisets of objects, evolution rules, multisets of evolution rules, applicability benchmarks (maximal and minimal) of a rule to a multiset of objects. Secondly, on the basis of these definitions, requirements are specified for an application algorithm of rule evolution.

3.1 Multisets of Objects

Definition 1 *Multiset of object. Let a finite and not empty set of objects be O and the set of natural numbers $\mathbb{N}$, is defined as a multiset of object m as a mapping:*

$$\begin{aligned} m : O &\rightarrow \mathbb{N} \\ o &\rightarrow n \end{aligned}$$

Possible notations for a multiset of objects are:

$$\begin{aligned} m &= \{(o_1, n_1), (o_2, n_2), ..., (o_m, n_m)\} \\ m &= o_1^{n_1} \cdot o_2^{n_2} \cdot ... \cdot o_m^{n_m} \end{aligned}$$

Definition 2 *Set of multisets of object over a set of objects. Let a finite set of objects be O, the set of all the multisets that can be formed over set O is defined as:*

$$M(O) = \{m : O \rightarrow \mathbb{N} \mid \ m \ is \ a \ Multiset \ over \ O\}$$

Definition 3 *Multiplicity of object in a multiset of objects. Let an object be $o \in O$ and a multiset of objects $m \in M(O)$, the multiplicity of an object is defined over a multiset of object such as:*

$$\begin{aligned} \mid \ \ |_o : O \times M(O) &\rightarrow \mathbb{N} \\ (o, m) &\rightarrow |m|_o = \ n \mid (o, n) \in m \end{aligned}$$

Definition 4 *Weight or Cardinal of a multiset of objects. Let a multiset of objects be $m \in M(O)$, the weight or cardinal of a multiset of objects is defined as:*

$$\begin{aligned} \mid \ \ | : M(O) &\rightarrow \mathbb{N} \\ m &\rightarrow |m| \ = \sum_{\forall o \in O} |m|_o \end{aligned}$$

Definition 5 *Multiset support. Let a multiset of objects be $m \in M(O)$ and $\mathcal{P}(O)$ the part set of O, the support for this multiset is defined as:*

$$\begin{aligned} Supp : M(O) &\rightarrow \mathcal{P}(O) \\ m &\rightarrow Supp(m) = \{o \in O \ \mid \ |m|_o > 0\} \end{aligned}$$

Definition 6 *Empty multiset. This is the multiset represented by $\emptyset_{M(O)}$ and which satisfies:*

$$\emptyset_{M(O)} \ \Leftrightarrow \ |m| = 0 \Leftrightarrow Supp(m) = \emptyset$$

Definition 7 *Inclusion of multisets of objects. Let two multisets of objects be $m_1, m_2 \in M(O)$, the inclusion of multisets of objects is defined as:*

$$m1 \subset m2 \ \Leftrightarrow \ |m_1|_o \leq |m_2|_o \quad \forall o \in O$$

Definition 8 *Sum of multisets of objects. Let two multisets of objects be $m_1, m_2 \in M(O)$, the sum of multisets of objects is defined as:*

$$+ : M(O) \times M(O) \to M(O)$$
$$(m_1, m_2) \to \{(o, |m_1|_o + \ |m_2|_o) \ \forall o \in O\}$$

Definition 9 *Subtraction of multisets of objects. Let two multisets of objects be $m_1, m_2 \in M(O)$ and $m_2 \subset m_1$, the subtraction of the multisets of objects is defined as:*

$$- : M(O) \times M(O) \to M(O)$$
$$(m_1, m_2) \to \{(o, |m_1|_o - \ |m_2|_o) \ \forall o \in O\}$$

Definition 10 *Intersection of multisets of objects. Let two multisets of objects be $m_1, m_2 \in M(O)$, the intersection of multisets of objects is defined as:*

$$\bigcap : M(O) \times M(O) \to M(O)$$
$$(m_1, m_2) \to \{(o, \min(|m_1|_o, |m_2|_o)) \ \forall o \in O\}$$

Definition 11 *Scalar product of multiset of objects by a natural number. Let a multiset be $m \in M(O)$ and a natural number $n \in \mathbb{N}$, the scalar product is defined as:*

$$\cdot : M(O) \times \mathbb{N} \to M(O)$$
$$(m, n) \to m \cdot n = \{(o, |m|_o \cdot n) \ \forall o \in O\}$$

3.2 Evolution Rules

Definition 12 *Evolution rule over a set of objects with target in T and with no dissolution capacity. Let a set of objects be O, $a \in M(O)$ a multiset over O, $T = \{here, out\} \bigcup \{in_j / 1 \leq j \leq p\}$ a set of targets and $c \in M(O \times T)$ a multiset over $O \times T$, an evolution rule is defined like tuple:*

$$r = (a, c)$$

Definition 13 *Set of evolution rules over set of objects and targets in T. This set is defined as:*

$$R(O,T) = \{r \mid r \ is \ a \ rule \ over \ O \ and \ T\}$$

Definition 14 *Antecedent of Evolution Rule. Let an evolution rule be $r \in R(O,T)$, the antecedent of an evolution rule is defined over a set of objects as:*

$$input : R(O,T) \to M(O)$$
$$(a, c) \to input(r) = a \mid r = (a, c) \in R(O,T)$$

Definition 15 *Evolution rule applicable over a multiset of Objects. Let an evolution rule be $r \in R(O,T)$ and a multiset of objects $m \in M(O)$, it is said that an evolution rule is applicable over a multiset if and only if:*

$$\Delta_r(m) \Leftrightarrow input(r) \subset m$$

Definition 16 *Set of evolution rules applicable to a multiset of objects. Let a set of evolution rules be $R \in \mathcal{P}(R(O,T))$ and a multiset of objects $m \in M(O)$, the set of evolution rules applicable to a multiset is defined as:*

$$\Delta^* : \mathcal{P}(R(O,T)) \times M(O) \to \mathcal{P}(R(O,T))$$
$$(R, m) \to \Delta^*_R(m) = \{r \in R \mid \Delta r(m) = true\}$$

Property 1 *Maximal applicability benchmark of evolution rule over a multiset of objects*. Let an evolution rule be $r \in R(O,T)$ and a multiset of objects $m \in M(O)$, the maximal applicability benchmark of a rule in a multiset is defined as:

$$\Delta \lceil \ \rceil : R(O,T) \times M(O) \to \mathbb{N}$$
$$(r, m) \to \Delta_r \lceil m \rceil = \min \left\{ \frac{|m|_o}{|input(r)|_o} \right.$$
$$\left. \forall o \in Supp(m) \wedge |input(r)|_o \neq 0 \right\}$$

Property 2 *Minimal applicability benchmark of evolution rule over a multiset of objects and set of evolution rules*. Let an evolution rule be $r \in R(O,T)$, a multiset of objects $m \in M(O)$ and a set of evolution rules $R \in \mathcal{P}(R(O,T))$, the minimal applicability benchmark is defined as the function:

$$\Delta \lfloor \ \rfloor : R(O,T) \times M(O) \times \mathcal{P}(R(O,T)) \to \mathbb{N}$$
$$(r, m, R) \to \Delta_r \lfloor m \rfloor = \Delta_r \left\lceil m - \right.$$
$$\left. \left(m \bigcap \sum_{\forall r_i \in R - \{r\}} input(r_i) \cdot \Delta r_i \lceil m \rceil \right) \right\rceil$$

Property 3 *An evolution rule* $r \in R(O,T)$ *is applicable to a multiset of objects* $m \in M(O)$ if and only if the maximal applicability benchmark is greater than or equal to 1.

$$\Delta_r(m) \Leftrightarrow \Delta_r \lceil m \rceil \geq 1$$

Property 4 *The maximal applicability level of a rule* $r \in R(O,T)$ *over an object multiset* $m \in M(O)$ *is greater than or equal to the maximal applicability level of the rule in a subset of the multiset.*

$$\Delta_r \lceil m_1 \rceil \geq \Delta_r \lceil m_2 \rceil \ \forall m_1, m_2 \in M(O) \mid m_2 \subset m_1$$

Property 5 *If the maximal applicability benchmark of a rule* $r \in R(O,T)$ *over a multiset of objects* $m \in M(O)$ *is 0, then the maximal applicability benchmark of the rule* r *over the sum of* $input(r)$ *and* m *is equal to the maximal applicability benchmark of the* $input(r)$ *equal to 1.*

$$\Delta_r \lceil m \rceil = 0 \Rightarrow \Delta_r \lceil input(r) + m \rceil = \Delta_r \lceil input(r) \rceil = 1$$

3.3 Multisets of Evolution Rules

Definition 17 *Multiset of evolution rules. Let a finite and not empty set of evolution rules be* $R(O,T)$ *and the set of natural numbers* $\mathbb{N}$*, a multiset of evolution rules is defined as the mapping:*

$$M_{R(O,T)} : R(O,T) \rightarrow \mathbb{N}$$
$$r \rightarrow n$$

All definitions related to multisets of objects can be extended to multisets of rules.

Definition 18 *Linearization of evolution multiset of rules. Let a multiset of evolution rules be* $m_R = r_1^{k_1} \cdot r_2^{k_2} \cdot ... \cdot r_q^{k_q} \in M_{R(O,T)}$ *linearization of* m_R *is defined as:*

$$\sum_{i=1}^{q} r_i \cdot k_i \in R(O,T)$$

3.4 Requirements of Application of Evolution Rules over Multiset of objects

Application of evolution rules in each membrane of P Systems involves subtracting objects from the multiset by using rules antecedents. Rules used are chosen in a non-deterministic manner. The process ends when no rule is applicable. In short, rules application to a multiset of object in a membrane is a process of information transformation with input, output and conditions for making the transformation.

Given an object set $O = \{o_1, o_2, ..., o_m\}$ where $m > 0$, the **input** to the transformation process is composed of a multiset $\omega \in M(O)$ and $R \subset R(O,T)$, where:

$$\begin{aligned} \omega &= o_1^{n_1} \cdot o_2^{n_2} \cdot ... \cdot o_m^{n_m} \\ R &= \{r_1, r_2, ..., r_q\} \ being \ q > 0 \end{aligned}$$

In fact, the transformation only needs rules antecedents because this is the part that acts on ω. Let these antecedents be:

$$input(r_i) = o_1^{n_1^i} \cdot o_2^{n_2^i} \cdot ... \cdot o_m^{n_m^i} \quad \forall i = \{1, 2, ..., q\}$$

The **output** of the transformation process will be a multiset of object $\omega' \in M(O)$ together with the multiset of evolution rules applied $\omega_R \in M_{R(O,T)}$.

$$\begin{aligned} \omega' &= o_1^{n_1'} \cdot o_2^{n_2'} \cdot ... \cdot o_m^{n_m'} \\ \omega_R &= r_1^{k_1} \cdot r_2^{k_2} \cdot ... \cdot r_q^{k_q} \end{aligned}$$

Conditions for making the transformation are defined according to the following requirements:

Requirement 1: *The transformation process is described through the following system of equations:*

$$\begin{aligned} n_1 &= n_1^1 \cdot k_1 + n_1^2 \cdot k_2 + ... + n_1^q \cdot k_q + n_1' \\ n_2 &= n_2^1 \cdot k_1 + n_2^2 \cdot k_2 + ... + n_2^q \cdot k_q + n_2' \\ &... \\ n_m &= n_m^1 \cdot k_1 + n_m^2 \cdot k_2 + ... + n_m^q \cdot k_q + n_m' \end{aligned}$$

That is:

$$\sum_{j=1}^{q} n_i^j \cdot k_j + n_i^{'} = n_i \quad \forall i = \{1,\ 2,\ ...,\ m\}$$

or

$$\sum_{i=1}^{q} input\,(r_i) \cdot k_i + \omega' = \omega$$

The number of equations in the system is the cardinal of the set O. The number of unknowns in the system is the sum of the cardinals of the set O and the number of rules of R. Thus, the solutions are in this form:

$$\left(n_1^{'}, n_2^{'}, ..., n_m^{'}, k_1, k_2, ..., k_q\right) \in \mathbb{N}^{m+q}$$

Meeting the following restrictions

$$0 \leq n_i^{'} \leq n_i \quad \forall i = \{1,\ 2,\ ...,\ m\}$$

Moreover, taking into account the maximal and minimal applicability benchmarks of each rule, the solution must satisfy the following system of inequations:

$$\Delta_{r_j} \lfloor \omega \rfloor \leq k_j \leq \Delta_{r_j} \lceil \omega \rceil \quad \forall j = \{1,\ 2,\ ...,\ q\}$$

Requirement 2: *No rule of the set R can be applied over the multiset of objects ω', that is:*

$$\Delta_r\left(\omega'\right) = false\ \forall r \in R$$

Having established the above requirements, the system of equations may be incompatible (no rule can be applied) determinate compatible (there is a single multiset of rules as the solution to the problem) or indeterminate compatible (there are many solutions). In the last case, the rule application algorithm must provide a solution that is randomly selected from all possible solutions in order to guarantee non-determinism inherent to P systems.

4 Algorithm of active rule elimination

This section describes an evolution rules application algorithm to a multiset of objects whose execution time depends on the number of rules. The general idea of the algorithm is to eliminate one by one the rules of the active rule set. Each step for elimination of a rule X requires the sequential execution of 2 actions:

1. Any rule other than X belonging to the set of active rules is applied. The number of times it is applied will be a random number between 0 and its maximal applicability. In this way, each of the active rules has a possibility of being applied.

2. The rule X is applied a number of times that is equal to its maximal applicability benchmark. In this way, the rule is no longer applicable and disappears of the set of active rules.

To facilitate coding of the algorithm, the set of active rules is transformed into an ordered sequence R. Initially, the position of any rule r_i in the sequence is i. The pseudo-code of the algorithm is as follows:

(1) $\omega' \leftarrow \omega$
(2) $\omega_R \leftarrow \emptyset_{M_{R(O,T)}}$
(3) **FOR** $Last = \mid R \mid$ **DOWNTO** 1
(4) **BEGIN**
(5) **FOR** $Ind = 1$ **TO** $Last - 1$ **DO**
(6) **BEGIN**
(7) $Max \leftarrow \Delta_{R[Ind]} \lceil \omega' \rceil$
(8) $K \leftarrow random(0, Max)$
(9) $\omega_R \leftarrow \omega_R + \left\{R\left[Ind\right]^K\right\}$
(10) $\omega' \leftarrow \omega' - input\left(R\left[Ind\right]\right) \cdot K$
(11) **END**
(12) $Max \leftarrow \Delta_{R[Last]} \lceil \omega' \rceil$
(13) $\omega_R \leftarrow \omega_R + \left\{R\left[Last\right]^{Max}\right\}$
(14) $\omega' \leftarrow \omega' - input\left(R\left[Last\right]\right) \cdot Max$
(15) **END**

Analysis of the behavior of the algorithm shows that a significant improvement can be made. Sometimes, in the step of elimination of rule $R\left[j\right]$, elimination also occurs of rule $R\left[i\right]$, where $i < j$. There are two reasons for this

1. Rules applied before $R\left[i\right]$ in the elimination step consume the objects necessary for $R\left[i\right]$ to no longer be applicable.

2. The random value which defines the number of times application is made of rule $R\left[i\right]$ coincides with its maximal applicability value.

If rule $R\left[i\right]$ is eliminated in elimination step $R\left[j\right]$, it is no longer necessary to execute the pertinent elimination step. In this way, the number of rule elimination steps is reduced, and therefore the execution time of the algorithm. The pseudo-code of the optimized algorithm is as follows:

(1) $\omega' \leftarrow \omega$
(2) $\omega_R \leftarrow \emptyset_{M_{R(O,T)}}$
(3) $Ind \leftarrow 1$
(4) **REPEAT**
(5) $Max \leftarrow \Delta_{R[Ind]} \lceil \omega' \rceil$
(6) **IF** $Max \neq 0$ **THEN**
(7) **IF** $Ind = \mid R \mid$ **THEN**
(8) $K \leftarrow Max$
(9) **ELSE**
(10) $K \leftarrow random(0, Max)$
(11) $\omega_R \leftarrow \omega_R + \left\{ R[Ind]^K \right\}$
(12) $\omega' \leftarrow \omega' - input(R[Ind]) \cdot K$
(13) **ELSE**
(14) $K \leftarrow 0$
(15) **IF** $Ind = \mid R \mid$ **THEN**
(16) $Remove(R, Ind)$
(17) $Ind \leftarrow 1$
(18) **ELSE**
(19) **IF** $Max = K$
(20) $Remove(R, Ind)$
(21) **ELSE THEN**
(22) $Ind \leftarrow Ind + 1$
(23) **UNTIL** $\mid R \mid = 0$

This algorithm is composed of a loop which ends when the indexed set of active rules R is empty. In each iteration, three main actions are executed:

1. The maximal applicability benchmark is calculated for rule $R[Ind]$ and is stored in Max.

2. If rule $R[Ind]$ is applicable ($Max \neq 0$), then it is applied K times. The value of the number K, will either coincide with Max if the rule is the last one in the set of active rules and it must be eliminated, or it will be a random number between *0* and Max if it is not the last rule of R.

3. Rules are eliminated if possible and the loop index is updated. If the rule applied is the last one in R, then it is eliminated by means of the operation $Remove()$ and the loop starts over again at $Ind = 1$. Otherwise, if the rule was not applicable or is no longer applicable because it has been applied up to its maximal applicability, it is eliminated by means of the operation $Remove()$ and the index does not change. In any other case, the index is advanced to the next rule.

The operation $Remove(R, Actual)$ eliminates from the sequence R the rule occupying the position indicated by the value of $Actual$.

4.1 Correctness

The algorithm presented is correct because:

Lemma 1 *The algorithm is finite.*

Proof: The algorithm ends when the condition of loop ending is attained (sentence 23); that is, when the sequence of active rules is empty. This occurs because in each iteration, either a rule is eliminated from the sequence by the operation $Remove()$ (sentences 16 and 20), or the index showing the position of the rule to be applied approaches the last element in sequence R (sentence 22), which in subsequent iterations will be eliminated (sentence 20).

Lemma 2 *No evolution rule is applicable to* ω'

Proof: The sequence R initially contains all rules applicable to ω'. To eliminate a rule from sequence R, its maximal applicability benchmark must be zero. Owing to **property 3** we know that a rule with a maximal applicability benchmark equal to zero is not applicable. Therefore, when a rule is eliminated from the sequence R, it is because it is not applicable. At the end of the algorithm, the sequence R is empty, so there can be no rule applicable to ω'.

Lemma 3 *Any result generated is a possible solution*

Proof: Starting with the input of the algorithm composed of the multiset ω and the sequence of rules R, construction of the solution follows these steps:

Let k_j^i be the number of times the rule j is applied in ith time the rule was applied.

Let R_j^i be the set of applicable rules when j was the last rule applied and i times an attempt was made to apply all active rules.

Let R^i be the set of applicable rules after all rules have been applied a maximal of i times

For $i = 1$ the following must be:

$$k_1^1 \in \{0, \ \ldots \ , \Delta_{r_1} \lceil \omega \rceil\}$$

$$\Delta_{r_1} \lceil \omega \rceil = 0 \vee k_1^1 = \Delta_{r_1} \lceil \omega \rceil \Rightarrow r_1 \notin R_1^0$$

$$R_1^0 = \Delta_{R^0}^*(\omega - k_1^1 \cdot input(r_1))$$

For application of the following rule, R_1^0 is considered.

If $r_2 \in R_1^0$

$$k_2^1 \in \left\{0, ..., \Delta_{r_2} \left\lceil \omega - k_1^1 \cdot input(r_1) \right\rceil\right\}$$

$$\Delta_{r_2}\lceil\omega\rceil = 0 \vee k_2^1 = \Delta_{r_2}\lceil\omega\rceil \Rightarrow r_2 \notin R_2^0$$
$$R_2^0 = \Delta^*_{R^0}(\omega - k_1^1 \cdot input(r_1) - k_2^1 \cdot input(r_2))$$

For application of the following rule R_2^0 is considered:

...
If $r_q \in R_{q-1}^0$

$$k_q^1 = \Delta_{r_q}\left\lceil \omega - \sum_{j=1}^{q-1} k_j^1 \cdot input(r_j)\right\rceil$$

$$r_q \notin R_q^0 = R^1 = \Delta^*_{R^0}(\omega - \sum_{j=1}^{q} k_j^1 \cdot input(r_j))$$

If $\left|R^1\right| = 0$ the algorithm ends, otherwise rules that are still applicable may be applied again.

For $i = 2$ it must be that:
If $r_1 \in R^1$

$$k_1^2 \in \left\{0,\ ...\ , \Delta_{r_1}\left\lceil \omega - \sum_{j=1}^{q} k_j^1 \cdot input(r_j)\right\rceil\right\}$$

$$\Delta_{r_1}\lceil\omega\rceil = 0 \vee k_1^2 = \Delta_{r_1}\lceil\omega\rceil \Rightarrow r_1 \notin R_1^1$$
$$R_1^1 = \Delta^*_{R^0}(\omega - \sum_{j=1}^{q} k_j^1 \cdot input(r_j) - k_1^2 \cdot input(r_1))$$

If $r_2 \in R_1^1$

$$k_2^2 \in \left\{0,\ ...\ , \Delta r_2\left\lceil \omega - \sum_{j=1}^{q} k_j^1 \cdot input(r_j) - k_1^2 \cdot input(r_1)\right\rceil\right\}$$

$$\Delta_{r_2}\lceil\omega\rceil = 0 \vee k_2^2 = \Delta_{r_2}\lceil\omega\rceil \Rightarrow r_2 \notin R_2^1$$

$$R_2^1 = \Delta^*_{R^0}(\omega - \sum_{j=1}^{q} k_j^1 \cdot input(r_j) - k_1^2 \cdot input(r_1)\ - k_2^2 \cdot input(r_2))$$

In general, if the rule r_j is applicable the ith time, the following is satisfied:

$$k_j^i \in \left\{0,\ ...\ , \Delta_{r_j}\left\lceil \omega - \sum_{ii=1}^{i-1}\sum_{jj=1}^{q-ii+1} k_{jj}^{ii} \cdot input(r_{jj})\ - \sum_{jj=1}^{j-1} k_{jj}^i \cdot input(r_{jj})\right\rceil\right\} \quad (1)$$

Except when r_j is the last rule in the sequence R^i, in which case:

$$k_j^i = \Delta_{r_j}\left\lceil \omega - \sum_{ii=1}^{i-1}\sum_{jj=1}^{q-ii+1} k_{jj}^{ii} \cdot input(r_{jj}) - \sum_{jj=1}^{j-1} k_{jj}^i \cdot input(r_{jj}\right\rceil \quad (2)$$

$$R_j^i = \Delta^*_{R^0}(\omega - \sum_{ii=1}^{i-1}\sum_{jj=1}^{q-ii+1} k_{jj}^{ii} \cdot input(r_{jj}) - \sum_{jj=1}^{j} k_{jj}^i \cdot input(r_{jj}))$$

and

$$R^{i+1} = R_x^i\ being\ x\ \ the\ last\ rule\ in\ R^i$$

The general expression of the rule multiset solution will be:

$$\omega_R = r_1^{\sum_{i=1}^{s_1} k_1^i} \cdot r_2^{\sum_{i=1}^{s_2} k_2^i} \cdot ... \cdot r_q^{\sum_{i=1}^{s_q} k_q^i}$$

$$being\ s_i \in \{1, ..., q - i + 1\}$$

or

$$\omega_R = r_1^{S_1^*} \cdot r_2^{S_2^*} \cdot ... \cdot r_q^{S_q^*}$$

$$being\ S_j^* = \sum_{i=1}^{s_j} k_j^i \mid s_j \in \{1, ..., q - j + 1\}$$

Let us see if the solution meets the established requirements:

1. The solution satisfies the system of equations $\sum_{j=1}^{q} input(r_j) \cdot S_j^* \subset \omega$ due to the construction of the solution, since whenever a rule is added to the set ω_R it is known to be applicable, as seen in the equation of the general (**equation 1**) term of each k_j^i, and therefore, fewer objects are always consumed than those containing ω.

2. No rule is applicable, that is ω_R is maximal. **Lemma 2** garantees this fact.

Lemma 4 *Any solution possible is generated by the algorithm*

Proof: Let $\omega_R = r_1^{k_1} \cdot r_2^{k_2} \cdot ... \cdot r_q^{k_q}$ be a solution to the problem.

It is known that:

$$\sum_{i=1}^{q} input(r_i) \cdot k_i + \omega' = \omega$$

Then

$$\forall i, , \ input(r_i) \cdot k_i + \omega' = \omega - \sum_{j=1 \wedge j \neq i}^{q} input(r_j) \cdot k_j$$

Applying the maximal applicability benchmark yields:

$$\forall i, , \ \Delta_{r_i} \left\lceil input(r_i) \cdot k_i + \omega' \right\rceil = \Delta_{r_i} \left\lceil \omega - \sum_{j=1 \wedge j \neq i}^{q} input(r_j) \cdot k_j \right\rceil$$

As $\Delta_r(\omega') = 0 \ \forall r \in R$, owing to **property 5**:

$$\forall i, , \ \Delta_{r_i} \left\lceil input(r_i) \cdot k_i \right\rceil = \Delta_{r_i} \left\lceil \omega - \sum_{j=1 \wedge j \neq i}^{q} input(r_j) \cdot k_j \right\rceil$$

So multiplicities of solution rules generated by the algorithm is characterized by:

$$\forall i, , \ k_i = \Delta_{r_i} \left\lceil \omega - \sum_{j=1 \wedge j \neq i}^{q} input(r_j) \cdot k_j \right\rceil$$

Any solution to the problem is achievable in q steps of the algorithm because, firstly, in the interval defining the value $k_i^1 \ \forall i \in \{1, ..., q-1\}$ (**equation 1**) we find the solution value k_i and secondly, the value assigned to k_q^1 (**equation 2**) coincides with the value of k_q.

Lemma 5 *The algorithm is not determinist*

Proof: This occurs when a rule is not the last one in the set, it is applied a randomly determined number of times (sentence 10) between 0 and its maximal applicability value.

4.2 Complexity

The worst case of the optimized algorithm for rule elimination occurs when sentence 20 is never executed, that is, when $\forall j = \{1, ..., q\}$ in the elimination step of a rule $R[j]$ we never eliminate another rule $R[i]$ where $i < j$. In this case, there is no improvement in the behavior of the optimized algorithm and the number of iterations the loop is repeated is:

$$\#iterations = \sum_{i=1}^{q} i = \frac{q \cdot (q+1)}{2}$$

Moreover, in the loop body, the heaviest operations are those for calculating the maximal applicability benchmark (sentence 5), the scalar product of the $input$ of a rule by a whole number (sentence 10) and the difference of two multisets (sentence 10). All these operations are linearly dependent on the $cardinal$ of the multiset support ω.

$$\#operations\ per\ iteration \approx 3 \cdot \mid Supp(\omega) \mid$$

Therefore, the number of operations of the algorithm is:

$$\#operations \approx 3 \cdot \mid Supp(\omega) \mid \cdot \frac{q \cdot (q+1)}{2}$$

So the execution time of the algorithm at worst is linear dependent on the square of the number of rules.

The algorithm performs best when all rules are applied once with the maximal applicability benchmark. In this case, the number of times the loop is executed is q, and thus the execution time of the algorithm is linear dependent on the number of rules.

5 Comparison tests

The experimental tests have compared the "Active Rules Elimination" algorithm with the fastest sequencial algorithm that is "Maximal Applicability Benchmark"[**?**]. The experimental trial game used to test both algorithms has taken into account 3 parameters:

1. Number of objects of the multiset is 16

2. Number of rules with a value belonging to the set $\{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$

3. Relationship between the cardinal of the multiset and the $cardinal$ of the sum of $inputs$ of active rules with a value belonging to the set $\{1, 10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7\}$

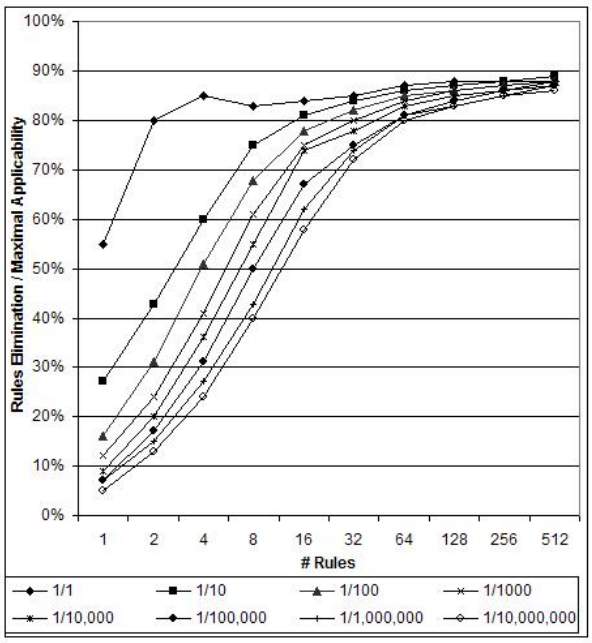

Figure 1: Comparison tests

Figure **??** shows a graphic with the results obtained in the tests. Each curve of the graphic represents the percentage of execution time of "Active Rules Elimination" algorithm with regards to the execution time of "Maximal Applicability benchmark" algorithm for each of the values of the relationship of the cardinals. In this graphic you can see that "Active Rules Elimination" algorithm is always better than "Maximal Applicability benchmark" algorithm independently of the numbers of rules and the relationship between cardinals. However, taking into account that the published P system have no more than 16 rules by membrane, the execution time used by "Active Rules Elimination" algorithm is much smaller than the one used by "Maximal Applicability benchmark" algorithm. For example, in the best case (1 rule in the membrane and the relationship of the cardinals equals to 10^7) the execution time "Active Rules Elimination" algorithm is only 5% of the "Maximal Applicability benchmark" algorithm.

6 Conclusions

This paper has described a new sequential algorithm of rules application to a multiset that attains a certain degree of parallelism, as a rule can be applied a number of times in a single step. This algorithm is the first one where the number of operations performed is limited, thus allowing for determination of execution time beforehand. This information is essential to calculate the number of membranes that have to be located in each processor in distributed implementation architectures of P systems to achieve optimal times with minimal resources. In experimental tests, this algorithm has shown a better execution time than previously published sequential algorithms.

References:

[1] G. Ciobanu, W. Guo , *P Systems Running on a Cluster of Computers*, Workshop on Membrane Computing LNCS 2933, 2004 pp.123-139

[2] L. Fernández, F. Arroyo, J. Castellanos, J. Tejedor, I. García , *New Algorithms for Application of Evolution Rules based on Applicability Benchmarks*, BioInformatics & Computational Biology, 2006 pp. 94-100

[3] L. Fernández, F. Arroyo, J. Tejedor, J. Castellanos, *Massively Parallel Algorithm for Evolution Rules Application in Transition P Systems*, Pre-Proceedings of Workshop on Membrane Computing (WMC7) Leiden, Netherlands, 2006, pp. 337-343

[4] A. Gutiérrez, L. Fernández, F. Arroyo, V. Martínez, *Design of a hardware architecture based on microcontrollers for the implementation of membrane system*, Proceedings on 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC-2006). Timisoara (Rumania) September, 2006, pp. 39-42.

[5] V. Martínez, L. Fernández, F. Arroyo, A. Gutiérrez, *HW Implementation of a Bounded Algorithm for Application of Rules in a Transition P-System*, Proceedings on 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC-2006). Timisoara, Romania, 2006, pp. 32-38.

[6] Gh. Păun *Membrane Computing. An Introduction*, Springer-Verlag, 2002

[7] J. Tejedor, L. Fernández, F. Arroyo, G. Bravo, *An Architecture for Attacking the Bottleneck Communication in P Systems*, The Twelfth International Symposium on Artificial Life and Robotics, 2007, pp. 500-505