SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2018, Vol. 61 092103:1–092103:12
doi: 10.1007/s11432-017-9103-5

Universal enzymatic numerical P systems with small number of enzymatic variables

Zhiqiang ZHANG[1], Tingfang WU[1], Andrei PĂUN[3,4] & Linqiang PAN[1,2*]

[1]*Key Laboratory of Image Information Processing and Intelligent Control of Education Ministry of China, School of Automation, Huazhong University of Science and Technology, Wuhan* 430074*, China;*
[2]*School of Electric and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou* 450002*, China;*
[3]*Department of Computer Science, Faculty of Mathematics and Computer Science, University of Bucharest, Bucuresti* 010014*, Romania;*
[4]*Bioinformatics Department, National Institute of Research and Development for Biological Sciences, Bucharest* 060031*, Romania*

Received 26 January 2017/Accepted 20 April 2017/Published online 20 November 2017

Abstract Numerical P systems (for short, NP systems) are distributed and parallel computing models inspired from the structure of living cells and economics. Enzymatic numerical P systems (for short, ENP systems) are a variant of NP systems, which have been successfully applied in designing and implementing controllers for mobile robots. Since ENP systems were proved to be Turing universal, there has been much work to simplify the universal systems, where the complexity parameters considered are the number of membranes, the degrees of polynomial production functions or the number of variables used in the systems. Yet the number of enzymatic variables, which is essential for ENP systems to reach universality, has not been investigated. Here we consider the problem of searching for the smallest number of enzymatic variables needed for universal ENP systems. We prove that for ENP systems as number acceptors working in the all-parallel or one-parallel mode, one enzymatic variable is sufficient to reach universality; while for the one-parallel ENP systems as number generators, two enzymatic variables are sufficient to reach universality. These results improve the best known results that the numbers of enzymatic variables are 13 and 52 for the all-parallel and one-parallel systems, respectively.

Keywords bio-inspired computing, numerical P system, mobile robot, membrane controller, universality

Citation Zhang Z Q, Wu T F, Păun A, et al. Universal enzymatic numerical P systems with small number of enzymatic variables. Sci China Inf Sci, 2018, 61(9): 092103, doi: 10.1007/s11432-017-9103-5

1 Introduction

Natural computing is a highly interdisciplinary field that connects the natural sciences with computing science [1]. The main idea of this field is to develop computational tools (algorithms) by taking inspiration from nature for the solution of complex practical problems from a variety of domains, such as engineering, biology, economics, manufacturing. Natural computing provides many ideas and models for solving problems from control field. For example, the model of neurofuzzy networks is proved to be very effective in the modeling and control of nonlinear processes; the multilayer neural network and the recurrent network are applied extremely successfully in pattern recognition and optimization problems [2].

As a branch of natural computing, membrane computing [3] aims to develop new problem solving techniques and new computing models (called membrane systems or P systems) based upon the architecture

* Corresponding author (email: lqpan@mail.hust.edu.cn; lqpanhust@gmail.com)

and the functioning of living cells. Besides the rich theoretical results [4, 5], membrane computing has been applied in a wide range of areas [6], such as systems and synthetic biology (e.g., modeling the process of biological and bio-medical [7]), computer science (e.g., sorting and ranking [8], approximate algorithms for optimization problems [9,10]), linguistics (e.g., making use of the parallelism for solving parsing problems in an efficient way [11]), and cryptography (e.g., attacking public-key cryptographic system [12,13]). Recently, membrane systems have also been applied in various complex real-world problems, such as, population dynamic P system being used to predict fluctuations of population size of rare wild animals in ecological systems [14, 15], spiking neural P systems [16] being used in fault diagnosis [17–19] and combinatorial optimization problems [20]. For the most up-to-date results and news, one can consult the P systems web page [1)].

Numerical P systems (for short, NP systems) are a class of cell-like membrane systems, which are inspired both from the structure of living cells and the economic reality [21]. Briefly, NP systems process numerical variables (instead of symbols) in the compartments defined by membranes. Each variable has initial value (integer or real number). The values of these variables are changed by evolution programs, which are composed of production functions and a repartition protocols. A production function (is a general real function), taking the local variables, computes a number, which is distributed among the local variables, the variables from the immediately upper compartment, and those from the immediately lower compartments. The distribution that a variable obtains from a program is proportional to its coefficient in the repartition protocol of this program. The values assumed by a distinguished variable is the set of numbers computed by the system.

NP systems have been applied to solve the classic control problems (e.g., pole balancing problem [22]), and to design membrane controllers to control autonomous mobile robots. Membrane controllers are developed to control autonomous mobile robots to generate various desired behaviors and cognitive abilities, like obstacle avoidance, localization, moving to a given position, wall following, following another robot [23]. A novel trajectory tracking control approach is proposed for nonholonomic wheeled mobile robots, where an enzymatic numerical membrane system is used to integrate two proportional-integral-derivative controllers [24]. The advantages of using NP systems in robotics include the parallel nature of the model, the modularity of encapsulating behavior and functionalities as modules, and the good adaptability for platforms of robot in different size scale. The performance of a membrane controller is comparable to or better than that of other controllers (such as fuzzy logic controllers). After designing a membrane controller, the numerical membrane system can be simulated by simulators such as SNUPS [25, 26], SimP [27] or the parallelized GPU-based simulators for ENP systems [28].

ENP systems (enzymatic numerical P systems) are a variant of NP systems, where a special variable called enzyme is introduced to control the use of programs [29]. A program can be applied only when the value of the enzyme is strictly greater than the smallest value of variables involved in the production function. Theoretical investigations have been made for NP systems, both for computational power [21] and computational efficiency [30]. Especially for ENP systems working in the all-parallel and one-parallel modes, the problem concerning the descriptional complexity has been investigated [31–33]. In most studies, the following complexity parameters are considered: the number of membranes/cells, the degrees and numbers of variables of polynomial production functions. Particularly, in [31] all these parameters are optimized to 1. However, an important complexity parameter, the number of enzymatic variables, still remain to be optimized. The known best numbers of enzymatic variables used in the all-parallel and one-parallel universal systems are 13 and 52, respectively [31].

In fact, enzymatic variables are essential for ENP systems to reach universality. Deterministic sequential numerical P systems without enzymatic variables compute more than semilinear sets, while with enzymatic variables they are universal [2)]. Hence, it is an interesting problem to find the smallest number of enzymatic variables needed for universal ENP systems [34].

In this work, we give an optimal boundary for the number of enzyme variables used in the universal

1) http://ppage.psystems.eu.

2) Zhang Z, Pan L. The computational power of enzymatic numerical P systems working in the sequential mode. to be submitted.

ENP systems. Specifically, we prove that for ENP systems as accepting devices, working in the all-parallel mode, one enzymatic variable is enough to reach universality. While as generating devices, working in the one-parallel mode, the number of enzymatic variables needed for universal ENP systems is two. These results improve the known best numbers of enzymatic variables 13 and 52 used in the all-parallel and one-parallel ENP systems to 1 and 2, respectively. These results show that simplified ENP systems, in terms of the number of enzymatic variables, are still powerful, that is, Turing universal, which implies that it is possible to simplify ENP systems without decreasing their computation power when ENP systems are used to solve real-life problems.

2 Enzymatic numerical P system

The definition of enzymatic numerical P system was proposed in [29]. To make the paper self-contained, we will recall the definition given in [31]. An ENP system is a construct

$$\Pi=(m,H,\mu,(\mathrm{Var}_1,\mathrm{Pr}_1,\mathrm{Var}_1(0)),\ldots,(\mathrm{Var}_m,\mathrm{Pr}_m,\mathrm{Var}_m(0)),V_{\mathrm{in}},V_{\mathrm{out}}),$$

where

- $m\geqslant 1$ is the number of membranes.
- H is an alphabet of labels for membranes in μ.
- μ is a rooted tree with m nodes labeled with the elements of H.
- Var_i, $1\leqslant i\leqslant m$, is the set of variables in region i.
- $\mathrm{Var}_i(0)$, $1\leqslant i\leqslant m$, is the set of initial values of the variables in region i.
- Pr_i, $1\leqslant i\leqslant m$, is the set of programs in region i, each program has one of the two following forms:

(i) non-enzymatic

$$F_{j,i}(x_{1,i},\ldots,x_{k_i,i})\rightarrow c_{l,i,1}|v_{l,i,1}+\cdots+c_{l,i,n_i}|v_{l,i,n_i},$$

where $F_{j,i}(x_{1,i},\ldots,x_{k_i,i})$ is the production function, and $c_{l,i,1}|v_{l,i,1}+\cdots+c_{l,i,n_i}|v_{l,i,n_i}$ is the repartition protocol of the program.

(ii) enzymatic

$$F_{j,i}(x_{1,i},\ldots,x_{k_i,i})|_{e_{j,i}}\rightarrow c_{l,i,1}|v_{l,i,1}+\cdots+c_{l,i,n_i}|v_{l,i,n_i},$$

where $e_{j,i}$ is an enzymatic variable from Var_i different from $x_{1,i},\ldots,x_{k_i}$ and from $v_{l,i,1},\ldots,v_{l,i,n_i}$.

- V_{in} and V_{out} are sets of input and output variables, respectively.

The kind of membrane structure considered here is the cell-like one, which mathematically corresponds to a rooted tree, and can be represented by an expression of correctly matching labeled square brackets. In the compartments of the membrane structure, the variables from region i are written in the form $x_{j,i}, j\geqslant 1$. The value of $x_{j,i}$ at time $t\in N$ (the set of natural numbers) is denoted by $x_{j,i}(t)$. (The time is supposed to be marked by a universal clock, the same for all membranes.) The values of the variables can be real numbers, but here we only work with integers, positive or negative.

The system uses programs to evolve the values of variables. Each program is composed of two parts: a production function and a repartition protocol. The former can be any function defined on the Cartesian product Var^k $(1\leqslant k\leqslant n)$ with values from $\mathbb{Z}$ (the set of integer numbers). Here only polynomial functions are considered. By using the production function, the system computes a production value from the values of its variables at that time. This value is distributed to the variables from the region where the program resides, and to the variables in its upper (parent) and lower (children) compartments, as specified by the repartition protocol. Specifically, for a repartition protocol $\mathrm{RP}_{l,i}$, variables $v_{l,i,1},\ldots,v_{l,i,n_i}$ come from the membrane i where the program resides, the parent membrane and the children membrane. Formally, $\{v_{l,i,1},\ldots,v_{l,i,n_i}\}\subseteq \mathrm{Var}_i\cup\mathrm{Var}_{\mathrm{par}(i)}\cup(\bigcup_{\mathrm{ch}\in\mathrm{Ch}(i)}\mathrm{Var}_{\mathrm{ch}})$, where $\mathrm{par}(i)$ is the parent of membrane i and $\mathrm{Ch}(i)$ is the set of children of membrane i. The coefficients $c_{l,i,1},\ldots,c_{l,i,n_i}$ are natural numbers (they may be 0, in this case the terms "$+0|x$" are omitted), which specify the proportion of the current production value distributed to each variable $v_{l,i,1},\ldots,v_{l,i,n_i}$. At time t, if we denote with $C_{l,i}=\sum_{s=1}^{n_i}c_{l,i,s}$ the sum

of all coefficients of the repartition protocol, and denote with

$$q_{l,i}(t) = \frac{F_{j,i}(x_{1,i}, \ldots, x_{k_i,i})}{C_{l,i}}, \tag{1}$$

the "unitary portion", then the value $\mathrm{ad}_{l,i,r}(t) = q_{l,i}(t) \cdot c_{l,i,r}$ represents the value added to variable $v_{l,i,r}$. If variable $v_{l,i,r}$ appears in several repartition protocols, for example, $\mathrm{RP}_{j,i_1}, \ldots, \mathrm{RP}_{j,i_k}$, all these values $\mathrm{ad}_{j,i_1,r_1}, \ldots, \mathrm{ad}_{j,i_k,r_k}$ are added to variable $v_{l,i,r}$. After computing the production function value, the variables involved in the production function are reset to zero. So, if at time t variable v_j is involved in at least one production function, then its value at time $t+1$ is $v_j(t+1) = \sum_{l=i_1}^{i_k} \mathrm{ad}_{j,l}(t)$; otherwise, $v_j(t+1) = v_j(t) + \sum_{l=i_1}^{i_k} \mathrm{ad}_{j,l}(t)$.

Note that in (1), $q_{l,i}(t)$ is an integer only if the value of the production function $F_{j,i}(x_{1,i}, \ldots, x_{k_i,i})$ is divisible by the respective sum $C_{l,i}$. If at any step, all the values of the production functions are divisible by the respective sums, then we associate this kind of systems with the notation div. In this paper, all the systems we construct are of the div type. Possible decisions for the case when the current production is not divisible by the associated coefficients total can be found in [21].

Besides the non-enzymatic programs, ENP systems also have enzymatic programs: $\mathrm{Pr}_{j,i} = F_{j,i}(x_{1,i}, \ldots, x_{k_i,i})|_{e_{j,i}} \to c_{l,i,1}|v_{l,i,1} + \cdots + c_{l,i,n_i}|v_{l,i,n_i}$. Such a program is applied at time t only if $e_{j,i}(t) > \min(x_{1,i}(t), \ldots, x_{k_i,i}(t))$. Note that enzymatic variable $e_{j,i}$ remains unchanged in the program where it appears as an enzymatic variable; in other programs, $e_{j,i}$ can appear as a usual variable in production functions or repartition protocols, and it can be "consumed" or receive "contributions".

The programs can be applied in the all-parallel mode, one-parallel mode, and sequential mode. The details of these three modes are as follows.

• All-parallel mode: at each step, in each membrane, all the applicable programs are applied, allowing that more than one program shares the same variable.

• One-parallel mode: programs are applied with the restriction that one variable can appear in only one of the applied programs; in the case of multiple choices, the programs to be applied are chosen in the non-deterministic way. More precisely, the set of programs chosen to apply in one membrane should satisfy two conditions: (1) the intersection of any two sets of production variables associated with two programs is empty; (2) the set of programs is maximal, that is, the condition (1) will be not satisfied if one more program is added to the set of programs.

• Sequential mode: at each step, only one program is applied in each membrane; if more than one program in a membrane can be used, then one of them is non-deterministically chosen.

In what follows, we give an example to illustrate these modes. Suppose in a membrane there are the programs $\mathrm{Pr}_1 : x_1 + x_2 \to 1|x_1$; $\mathrm{Pr}_2 : x_1 \to 1|x_2$; and $\mathrm{Pr}_3 : x_2 \to 1|x_2$. In the all-parallel mode, the applicable set of programs is $\{\mathrm{Pr}_1, \mathrm{Pr}_2, \mathrm{Pr}_3\}$. In the one-parallel mode, the applicable set of programs can be $\{\mathrm{Pr}_1\}$ or $\{\mathrm{Pr}_2, \mathrm{Pr}_3\}$, but the set of programs such as $\{\mathrm{Pr}_2\}$ (the set of programs is not maximal, since program Pr_3 can be added and condition (1) is still satisfied) and $\{\mathrm{Pr}_1, \mathrm{Pr}_2, \mathrm{Pr}_3\}$ (condition (1) is not satisfied) are not applicable. In the sequential mode, the applicable set of programs can be any one of the three programs. In this work, we only consider the systems working in the all-parallel and one-parallel modes.

The values of all the variables of Π at time $t \in N$ form a configuration of Π, which can be denoted as the sequence $(\mathrm{Var}_1(t), \ldots, \mathrm{Var}_m(t))$. At time 0, the configuration $(\mathrm{Var}_1(0), \ldots, \mathrm{Var}_m(0))$ is called the initial configuration. Using programs in the way mentioned above, the system transits from one configuration to another one. A sequence of such transitions forms a computation. If no applicable set of programs produces a change in the current configuration, we say that the system reaches a halting configuration. The output of a computation is the set of values taken by the variables of V_{out} in the halting configuration. If the system never reaches a halting configuration, then no result is obtained.

In this way, an ENP system can compute a function f: $N^\alpha \to N^\beta$ $(\alpha, \beta \geqslant 0)$: the α values of the arguments are introduced in the system as the initial values of variables in V_{in} and the β-vector of the function value is obtained in the variables from V_{out} in the halting configuration of the system. If the system never reaches a halting configuration, then no result is obtained.

3 Previous results

Here we give some denotations used in the following proofs. The set of natural numbers generated or accepted by an ENP system Π is denoted by $N_\alpha(\Pi)$, $\alpha \in \{\text{gen}, \text{acc}\}$, where gen and acc denote the system working in the generating and accepting mode respectively. In what follows, only polynomials with integer coefficients are considered as production functions of enzymatic numerical P systems. Two important parameters describing the complexity of a polynomial are the degree of the polynomial and the number of variables. According to the parameters describing the complexity of an enzymatic numerical P system (the number of enzymatic variables and the membranes, the polynomials used as production functions), we use

$$N_\alpha E_{m_1} NP_{m_2}(\text{poly}^n(r), \beta)$$

to denote the family of all sets $N_\alpha(\Pi)$ of numbers computed by systems Π working in α mode with m_1 enzymatic variables and at most m_2 membranes; production functions which are polynomials of degree at most n, with integer coefficients, with at most r variables in each polynomial; using the rules in the mode $\beta \in \{\text{allP}, \text{oneP}\}$, where allP stands for all-parallel, and oneP stands for one-parallel. If one of the parameters m_1, m_2, n, r is not bounded, then it is replaced with $*$.

With this notation, the results concerning the computational power of ENP systems working in the all-parallel and one-parallel modes in [33] can be denoted as follows:

$$\begin{aligned} \text{NRE} &= N_{\text{gen}} E_* NP_*(\text{poly}^1(2), \text{oneP}) &(2)\\ &= N_{\text{gen}} E_{771} NP_{254}(\text{poly}^2(253), \text{allP}), &(3)\end{aligned}$$

where NRE denotes the family of computable (recursively enumerable) sets of natural numbers. Eq. (3) was improved to the following result [32]:

$$\text{NRE} = N_{\text{gen}} E_{263} NP_4(\text{poly}^1(6), \text{allP}).$$

The latest improvements of the above results are shown in [31]:

$$\begin{aligned} \text{NRE} &= N_{\text{acc}} E_{13} NP_1(\text{poly}^1(1), \text{allP})\\ &= N_{\text{acc}} E_{52} NP_1(\text{poly}^1(1), \text{oneP})\\ &= N_{\text{gen}} E_{52} NP_1(\text{poly}^1(1), \text{oneP}).\end{aligned}$$

In the following sections we further improve these results in terms of the number of enzymatic variables.

4 Number of enzymatic variables in all-parallel ENP systems

In this section, we will prove that for ENP systems working in the all-parallel mode, one enzymatic variable suffices to reach universality.

Theorem 1. $N_{\text{acc}} E_1 NP_1(\text{poly}^1(2), \text{allP}) = \text{NRE}$.

Proof. We only prove $N_{\text{acc}} E_1 NP_1(\text{poly}^1(2), \text{allP}) \supseteq \text{NRE}$, since the converse inclusion is straightforward but cumbersome (for similar technical details, please refer to Subsection 8.1 in [5]) or we can invoke for it from the Turing-Church thesis.

It is known that deterministic register machines compute all sets of numbers which are Turing computable, hence they characterize NRE [35]. This theorem is proved by simulating deterministic register machine.

A deterministic register machine is a construct $M = (m, H, l_1, l_n, I)$, where m is the number of registers, H is the set of instruction labels, l_1 is the start label, l_n is the halt label, and I is the set of instructions; each element of H labels only one instruction from I, thus precisely identifying it. The instructions are of the following forms.

- $l_i : (\text{ADD}(r), l_k)$ (add 1 to register r and then go to the instruction with label l_k).

• l_i : (SUB$(r), l_j, l_k$) (if register r is non-zero, then subtract 1 from it and go to the instruction with label l_j; otherwise, go to the instruction with label l_k).

• l_n : HALT (the halt instruction).

A register machine can be used as an acceptor: the machine starts with all registers empty (namely the number stored in each register is zero), except for the first register, the input one, where a number x is introduced; the computation starts with instruction labeled by l_1 and, if it halts, then the number x is accepted. In this way, again all sets of numbers from NRE are characterized (even by deterministic register machines).

Let $M = (m, H, l_1, l_n, I)$ be a deterministic register machine working in the accepting mode. The number z is introduced in the first register. The computation starts with instruction labeled by l_1 and, if it halts, then the number z is accepted. In what follows, a specific ENP system Π_M working in the all-parallel mode is constructed to simulate the register machine M.

$$\Pi_M = (1, \{0\}, [\ ]_0, (\mathrm{Var}_0, \mathrm{Pr}_0, \mathrm{Var}_0(0)), V_{\mathrm{in}}),$$

where $\mathrm{Var}_0 = \{x_i, x'_1, l_j, l'_j, l'_1, e \mid 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\}$. $\mathrm{Var}_0(0)$ is the vector of initial values of all the variables, obtained by setting $x'_1 = z$; $e = x_i = 0$, for all $1 \leqslant i \leqslant m$; $l'_1 = 1$; $l_j = 0$, for all $1 \leqslant j \leqslant n$. $V_{\mathrm{in}} = \{x'_1\}$. $\mathrm{Pr}_0 = \{-x'_1 \to 1|x_1, l'_1 \to 1|l_1\} \cup \{-l_i \to 1|x_r, l_i \to 1|l_k \mid \text{for } l_i : (\mathrm{ADD}(r), l_k) \in I\} \cup \{l_i \to 1|l_k, 3(l_i + x_r)|_e \to 1|l'_j + 1|l_k + 1|x_r, -2(2l_i + x_r)|_e \to 1|l'_j + 1|l_k, -l'_j|_e \to 1|l_j \mid \text{for } l_i : (\mathrm{SUB}(r), l_j, l_k) \in I\}$.

For each $1 \leqslant i \leqslant m$, there is a variable x_i in ENP system Π_M storing the number saved in register i of register machine M. Specifically, when the value stored in register i is q, the value of variable x_i is $-q$. Variable e is the only enzymatic variable, which does not appear in any production function or repartition protocol, hence its initial value 0 keeps unchanged during computations. For each instruction l_i of register machine M, there is a variable l_i in Π_M. At the beginning of each computation step, the value of some variable l_i equals 1 indicates that the corresponding instruction l_i of M will be simulated, while all the others are equal to zero. Variables l'_1 and x'_1 are auxiliary variables. Variable x'_1 is used as input variable. Its initial value equals the input of register machine.

At the first computation step, programs $-x'_1 \to 1|x_1$ and $l'_1 \to 1|l_1$ are applied (other programs may also be applied, but have no effect, meaning that the application of these programs does not change any value of the variables). Program $-x'_1 \to 1|x_1$ changes the input value z of variable x'_1 to $-z$ and sends this value to variable x_1. Program $l'_1 \to 1|l_1$ increases the value of variable l_1 from 0 to 1, which will activate the programs simulating the instruction l_1 of register machine M.

For each ADD instruction l_i : (ADD$(r), k$) of M, there are two programs in system Π_M to simulate it in one step: $-l_i \to 1|x_r$ and $l_i \to 1|l_k$. When l_i equals to 1, program $-l_i \to 1|x_r$ decreases the value of x_r by 1, hence correctly simulating the action of increasing the value stored in register r; simultaneously, program $l_i \to 1|l_k$ changes the value of l_k from 0 to 1, which also correctly indicates the next instruction l_k of register machine M to be simulated. After the application of the two programs, l_i is reset to zero, hence the two programs will have no effect at the next step.

For each SUB instruction l_i : (SUB$(r), l_j, l_k$) of register machine M, there are four programs in system Π_M to simulate it in two steps:

$$l_i \to 1|l_k, \tag{4}$$

$$3(l_i + x_r)|_e \to 1|l'_j + 1|l_k + 1|x_r, \tag{5}$$

$$-2(2l_i + x_r)|_e \to 1|l'_j + 1|l_k, \tag{6}$$

$$-l'_j|_e \to 1|l_j. \tag{7}$$

At first we state that at any time t, when the value of variable l_i equals 0, programs (4) to (6) have no effect. To prove this statement we consider the following two cases.

Case 1. when $x_r(t) = 0$ (corresponding to the number stored in register r is 0), programs (5) and (6) cannot be applied due to the enzymatic control. Program (4) resets l_i to zero again, and sends 0 to l_k, hence no variable is changed.

Case 2. when $x_r(t) < 0$ (corresponding to the number stored in register r is greater than 0), all the three programs (4) to (6) are applied. Variables l'_j and l_k simultaneously get a contribution $x_r(t)$ from program (5) and a contribution $-x_r(t)$ from program (6). Hence the effect of the two programs (5) and (6) is to send a contribution 0 to variables l'_j and l_k. For variable x_r, it is reset to zero by programs (5) and (6) and simultaneously receives a contribution $x_r(t)$ from program (5), hence its value is not changed. Hence, in this case no variable is changed.

In conclusion, when $l_i = 0$, no matter the value of $x_r(t)$ is equal to 0 or less than 0, no variable is changed.

When $l_i = 1$, programs (4) to (7) start to simulate the instruction $l_i : (\mathrm{SUB}(r), l_j, l_k)$. When $x_r(t) = 0$ (corresponding to the number stored in register r is 0), only program (4) is applied, which sends a contribution 1 to variable l_k, hence correctly indicating the next instruction l_k of M to be simulated.

When $l_i = 1$ and $x_r(t) < 0$ (corresponding to the number stored in register r is greater than 0), programs (4) to (6) are applied. Program (5) resets variable x_r to zero and sends a contribution $x_r(t)+1$ to it, hence correctly simulating the instruction of decreasing the number stored in register r. Programs (5) and (6) send a contribution $x_r(t) + 1$ and $-(x_r(t) + 2)$ respectively to the two variables l'_j and l_k. Hence, the sum effect of the two programs (5) and (6) is sending a contribution -1 to variables l'_j and l_k. Simultaneously, variable l_k also gets a contribution 1 from program (4). Hence, the total contribution sent to variable l_k is 0. At the next step variable l'_j equal to -1 activates program (7), which increases variable l_j to 1, correctly indicating the next instruction l_j to be simulated.

After the simulation of each instruction of M, the value of variables x_i is equal to the negative value of the number stored in register i ($1 \leqslant i \leqslant m$), while only one of variables $l_1, \ldots, l_{n-1}$ is equal to 1, indicating the next instruction of M to be simulated. When register machine M reaches the halt instruction, the corresponding value of variable l_n is equal to 1. Since variable l_n is contained in no production function, no program can be applied to change the configuration, hence Π_M reaches a final configuration, thus the input value z is accepted.

5 Number of enzymatic variables in one-parallel ENP systems

In this section, we consider ENP systems working in another variant of parallel mode, the one-parallel mode.

Theorem 2. $N_{\mathrm{gen}}E_*NP_*(\mathrm{poly}^1(2), \mathrm{oneP}) = \mathrm{NRE}$.

Proof. The same as in the proof of Theorem 1, here we also only prove $N_{\mathrm{gen}}E_*NP_*(\mathrm{poly}^1(2), \mathrm{oneP}) \supseteq \mathrm{NRE}$.

Let $M = (m, H, l_1, l_n, I)$ be a non-deterministic register machine working in the generating mode. Different from the deterministic register machine, the ADD instructions are of the following forms: $l_i : (\mathrm{ADD}(r), l_j, l_k)$ (add 1 to register r and then go to one of the instructions with labels l_j, l_k, non-deterministically chosen).

The machine starts with all registers empty (i.e., storing the number zero). If the register machine reaches the halt instruction l_n, then the number stored at that time in the first register is said to be computed/generated by M. The set of all numbers computed by M is denoted by $N(M)$.

A specific ENP system Π_M working in the one-parallel mode is constructed to simulate the register machine M as follows:

$$\Pi_M = (n+1, H, \mu, (\mathrm{Var}_s, \mathrm{Pr}_s, \mathrm{Var}_s(0)), (\mathrm{Var}_1, \mathrm{Pr}_1, \mathrm{Var}_1(0)), \ldots, (\mathrm{Var}_n, \mathrm{Pr}_n, \mathrm{Var}_n(0)), V_{\mathrm{out}}),$$

where $H = \{s, 1, 2, \ldots, n\}$, $\mu = [\,[\,]_1[\,]_2 \cdots [\,]_n]_s$, $\mathrm{Var}_s = \{l_{i,s}, l'_{i,s} | l_i : (\mathrm{ADD}(r), l_j, l_k) \in I\} \cup \{l_{(j,i),s} \mid l_i : (\mathrm{SUB}(r), l_j, l_k) \in I, 1 \leqslant j \leqslant n\}$. $\mathrm{Var}_s(0)$ is the vector of initial values of all the variables in membrane s, obtained by setting all the variables with 0. $\mathrm{Pr}_s = \{-2nl_{i,s} \to \sum_{t=1}^{n}(1|x_{(r,t)} + 1|x'_{(r,t)}), 2l'_{i,s} \to 1|l_j + 1|l'_j, 2l'_{i,s} \to 1|l_k + 1|l'_k \mid \text{for } l_i : (\mathrm{ADD}(r), l_j, l_k)\} \cup \{2nl_{(j,i),s} \to 1|l_j + 1|l'_j + \sum_{t\neq i}(1|x_{(r,t)} + 1|x'_{(r,t)}), 2l_{(k,i),s} \to 1|l_k + 1|l'_k \mid \text{for } l_i : (\mathrm{SUB}(r), l_j, l_k) \in I\}$. $\mathrm{Var}_i = \{l_i, l'_i, x_{(j,i)}, x'_{(j,i)}, e_i \mid 1 \leqslant j \leqslant m\}$, $1 \leqslant i \leqslant n-1$. $\mathrm{Var}_i(0)$, $1 \leqslant i \leqslant n-1$, is the vector of initial values of all the variables in membrane i, obtained by

putting all the variables with 0 except of l_1 and l'_1 with 1. $\Pr_i = \{-2l_i \to 1|l_{(k,i),s}+1|l'_i, 5(l'_i+x_{(r,i)})|_{e_i} \to 1|l_{(j,i),s}+1|l_{(k,i),s}+1|l'_i+1|x_{(r,i)}+1|x'_{(r,i)}, -3x'_{(r,i)}|_{e_i} \to 1|l_{(j,i),s}+1|l_{(k,i),s}+l'_i \mid \text{for } l_i : (\mathrm{SUB}(r), l_j, l_k) \in I\}$ or $\Pr_i = \{l_i + l'_i \to 1|l_{i,s}+1|l'_{i,s} \mid \text{for } l_i : (\mathrm{ADD}(r), l_j, l_k) \in I\}$, $1 \leqslant i \leqslant n-1$. $\mathrm{Var}_n = \{l_n, l'_n, x_{(j,n)}, x'_{(j,n)} \mid 1 \leqslant j \leqslant m\}$. $\mathrm{Var}_n(0)$ is the vector of initial values of all the variables in membrane n, obtained by putting all the variables with 0. $\Pr_n = \{l_n \to 1|x'_{(1,n)}, -x_{(1,n)}|_{x'_{(1,n)}} \to 1|x_1\}$. $V_{\mathrm{out}} = \{x_1\}$.

For each label $l_i \in H$, $1 \leqslant i \leqslant n$, we associate a membrane i, all of which are placed in the skin membrane (labeled with s). Each membrane simulates the corresponding instruction in I. In each membrane i, there are two variables $x_{(j,i)}$ and $x'_{(j,i)}$ to save the value stored in register j of M, for $1 \leqslant j \leqslant m$. For each instruction label $l_i \in H$ there are two variables l_i and l'_i in in membrane i of Π_M. At the beginning of each computation step, the variables l_i and l'_i equal to 1 indicate that the corresponding instruction of M to be simulated is l_i. After simulating an ADD or SUB instruction, all variables are reset to zero, with the exception of variables $x_{(r,i)}$ and $x'_{(r,i)}$, and of one couple of variables l_i and l'_i.

When the programs in membrane i simulating the SUB or ADD instruction, all the values of variables $x_{(r,t)}$ and $x'_{(r,t)}$ in membrane t ($1 \leqslant t \leqslant n$) should be increased or decreased. But membranes 1 to n cannot directly communicate with each other. They communicate by means of auxiliary variables placed in the skin membrane.

The ADD instruction $l_i : (\mathrm{ADD}(r), l_j, l_k)$ of M is simulated in one step by the following four programs:

$$l_i + l'_i \to 1|l_{i,s} + 1|l'_{i,s}, \tag{8}$$

$$-2nl_{i,s} \to \sum_{t=1}^{n}(1|x_{(r,t)} + 1|x'_{(r,t)}), \tag{9}$$

$$2l'_{i,s} \to 1|l_j + 1|l'_j, \tag{10}$$

$$2l'_{i,s} \to 1|l_k + 1|l'_k. \tag{11}$$

When $l_i = l'_i = 0$, all the four programs (8) to (11) have no effect. When $l_i = l'_i = 1$, program (8) increases the values $l_{i,s}$ and $l'_{i,s}$ to 1. At the next step, program (9) sends a contribution -1 to variables $x_{(r,t)}$ and $x'_{(r,t)}$ ($1 \leqslant t \leqslant n$), hence correctly simulating the fact of decreasing the number stored in register r in membranes 1 to n. Program (10) passes the value of variable $l'_{i,s}$ to two variables l_j and l'_j, indicating the next instruction l_j of M to be simulated. Similarly, program (11) passes the value of variable $l'_{i,s}$ to two variables l_k and l'_k. Programs (10) and (11) sharing a common variable $l'_{i,s}$, only one of them can be non-deterministically chosen and applied, hence the system proceeds to simulate the instruction l_j or l_k at the next step.

The SUB instruction $l_i : (\mathrm{SUB}(r), l_j, l_k)$ of M is simulated in two steps by the following five programs:

$$-2l_i \to 1|l_{(k,i),s} + 1|l'_i, \tag{12}$$

$$5(l'_i + x_{(r,i)})|_{e_i} \to 1|l_{(j,i),s} + 1|l_{(k,i),s} + 1|l'_i + 1|x_{(r,i)} + 1|x'_{(r,i)}, \tag{13}$$

$$-3x'_{(r,i)}|_{e_i} \to 1|l_{(j,i),s} + 1|l_{(k,i),s} + 1|l'_i, \tag{14}$$

$$2nl_{(j,i),s} \to \sum_{t\neq i}(1|x_{(r,t)} + 1|x'_{(r,t)}) + 1|l_j + 1|l'_j, \tag{15}$$

$$2l_{(k,i),s} \to 1|l_k + 1|l'_k. \tag{16}$$

When $l_i = l'_i = 0$, these five programs (12) to (16) have no effect. Program (12) resets l_i to zero and sends a contribution 0 to variables $l_{(k,i),s}$ and l'_i, hence no variable is changed. For programs (13) and (14), only when the values of variables $x_{(r,i)}$ and $x'_{(r,i)}$ are negative, they can be applied. In this case, for $l_i(t) = l'_i(t) = 0$, the contributions sent to variables $l_{(j,i),s}$, $l_{(k,i),s}$ and l'_i by programs (13) and (14) are $x_{(r,i)}(t)$ and $-x'_{(r,i)}(t)$ respectively. For variables $x_{(r,i)}$ and $x'_{(r,i)}$ equal to each other, hence for variables $l_{(j,i),s}$, $l_{(k,i),s}$ and l'_i the sum of the contributions received from programs (13) and (14) is 0. Variables $x_{(r,i)}$ and $x'_{(r,i)}$ are first reset to zero and then receive contribution $x_{(r,i)}(t)$ from program (13), hence their values remain unchanged. In conclusion, when $l_i = l'_i = 0$, no variable is changed.

If at some time t, $l_i = l'_i = 1$ and $x_{(r,i)}(t) = x'_{(r,i)}(t) = 0$ (corresponding to the fact that the number stored in register r is 0), then programs (13) and (14) cannot be applied. Program (12) resets variable

l_i to zero and sends a contribution -1 to variables $l_{(k,i),s}$ and l'_i (hence the value of $l_{(k,i),s}$ is changed to -1, and l'_i is changed to 0). At the next step, program (16) in the skin membrane increases the variables l_k and l'_k to 1, which means that the instruction l_k of M will be simulated.

If at some time t, $l_i = l'_i = 1$ and $x_{(r,i)}(t) = x'_{(r,i)}(t) < 0$ (corresponding to the fact that the number stored in register r is greater than 0), then all the three programs (12) to (14) in membrane i are applied. For variables $l_{(k,i),s}$ and l'_i, the contributions that they receive from programs (12) to (14) are -1, $x_{(r,i)}(t)+1$ and $-x'_{(r,i)}(t)$ respectively, the sum of which is zero. Variables $x_{(r,i)}$ and $x'_{(r,i)}$ are first reset to zero by programs (13) and (14), then they receive a contribution $x_{(r,i)}(t)+1$ from program (13), hence their final values are $x_{(r,i)}(t)+1$, correctly simulating that the value of register r of M is decreased by one. Variable $l_{(j,i),s}$ gets a contribution $x_{(r,i)}(t)+1$ from program (13) and $-x'_{(r,i)}(t)$ from program (14), hence its final value is 1. At the next step, program (15) in the skin membrane is applied, which passes the value of $l_{(j,i),s}$ to variables $x_{(r,t)}$ and $x'_{(r,t)}$ in membrane t, for $1 \leqslant t \leqslant n, t \neq i$, hence simulating that the value stored in register r is decreased by one. Simultaneously, this program also increases the value of variables l_j and l'_j to 1, which correctly indicates the next instruction l_j of M to be simulated at the next step.

When the register machine halts, it reaches the halting instruction l_n, the corresponding variables l_n and l'_n in Π_M are equal to 1. Suppose the value stored in register 1 is number K, which is the number generated by M for this computation. Then the values of variable $x_{(1,n)}$ and $x'_{(1,n)}$ are $-K$. At the next step, program $l_n \to 1|x'_{(1,n)}$ increases the value of $x'_{(1,n)}$ by 1, hence its value is greater than $x_{(1,n)}$, which activates program $-x_{(1,n)}|_{x'_{(1,n)}} \to 1|x_1$. This program resets the variable $x_{(1,n)}$ to zero and passes the value K to variable x_1. After that, no program can be applied to change the configuration of Π_M. The system halts. The number K stored in variable x_1 is the number generated by the system Π_M.

The results of Theorem 2 can be improved by using Theorem 1 given in [31]. Here, we restate the theorem as a lemma.

Lemma 1 ([31]). Let Π be any computing (or generating, or accepting) ENP system of degree $m \geqslant 1$, working in the all-parallel or in the one-parallel mode. Then there exists an ENP system Π' of degree 1 that computes (respectively, generates, accepts) the same function (respectively, family of sets) using the same rule application mode.

In what follows, we will improve Theorem 2 to the following Theorem.

Theorem 3. $N_{\text{gen}}E_2NP_1(\text{poly}^1(2), \text{oneP}) = \text{NRE}$.

Proof. At first, according to the flattening technique introduced in the Theorem 1 of [31], for ENP system Π of degree $m \geqslant 1$ as a generating device working in the one-parallel mode, there exists an ENP system Π_0 with one membrane that generates the same family of sets working also in the one-parallel mode. In fact, according to Theorem 1 in [31], a new ENP system Π_0 with one membrane can be constructed by putting all the variables and all the programs of Π_M — keeping both indexes, also in the variables occurring in programs — in the membrane of Π_0. Then Π_0 generates the same family of sets working also in the one-parallel mode. Hence, the result of Theorem 2 can be improved to $N_{\text{gen}}E_*NP_1(\text{poly}^1(2), \text{oneP}) = \text{NRE}$.

Secondly, the enzymatic variables e_i in membrane i of Π_M, $1 \leqslant i \leqslant n-1$ are constant and all equal to 0. Then in the system Π_0 these enzymatic variables can be replaced by one variable e_0 with value 0. That enzymatic variable is used to control the application of programs by comparing the value of enzymatic variable with the variables involved in the production function, the above modification only changes the alphabet not its value, hence these programs with enzymatic variable e_0 operate in the same way as they with enzymatic variable e_i. Except of the constant enzymatic variable e_0, there is also a non-constant enzymatic variable $x'_{(1,n)}$. The number of the enzymatic variables used in the system Π_0 is 2. So, this theorem holds true.

Note that the result stated in Theorem 2 looks the same with the result given in Theorem 3 of [33] presented as (2). The difference is that the values of enzymatic variables are constant in this work; however the enzymatic variables are not constant in Theorem 3 of [33].

For one-parallel ENP systems working in the accepting mode, one enzymatic variable suffices to reach

universality.

Theorem 4. $N_{\rm acc}E_1NP_1(\text{poly}^1(2), \text{oneP}) = \text{NRE}$.

Proof. Let $M = (m, H, l_1, l_n, I)$ be a deterministic register machine working in the accepting mode as specified in the proof of Theorem 1 of Section 4. Number c is the value introduced in the register 1. The computation starts with instruction l_1 and, if it halts, then the number c is accepted.

Now, we construct an ENP system Π'_M to simulate M by modifying the system Π_0 in the Theorem 3. The variables and programs of Π_0 used to simulate the ADD and SUB instructions are kept in Π'_M. For the system Π'_M working in the accepting mode, there is no need to change the negative value stored in $x_{(1,n)}$ to the positive one. Hence programs $l_n \to 1|x'_{(1,n)}$ and $-x_{(1,n)}|_{x'_{(1,n)}} \to 1|x_1$ in Π_0 are no longer kept in Π'_M. Thus, the number of the enzymatic variables used in Π'_M is 1. Simultaneously, we add a variable x_{11} as the input variable. Its initial value equals to the input value c stored in register 1. Two programs $-2(n-1)x_{11} \to \sum_{t=1}^{n-1}(1|x_{(1,t)} + 1|x'_{(1,t)})$ and $2l_0 \to 1|l_1 + 1|l'_1$ are used to initialize the computation. The former program changes the input value c of variable x_{11} to $-c$ and sends it to all the variables $x_{(1,t)}$ and $x'_{(1,t)}$ for $1 \leqslant t \leqslant n$. The second program increases the values of variables l_1 and l'_1 to 1, which will activate the programs simulating the instruction l_1. When the register machine halts, the value of variable l_n in Π'_M equals to 1. Since variable l_n is contained in no production function, no program can be applied to change the configuration, hence Π_M reaches a final configuration, the input value c is accepted.

6 Conclusion and discussion

In this work, we optimized universal ENP systems in terms of the number of enzymatic variables. Specifically, it was proved that for all-parallel or one-parallel ENP systems working in the accepting mode, one membrane suffices to reach universality. But for the universal one-parallel ENP systems working in the generating mode, two enzymatic variables are needed. We list the main results as follows (the left column is the results obtained in this work and the right column is the results given in [31]):

$$\begin{aligned}\text{NRE} &= N_{\rm acc}E_1NP_1(\text{poly}^1(2), \text{allP}) = N_{\rm acc}E_{13}NP_1(\text{poly}^1(1), \text{allP})\\ &= N_{\rm acc}E_1NP_1(\text{poly}^1(2), \text{oneP}) = N_{\rm acc}E_{52}NP_1(\text{poly}^1(1), \text{oneP})\\ &= N_{\rm gen}E_2NP_1(\text{poly}^1(2), \text{oneP}) = N_{\rm acc}E_{52}NP_1(\text{poly}^1(1), \text{oneP}).\end{aligned}$$

Constructing a small universal ENP system with all the complexity parameters optimized need further investigations. In [31], five classes of complexity parameters have been investigated: number of membranes, degrees of polynomials, number of variables used in the production function, number of programs and number of variables used in the system. Here, we considered the sixth complexity parameter, number of enzymatic variable. In the universal system constructed in this paper, some of the parameters considered in [31] are not increased, such as number of membranes and degrees of the polynomial; while others are increased, such as the number of the variables used in the production function, and the number of programs and variables used in the system.

Recently, several variants of numerical P systems have been proposed, such as numerical P systems with thresholds [36], numerical P systems with migrating variables [37]. It has been proved that both the strategy of using thresholds and the feature of migrating variables can improve the computational power of numerical P systems. A natural question is to optimize the number of programs with threshold for numerical P systems with thresholds reaching universality. For universal numerical P systems with migrating variables, it remains open whether the number of enzymatic variable can be optimized to zero.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61772214, 61320106005, 61033003, 61472154) and Innovation Scientists and Technicians Troop Construction Projects of Henan Province (Grant No. 154200510012). The work of Andrei Păun was supported by UEFSCDI Project RemoteForest Project (Grant No. PN-II-PTPCCA-2011-3.2-1710). This paper was written during a three-months stay of Zhiqiang Zhang and Tingfang Wu in Curtea de Argeş, Romania, in the fall of 2015.

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Rozenberg G, Bäck T, Kok J N. Handbook of Natural Computing. Berlin: Springer, 2012
2 Miller W T, Werbos P J, Sutton R S. Neural Networks for Control. Cambridge: MIT Press, 1995
3 Păun G. Computing with membranes. J Comput Syst Sci, 2000, 61: 108–143
4 Păun G, Rozenberg G, Salomaa A. The Oxford Handbook of Membrane Computing. New York: Oxford University Press, 2010
5 Păun G. Membrane Computing: An Introduction. Berlin: Springer, 2012
6 Ciobanu G, Păun G, Pérez-Jiménez M J. Applications of Membrane Computing. Berlin: Springer, 2006
7 Frisco P, Gheorghe M, Pérez-Jiménez M J. Applications of Membrane Computing in Systems and Synthetic Biology. Berlin: Springer, 2014
8 Alhazov A, Sburlan D. Static sorting P systems. In: Applications of Membrane Computing. Berlin: Springer, 2006. 215–252
9 Nishida T Y. An approximate algorithm for NP-complete optimization problems exploiting P systems. In: Proceedings of the 1st Brainstorming Workshop on Uncertainty in Membrane Computing, Palma de Mallorca, 2004. 185–192
10 Nishida T Y. Membrane algorithms: approximate algorithms for NP-complete optimization problems. In: Applications of Membrane Computing. Berlin: Springer, 2006. 303–314
11 Enguix G B. Unstable P systems: applications to linguistics. Lect Notes Comput Sci, 2005, 3365: 190–209
12 Michel O, Jacquemard F. An analysis of a public key protocol with membranes. In: Applications of Membrane Computing. Berlin: Springer, 2006. 283–302
13 Dinneen M J, Kim Y B, Nicolescu R. P systems and the byzantine agreement. J Logic Algebr Program, 2010, 79: 334–349
14 Colomer M À, Montori A, García E, et al. Using a bioinspired model to determine the extinction risk of calotriton asper populations as a result of an increase in extreme rainfall in a scenario of climatic change. Ecol Model, 2014, 281: 1–14
15 Margalida A, Colomer M À. Modelling the effects of sanitary policies on european vulture conservation. Sci Rep, 2012, 2: 753
16 Ionescu M, Păun G, Yokomori T. Spiking neural P systems. Fund Inform, 2006, 71: 279–308
17 Peng H, Wang J, Pérez-Jiménez M J, et al. Fuzzy reasoning spiking neural P system for fault diagnosis. Inform Sci, 2013, 235: 106–116
18 Zeng X, Song T, Zhang X, et al. Performing four basic arithmetic operations with spiking neural P systems. IEEE Trans Nanobiosci, 2012, 11: 366–374
19 Zhang G, Cheng J, Wang T, et al. Membrane Computing: Theory & Applications (in Chinese). Beijing: Science Press, 2015
20 Zhang G, Rong H, Neri F, et al. An optimization spiking neural P system for approximately solving combinatorial optimization problems. Int J Neural Syst, 2014, 24: 1–16
21 Păun G, Păun R. Membrane computing and economics: numerical P systems. Fund Inform, 2006, 73: 213–227
22 Rivera D L, Naranjo M Á G. The pole balancing problem with enzymatic numerical P systems. In: Proceedings of the 13th Brainstorming Week on Membrane Computing, Sevilla, 2015. 195–206
23 Buiu C, Vasile C, Arsene O. Development of membrane controllers for mobile robots. Inform Sci, 2012, 187: 33–51
24 Wang X, Zhang G, Neri F, et al. Design and implementation of membrane controllers for trajectory tracking of nonholonomic wheeled mobile robots. Integr Comput Aid Eng, 2015, 23: 15–30
25 Arsene O, Buiu C, Popescu N. Snups—a simulator for numerical membrane computing. Int J Innov Comput Inform Control, 2011, 7: 3509–3522
26 Buiu C, Arsene O, Cipu C, et al. A software tool for modeling and simulation of numerical P systems. BioSystems, 2011, 103: 442–447
27 Pavel A B. Membrane controllers for cognitive robots. Dissertation for M.S. Degree. Bucharest: University of Bucharest, 2011
28 García-Quismondo M, Pavel A B, Pérez-Jiménez M d J, et al. Simulating large-scale ENPS models by means of GPU. In: Proceedings of the 10th Brainstorming Week on Membrane Computing, Sevilla, 2012. 137–152
29 Pavel A, Arsene O, Buiu C. Enzymatic numerical P systems—a new class of membrane computing systems. In:

Proceedings of IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), Liverpool, 2010. 1331–1336

30 Leporati A, Mauri G, Porreca A E, et al, Enzymatic numerical P systems using elementary arithmetic operations. In: Proceedings of the 14th International Conference on Membrane Computing, Moldova, 2016. 249–264

31 Leporati A, Porreca A E, Zandron C, et al. Improved universality results for parallel enzymatic numerical P systems. Int J Unconv Comput, 2013, 9: 385–404

32 Vasile C I, Pavel A B, Dumitrache I. Universality of enzymatic numerical P systems. Int J Comput Math, 2013, 90: 869–879

33 Vasile C I, Pavel A B, Dumitrache I, et al. On the power of enzymatic numerical P systems. Acta Inform, 2012, 49: 395–412

34 Păun G. Some open problems about catalytic, numerical, and spiking neural P systems. In: Proceedings of Revised Selected Papers of the 14th International Conference on Membrane Computing, Chişinău, 2013. 33–39

35 Minsky M L. Computation: Finite and Infinite Machines. Englewood Cliffs: Prentice-Hall, 1967

36 Zhang Z, Pan L. Numerical P systems with thresholds. Int J Comput Commun Control, 2016, 11: 292–304

37 Zhang Z, Wu T, Păun A, et al. Numerical P systems with migrating variables. Theor Comput Sci, 2016, 641: 85–108