A membrane computing approach to the generalized Nash equilibrium

Citation for the original published paper (version of record):
Luque Cerpa, A., Gutiérrez-Naranjo, M. (2025). A membrane computing approach to the generalized Nash equilibrium. Natural Computing, 24(2): 321-336.
http://dx.doi.org/10.1007/s11047-025-10014-z

Natural Computing (2025) 24:321–336
https://doi.org/10.1007/s11047-025-10014-z

A membrane computing approach to the generalized Nash equilibrium

Alejandro Luque-Cerpa[1] · Miguel Á. Gutiérrez-Naranjo[2]

Accepted: 17 March 2025 / Published online: 18 April 2025

Abstract
Generalized Nash Equilibrium is an extended version of the standard Nash Equilibrium with important implications in real-life problems such as economics, wireless communication, the electricity market, or engineering among other areas. In this paper, we propose a first approach to computing Generalized Nash Equilibria using Membrane Computing techniques. We model an efficient P system that, based on Euler's method, computes approximations of Generalized Nash Equilibria of population games under Brown–von Neumann–Nash dynamics, bridging both areas and opening a door for a flow of problems and solutions in both directions.

Keywords Membrane computing · Generalized Nash equilibrium · Evolutionary game theory

1 Introduction

Evolutionary Game Theory (EGT) studies the evolution of a population of agents that interact with each other and get a payoff in each interaction Hofbauer and Sigmund (2000). The obtained payoff depends on the chosen strategies of the agents which participate in the interaction. Each agent selects only one strategy at a time, but this choice can be modified over time. The driving principle in this situation is that individuals tend to be selfish, choosing strategies that result in higher payoffs for themselves. In this context, a Nash equilibrium is reached when no agent can increase its payoff by changing its strategy while other agents maintain their current ones Nash (1951).

In a Nash equilibrium problem, all the agents compete among them to maximize their payoffs, and each agent can freely choose its strategy. The generalized Nash equilibrium problem (GNEP) is a variant of the Nash problem introduced in 1952 by G. Debreu Debreu (1952). In a GNEP, the strategy set of each player may also depend on the other players' strategies. This GNEP models a large number of real-life situations, such as power allocation in a telecommunication system, environmental pollution control, or energy market model (for a detailed survey, see, e.g., Facchinei and Kanzow (2007)).

In this paper, we propose to study the GNEP in the framework of Membrane Computing Păun (2002); Păun et al. (2010). Membrane Computing is a well-known area of Computer Science that takes inspiration from the biochemical reactions inside the vesicles of living cells. P systems Păun (2000), the so-called Membrane Computing devices, have been successfully considered to model many dynamic processes in real-life problems Colomer et al. (2011, 2010); García-Quismondo et al. (2017). From the initial definition of P systems, many variants have been explored by adding new features to the initial model (see, e.g., Song (2021) for a recent survey). Recently, Probabilistic P systems Cardona et al. (2011), a kind of P system designed to deal with probability distributions in the application of rules, was considered to model the spread of behaviors in structured populations in the framework of EGT García-Victoria et al. (2022). In this paper, we study the GNEP by considering transition P systems with active membranes Păun (2001), also called membrane polarization.

✉ Alejandro Luque-Cerpa
luque@chalmers.se

Miguel Á. Gutiérrez-Naranjo
magutier@us.es

[1] Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden

[2] Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, Seville, Spain

The paper is organized as follows: Sect. 2 establishes some background on P systems and specifies the type of P system we use: transition P systems with membrane polarization. Section 3 introduces population games under the Brown-von Neumann-Nash (BNN) dynamics, that will be used as the framework to define our P system. In Sect. 4, we describe the design of our P system and analyze its time complexity, showing that it does not depend on the number of players or strategies involved. We also present an experiment to illustrate its functioning. Finally, some conclusions and hints for future work are presented.

2 Transition P systems with membrane polarization

In this section, we define the variant of P systems that we used to solve our problem: transition P systems with membrane polarization. Then, in section 2.1, we provide an example of such a P system.

After the development of the first model of P system by Gh. Păun in 1998 Păun (2000), many variations have been presented. In this work, we use a combination of two proposed variants. The P system designed is a transition P system Păun (2000) with active membranes Păun (2001) without division rules, i.e., a transition P system with membrane polarization. A transition P system with membrane polarization of degree $q \geq 1$ is a construct:

$$\Pi = \langle \Gamma, \mu, w_1, \ldots, w_q, (\mathcal{R}_1, \rho_1), \ldots, (\mathcal{R}_q, \rho_q), i_{out} \rangle$$

where:

1. Γ is the alphabet of *objects*;
2. μ is a hierarchical tree-like *membrane structure* of q membranes that have a polarization among $0, +, -$;
3. $w_1, \ldots, w_q$ are multisets of objects over Γ;
4. $\mathcal{R}_1, \ldots, \mathcal{R}_q$ are finite sets of *evolution rules* of the form:
   - $u[v]_h^\alpha \rightarrow u'[v']_h^\beta$ where u, u', v, v' are multisets over Γ, $h \in \{1, \ldots, q\}$, h is not the label of the root membrane in μ, and $\alpha, \beta \in \{0, +, -\}$.
   - $[v]_h^\alpha \rightarrow u'[v']_h^\beta$ where u, u', v, v' are multisets over Γ, $h \in \{1, \ldots, q\}$, h is the label of the root membrane in μ, and $\alpha, \beta \in \{0, +, -\}$.

   The difference between the expressions resides in that no objects should be able to enter the *skin* membrane (the root of μ) from the environment. The meaning of these rules can be easily understood as combinations of the following examples:
   - $[u]_h^\alpha \rightarrow [v]_h^\alpha$, also expressed as $[u \rightarrow v]_h^\alpha$, is an object evolution rule, that transforms the multiset u into the multiset v.
   - $[u]_h^\alpha \rightarrow v[\ ]_h^\beta$ is a send-out communication rule, that ejects the multiset u, and transforms it into the multiset v.
   - $u[\ ]_h^\alpha \rightarrow [v]_h^\beta$ is a send-in communication rule, that absorbs the multiset u, and transforms it into the multiset v.

   The general expression considers combinations of these cases, where some multisets can be absorbed into membrane h at the same time as others are transformed or ejected.
5. $\rho_1, \ldots, \rho_q$ are partial order relations over $\mathcal{R}_1, \ldots, \mathcal{R}_q$, called *priority* relations. Given two rules r, r', we represent that r has higher priority than r' by $\rho_r > \rho_{r'}$. Priority indicates what rule should be applied if both are applicable.
6. $i_{out} \in \{0, 1, \ldots, q\}$ is the output region, where 0 represents the environment.

A configuration of Π is defined by $C_t = ((w_{1,t}, \alpha_{1,t}), \ldots, (w_{1,t}, \alpha_{1,t}), w_{0,t})$ for an instant t, where $w_{h,t}$ is the multiset of objects in membrane h at instant t, $\alpha_{h,t}$ is the membrane polarization of membrane h, and $w_{0,t}$ is the multiset of objects of the environment. The initial configuration of Π is $C_0 = ((w_1, 0), \ldots, (w_q, 0), \emptyset)$. We use the notation $\mathbb{C}_t = \mu'$ to denote specific parts from the configuration C_t where only the membranes in the subtree μ' from μ are considered.

For each configuration, the rules are applied in a parallel and maximal way. By maximal, we indicate that no more rules can be applied at the same time. Formally, a multiset U of rules is maximal if there is no multiset of applicable rules U' such that $U \subset U'$. If two applicable rules with the same priority are exclusive, this is, triggering one would prevent the other one from triggering, then only one of them is selected at random and applied.

As in García-Victoria et al. (2022), the semantics of the P system follow the next principles:

(I1) When an object crosses a membrane, its polarization may change. Rules can only be applied if the polarization is appropriate.

(I2) If two rules that affect the same membrane can be applied at the same time, and one of the rules changes the polarization of the membrane, both rules are applied. This means that the change of polarization is performed *after* all other evolution rules are applied.

Notice that principle (I2) ensures that rules are applied in a parallel and maximal way. If this principle is not assumed, and a rule can change the membrane polarization, then the order of application of the rules would be important during a single transition step. In that case, multisets of rules that

can be applied would not be well-defined, breaking the parallelism and maximality of the system.

Notice also that, while the membranes in Π have labels in $\{1,\ldots,q\}$, we can always define a set H of labels with $|H| = q$ such that there is a bijection between the elements of H and the membranes of μ. The same applies to the rules, that we can express as the tuple $(\mathcal{R}, \rho)$. We use this fact in Sect. 4.1 to provide a better indexing.

2.1 Example of a P system

Let

$$\Pi = \langle \Gamma, \mu, w_1, w_2, (\mathcal{R}_1, \{\rho_{r_1} > \rho_{r_2}\}), (\mathcal{R}_2, \emptyset), i_{out} \rangle$$

be a transition P system with membrane polarization of degree 2 where:

- $\Gamma = \{a, b, c\}$;
- $\mu = [\ [\ ]_2^0\ ]_1^0$;
- $w_1 = \{k_0\}$;
- $w_2 = \{a^3 c\}$;
- $\mathcal{R}_1 = \{r_1 \equiv k_0[\ ]_2^0 \to [k]_2^+, r_2 \equiv [k_0 \to k]_1^0\}$;
- $\mathcal{R}_2 = \{r_3 \equiv [a^2 \to b]_2^0, r_4 \equiv [a \to c]_2^0, r_5 \equiv [c \to b]_2^0\}$;
- $i_{out} = 2$.

We have that $U_1 = \{r_1, r_3, r_4, r_5\}$, $U_2 = \{r_3\}$, $U_3 = \{r_1, r_4^3, r_5\}$ are multisets of applicable rules. U_2 is not maximal because $U_2 \subset U_1$. U_1 and U_3 are the only maximal multisets of applicable rules, and they could both be applied in this configuration because there is no priority between the rules involved in each set. r_2 can not be part of a maximal multiset of applicable rules because rule r_1 has priority over rule r_2 ($\rho_{r_1} > \rho_{r_2}$).

The multiset U_1 would lead to the configuration $\mathbb{C}_1 = [\ [b^2 ck]_2^+]_1^0$, and the multiset U_3 would lead to $\mathbb{C}_1 = [\ [bc^3 k]_2^+]_1^0$. Because the polarization of membrane 2 changed to $+$ in both configurations, none of the rules in $\mathcal{R}_2$ can now be applied. Because there are no objects k_0 in membrane 1, none of the rules in $\mathcal{R}_1$ can be applied. The computation of the system is then finished for both cases after one transition step.

3 Population games under BNN dynamics

The purpose of this section is to introduce population games. Specifically, we introduce population games under BNN dynamics, which are central to this paper. In section 3.1, we give an example of such a game: the Energy Market Game, where players decide when to buy energy and modify their strategies depending on the decisions of the rest until an equilibrium is reached. We use the Energy Market Game as a framework to define our P system, and we explain how the P system can be modified to adapt it to other population games under BNN dynamics.

In a population game, we have a society of decision-making agents divided into disjoint populations that receive different payoffs depending both on the decisions they make and the decisions the rest of the agents make. The goal of each population is to maximize the payoff received. The decisions that agents can make depend on the population they form part of. Each agent is endowed with a revision protocol, which provides conditional switch rates between strategies according to their associated payoffs Sandholm (2010). These rates allow the agents to change their strategies over time. When the number of agents is large enough, this process can be described by differential equations, referred to as the evolutionary dynamics model (EDM). In EDMs, the agents can be modeled as real numbers, the *mass* of agents, instead of being modeled as discrete independent entities. There are multiple EDMs, but we focus on a specific EDM known as BNN dynamics Brown and von Neumann (1951), which are described next (see Martinez-Piazuelo et al. (2022) for details).

Let us consider a society of agents divided into $N \in \mathbb{Z}_{\geq 1}$ disjoint populations indexed by $\mathcal{P} = \{1, 2, ..., N\}$. Each population $k \in \mathcal{P}$ is comprised of a constant mass of decision-making agents $m^k \in \mathbb{R}_{>0}$. The set of strategies of each agent in population $k \in \mathcal{P}$ is $\mathcal{S}^k \subset \mathbb{Z}_{\geq 1}$ with $2 \leq n^k = |\mathcal{S}^k| < \infty$. The amount of agents selecting strategy $i \in \mathcal{S}^k$ at population k is denoted as $x_i^k \in \mathbb{R}_{\geq 0}$. Notice that agents from different populations k_1 and k_2 can select the same strategy i if $i \in S^{k_1}$ and $i \in S^{k_2}$. Similarly, the proportion of agents selecting strategy $i \in \mathcal{S}^k$ at population k is denoted as $z_i^k = x_i^k / m^k$. Furthermore, $x^k = (x_{i_1}^k, ..., x_{i_{n^k}}^k)^\top$ and $z^k = (z_{i_1}^k, ..., z_{i_{n^k}}^k)^\top$ denote the strategic distributions of population k, $x = (x^{1\top}, x^{2\top}, ..., x^{N\top})^\top$, and $z = (z^{1\top}, z^{2\top}, ..., z^{N\top})^\top$. Let $t \in \mathbb{Z}_{\geq 0}$ be the discrete-time index; $x(t)$ the value of x at time t; $z(t)$ the value of z at time t and $p_i^k(t) \in \mathbb{R}$ the payoff received by the agents selecting strategy $i \in \mathcal{S}^k$ at population $k \in \mathcal{P}$.

Following the revision protocol introduced in Martinez-Piazuelo et al. (2022), the equations that define the EDM describing the evolution of $x(t)$ over time are:

$$\dot{x}_i^k(t) = m^k [\hat{p}_i^k(t)]_+ - x_i^k(t) \sum_{j \in \mathcal{S}^k} [\hat{p}_j^k(t)]_+ \tag{1}$$

$$\hat{p}_j^k(t) = p_j^k(t) - \frac{1}{m^k} \sum_{l \in \mathcal{S}^k} x_l^k(t) \cdot p_l^k(t) \tag{2}$$

where $[\cdot]_+ = \max(\cdot, 0)$, and $\dot{x}$ denotes the derivative of x. This EDM is known as the BNN dynamics Brown and von Neumann (1951).

Intuitively, Eq. (2) computes the benefit of having more agents following strategy j in population k. The reason is that Equation (2) computes the difference between the payoff obtained by agents x_j^k and the average payoff obtained at time step t. A positive value of $\hat{p}_j^k(t)$ would indicate that it would be better for population k to have agents switch to strategy j. Equation (1) can then be used to decide how many agents should switch to other strategies at the next time step, given by the derivatives $\dot{x}_i^k(t)$.

A payoff dynamics model (PDM) that describes the evolution of $p(t)$ is also introduced, defined by:

$$\dot{\mu}(t) = Ax(t) - b \tag{3}$$

$$p(t) = f(x(t)) - A^\top \mu(t) \tag{4}$$

where f is a fitness function that provides the payoff for the strategies chosen at a given population state, μ represents some constraints over such decisions, and such constraints are given by a matrix A and a vector b that depends on the specific problem. In this context, A and b just introduce penalizations to the payoff, instead of introducing hard constraints.

Since the importance of this system lies in updating the payoff signal $p(t)$ and having a closed-loop configuration between $p(t)$ and $x(t)$, a simplified version of this system, where we remove the constraints over the strategies chosen, is considered:

$$p(t) = f(x(t)) \tag{5}$$

An EDM whose payoff function follows a PDM is then called an EDM-PDM.

It is not hard to modify the P system proposed later in Sect. 4.1 to compute the effect of the constraints introduced by A and b in Eq. (5). Because they are linear, the extra computation time required is constant per iteration t. However, because our goal is to show how can P systems be used for computing Generalized Nash Equilibria (GNE), we limit ourselves to the case without constraints.

3.1 Energy market game

In the previous section, we expressed a population game under BNN dynamics through Eqs. (1) (2) and (5). To solve a specific population game, we need to define the payoff function (Eq. 5) considering a specific function f that is different for each game. Taking this into account, a specific EDM must be selected as a framework to define our P system. Because of this, we consider an example of the Energy Market Game Martinez-Piazuelo et al. (2022) as the framework.

In the Energy Market Game, $N \in \mathbb{Z}_{\geq 1}$ players compete to purchase energy over a time horizon of $T \in \mathbb{Z}_{\geq 1}$ time slots. Players who try to purchase energy in the same time slots will end up paying more for the energy, and the base price for energy is higher for some time slots than for others. The goal is to buy energy as cheaply as possible, considering that other players have the same goal. For this problem, we can consider each player a population k, and the agents x_i^k represent the amount of energy purchased by player k in the time slot i.

Following the notation described at the beginning of Sect. 3, let $C^k \in \mathbb{R}^{T \times n^k}$ be a matrix such that each column of C^k has exactly one element equal to 1 and the rest equal to 0, each row of C^k has at most one element equal to 1, and the j-th element of the i-th column of C^k is 1 iff player k competes in time slot $j \leq T$. Let $C = [C^1, C^2, \ldots, C^N] \in \mathbb{R}^{T \times n}$ be the concatenation of the C^k matrices of all players, where $n = \sum_{k \in \mathcal{P}} n^k$. Then Cx corresponds to the collective energy demand for all time slots. Let $J : \mathbb{R}^n \to \mathbb{R}^T$ be the pricing function given by $J(x) = DCx + \bar{J}$, where $D \in \mathbb{R}^{T \times T}_{\geq 0}$ is diagonal and $\bar{J} \in \mathbb{R}^T_{\geq 0}$, and let $Q^k : \mathbb{R}^{n^k}_{\geq 0} \to \mathbb{R}$ be the individual cost of each player $k \in \mathcal{P}$, given by $Q^k(x^k) = \sum_{i \in \mathcal{S}^k} \left((\alpha_i^k/2)(x_i^k)^2 + \beta_i^k \cdot x_i^k \right)$, where $\alpha_i^k \in \mathbb{R}_{\geq 0}$ and $\beta_i^k \in \mathbb{R}_{\geq 0}$.

Following the results from Martinez-Piazuelo et al. (2022), the payoff function $p(t) = f(x(t))$ for the Energy Market Game can be expressed by $f(x(t)) = -S \cdot x(t) - C^\top \bar{J} - \alpha \odot x(t) - \beta$ where

- $M = \text{diag}(m^1 \mathbf{I}_{n^1}, m^2 \mathbf{I}_{n^2}, \ldots, m^N \mathbf{I}_{n^N})$,
- $S = \text{diag}(C^{1\top} D C^1, \ldots, C^{N\top} D C^N) + R^\top R$, and
- $R = [\sqrt{D}C^1, \sqrt{D}C^2, \ldots, \sqrt{D}C^N]$.
- diag is the operation that constructs a matrix using the input elements as the diagonal, and where the rest of the elements are null.

To define the payoff over $z_i^k(t)$, the following transformation is performed over Eq. (5):

$$\begin{aligned} p(t) &= f(x(t)) = f(M \cdot z(t)) \\ &= -S \cdot M \cdot z(t) - C^\top \bar{J} - \alpha \odot (M \cdot z(t)) - \beta \end{aligned} \tag{6}$$

Equations (1) and (2) also change for $z_i^k(t)$:

$$\dot{z}_i^k(t) = \frac{\dot{x}_i^k(t)}{m^k} = [\hat{p}_i^k(t)]_+ - z_i^k(t) \cdot \sum_{j \in \mathcal{S}^k} [\hat{p}_j^k(t)]_+ \tag{7}$$

$$\hat{p}_j^k(t) = p_j^k(t) - \sum_{l \in \mathcal{S}^k} z_l^k(t) \cdot p_l^k(t) \tag{8}$$

4 Design and functioning of the P system

In this section, we introduce first the general idea behind the design of a P system proposed to compute approximations of GNE for population games under BNN dynamics. Then, we define the P system in Sect. 4.1. After that, we perform a computation analysis in Sect. 4.2, where we indicate the evolution rules defined in Sect. 4.1 that are applied to the configurations. Finally, we include our experimental results in Sect. 4.3.

Let us consider the EDM-PDM system introduced in Sect. 3.1 by Eqs. (6), (7), and (8). In this section, a P system that computes approximations of GNE under the BNN dynamics for this system is described. The computation can be summarized in a loop of five stages, represented in Algorithm 1. Stages 1 and 2 are used to compute $\hat{p}(t)$ using Eq. (8), Stages 3 and 4 are used to compute $\dot{z}(t)$ using Eq. (7), and Stage 5 is used to update the value of $z(t)$. To solve any other EDM-PDM system, only the first stage of the P system has to be modified, while the rest remains unchanged.

The fundamental idea behind the system is to compute approximations and discretize the values involved in the EDM-PDM system by rounding to n decimals and multiplying by 10^n. To show the functioning of our P system, we fix $n = 2$ from now on. However, the system can easily be modified for other values of n, providing better approximations with the cost of a longer runtime. After discretizing, a P system can evolve objects representing $z(t)$ to compute GNE. For $n = 2$, a single object that represents $z_i^k(t)$, represents 1% of the agents of population k that follow the strategy i. For example, if we have 16 objects that represent $z_i^k(t)$, then $z_i^k(t) = 0.16$. Formally, to approximate and discretize, we perform $\text{round}(x, n) = \lfloor 10^n \cdot x \rfloor$. To obtain the next value of the variables in the next instant $t + t_{step}$ using the values of instant t, we use Euler's method Butcher (2016), this is, $z(t + t_{step}) = z(t) + \dot{z}(t) \cdot t_{step}$.

In Algorithm 1, other stop conditions can be easily defined, for example, comparing the $z(t)$ values of one iteration with those of the previous one (in constant time) and stopping if no difference is found, but more rules would be necessary. For the sake of simplicity, our stop condition is to limit the number of iterations in the loop.

Because performing multiplications using P systems is not trivial, we define a P system that replicates the Russian peasant multiplication algorithm Cameron (1994) in Appendix A. The reason for using this specific algorithm is that the number of time steps required to compute a multiplication is upper bound by a constant for all multiplications of our P system. We use this multiplication P system as a module for our P system.

Algorithm 1 General overview of the P system computation

Require: $L \geq 0$, $t \geq 0$, $s = 0$
while $s \leq L$ **do**
1. Compute payoff $p(t)$ for current iteration t.
2. Compute sums $\sum_{j \in S^k} z_j^k(t) \cdot p_j^k(t)$
3. Compute $[\hat{p}_i^k]_+$ and $[\sum_{j \in S^k} \hat{p}_j^k]_+$
4. Compute $\dot{z}_i^k(t)$
5. Update $z(t)$ and output results.
6. $s := s + 1$

end while

4.1 Definition of the P system

The P system to compute approximations of GNE under the BNN dynamics is defined as the construct:

$$\Pi = \langle \Gamma, H, \mu, (w_h)_{h \in H}, (\mathcal{R}, \rho), i_{out} \rangle$$

where the alphabet of objects is given by:

$$\begin{aligned}
\Gamma = {} & \{\langle Prod, k, i, l\rangle, \langle k, i, l\rangle \mid k \in \mathcal{P}, i \in S^k, l = \sum_{j<k} |S^j| + i\} \\
& \cup \{\langle Prod2, k, i, l\rangle, \langle a, k, i, l\rangle \mid k \in \mathcal{P}, i \in S^k, l = \sum_{j<k} |S^j| + i\} \\
& \cup \{c, rem, mult_0, mult_1, prod, prod_0, e, e_0, pos, q, s_0, s_1, zneg, zvarp, zvarn\} \\
& \cup \{a, b, d, m, f, y_0, p, n, comp\} \cup \{p_l \mid 1 \leq l \leq \sum_{k \in \mathcal{P}} |S^k|\} \\
& \cup \{m_i, k_i, a_i, b_i, f_i, y_i \mid 1 \leq i \leq 6\} \cup \{over, p_1, err, v\} \\
& \cup \{y_{0,0}, y_{0,1}, y_{0,2}, y_{2,0}, y_{2,1}, y_{2,2}, y_{3,0}, y_{4,0}, y_{4,1}, y_{4,2}, y_{4,3}, y_{5,0}\} \\
& \cup \{y_{7,k} \mid k \in \mathcal{P}\} \cup \{\langle AUX, n\rangle, \langle AUX1, n\rangle, \langle CLK, n\rangle | n \geq 0\} \\
& \cup \{y_{3,j,i}, multz_{i,0}, multz_{i,1} \mid k \in \mathcal{P}, i \in S^k, 1 \leq j \leq 7\} \\
& \cup \{C_k, \langle q, i\rangle, d_i, q_i, neg_i, pos_i \mid k \in \mathcal{P}, i \in S^k, 1 \leq j \leq 7\} \\
& \cup \{y_{0,k}, y_{0,i}, y_{5,j}, y_{5,3,i}, y_{5,11,i} \mid k \in \mathcal{P}, i \in S^k, 1 \leq j \leq 10\} \\
& \cup \{y_{5,12,k}, w_i, compw_i, z_i, \mid k \in \mathcal{P}, i \in S^k, 1 \leq j \leq 10\} \\
& \cup \{EXIT_i, \langle EXIT, k, i, l, n\rangle \mid k \in \mathcal{P}, i \in S^k, 1 \leq j \leq 10, \\
& n \geq 1\} \cup \{\langle INIT, k, i, l\rangle \mid k \in \mathcal{P}, i \in S^k, 1 \leq j \leq 10, n \geq 1\};
\end{aligned}$$

the set of membrane labels is given by

$$\begin{aligned}
H = {} & \{0\} \cup \mathcal{P} \cup \{S_{i,k}\}_{\forall i \in S^k \forall k \in \mathcal{P}} \cup \{RES_{i,k}\}_{\forall i \in S^k \forall k \in \mathcal{P}} \\
& \cup \{MULT_{i,k}\}_{\forall i \in S^k \forall k \in \mathcal{P}} \cup \{M1, M2\} \\
& \cup \{MULT2_{i,k}\}_{\forall i \in S^k \forall k \in \mathcal{P}} \cup \{UPD_{i,k}\}_{\forall i \in S^k \forall k \in \mathcal{P}} \\
& \cup \{ACUM_k\}_{\forall k \in \mathcal{P}};
\end{aligned}$$

the membrane structure μ is represented in Fig. 1, and is defined as follows:

- Membrane skin with label 0, inside of which we find:
  1. One membrane with label P.
  2. N membranes with labels $\mathcal{P}$. Inside of each membrane $k \in \mathcal{P}$:
     2.1. S^k membranes with labels $S_{i,k}$ $\forall i \in S^k$. Inside each $S_{i,k}$:

– One membrane with label $RES_{i,k}$

2.2. S^k membranes with labels $MULT_{i,k}$ $\forall i \in S^k$. Inside each membrane $MULT_{i,k}$:

– One membrane with label $M1$
– One membrane with label $M2$

2.3. S^k membranes with labels $MULT2_{i,k}$ $\forall i \in S^k$. Inside each membrane $MULT2_{i,k}$:

– One membrane with label $M1'$
– One membrane with label $M2'$

2.4. S^k membranes with labels $UPD_{i,k}$ $\forall i \in S^k$
2.5. One membrane with label $ACUM_k$

the output region i_{out} is the skin (label 0);

the initial multisets are $w_P = y_0$, $w_{S_{i,k}} = \langle k, i, l\rangle^{z_i^k}$ $\forall i \in S^k$ $\forall k \in \mathcal{P}$ with $z_i^k := \lfloor \frac{100}{n^k} \rfloor$ for $1 \leq i < \max\{S^k\}$ and $z^k_{max\{S^k\}} := 100 - (|S^k| - 1) \cdot \lfloor \frac{100}{n^k} \rfloor$, $w_{RES_{i,k}} = \langle AUX, 0\rangle$ $\forall i \in S^k$ $\forall k \in \mathcal{P}$, and for any other membrane m, the initial multiset is $w_m = \emptyset$;

and the set of rules $\mathcal{R}$ is given by the following rules, separated by the corresponding stage of Algorithm 1, where λ represents the empty multiset, the rule $RS_{m,n}$ represents the n-th rule of the m-th stage, the rules are defined $\forall k \in \mathcal{P}$, $\forall i \in S^k$, and $l = \sum_{j<k} |S^j| + i$, and $\rho_{m,n}$ represents the priority of the rule $RS_{m,n}$:

Stage 1 (Computate payoff $p(t)$)

To use $f(z(t))$ in the P system, let $\kappa := \lfloor 100 \cdot (-C^\top \bar{J} - \beta) \rfloor$ be the constant part of Eq. (6), and let $a_{j,l} := \lfloor (S \cdot M)_{jl} \rfloor$ and $b_l := \lfloor (S \cdot M)_{ll} + (\alpha \cdot M)_l \rfloor$ for $1 \leq j, l \leq \sum_{k \in \mathcal{P}} |S^k|$ be the coefficients that will multiply z_i^k in Eq. 6. Notice that because $a_{j,l}$ and b_l are used to compute products by multiplying them by z_i^k, and z_i^k is already multiplied by 100, neither $a_{j,l}$ nor b_l are multiplied by 100 when rounded.

The intuition behind the rules of this stage is that objects $\langle k, i, l\rangle$ represent $z_i^k(t)$, and they will be used to compute $p_l(t)$ by generating objects p_l (see Eq. (6)).

Rules $RS_{1,1}$, $RS_{1,3}$, and $RS_{1,4}$ move objects $\langle k, i, l\rangle$, which represent $z_i^k(t)$, until they are in membrane P. At the same time, they produce objects that will be used in later stages (c, $\langle Prod, k, i, l\rangle$), and objects that will be used for

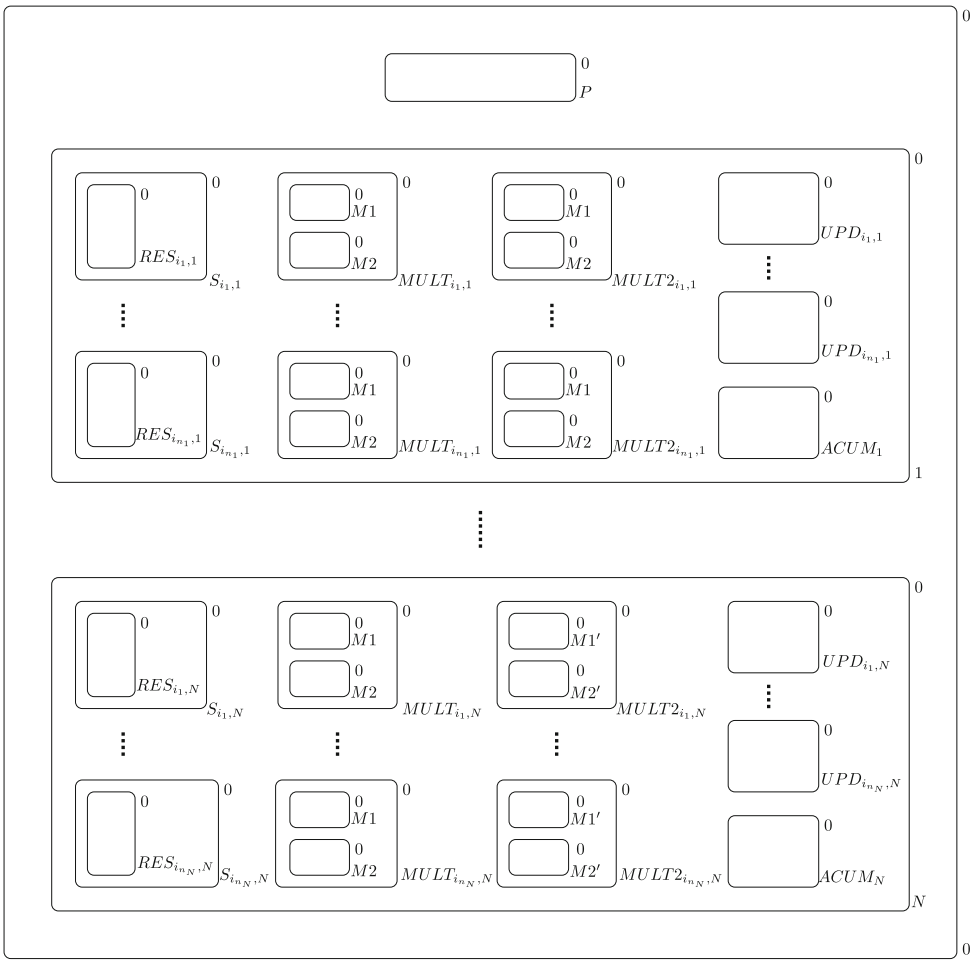

Fig. 1 Membrane structure of our P system. All membranes start with polarization 0

coordination (C_k). Rule $RS_{1,2}$ generates objects p_l representing the constant part of Eq. 6.

$RS_{1,1} \equiv [\langle k,i,l\rangle]^0_{S_{i,k}} \rightarrow [c]^0_{S_{i,k}}\langle k,i,l\rangle$

$RS_{1,2} \equiv [y_0 \rightarrow p_1^{\kappa_1}p_2^{\kappa_2}\ldots p_n^{\kappa_n}]^0_P$

$RS_{1,3} \equiv [\langle k,i,l\rangle]^0_k \rightarrow [\langle Prod,k,i,l\rangle]^0_k\langle k,i,l\rangle C_k$

$RS_{1,4} \equiv \langle k,i,l\rangle[\ ]^0_P \rightarrow [\langle k,i,l\rangle]]^0_P$

Rules $RS_{1,5}$ and $RS_{1,6}$ are used to coordinate the system. Once the polarization of the membrane P changes to $+$, rule $RS_{1,7}$ is applied, and the objects p_l generated are mixed with the objects p_l that were already in membrane P because of rule $RS_{1,2}$, computing indeed $p(t)$ as expressed in Eq. 6.

$RS_{1,5} \equiv [C_1^{100}C_2^{100}\ldots C_N^{100} \rightarrow y_1]^0_0$

$RS_{1,6} \equiv y_1[\ ]^0_P \rightarrow [y_2]^+_P$

$$RS_{1,7} \equiv [\langle k,i,l\rangle \rightarrow p_1^{a_{1,l}}p_2^{a_{2,l}}\ldots p_{l-1}^{a_{l-1,l}}p_l^{b_l}p_{l+1}^{a_{l+1,l}}\ldots p_n^{a_{n,l}}]^+_P$$

Rules $RS_{1,8}$, $RS_{1,9}$, and $RS_{1,11}$ are used for coordinating. The coordination is achieved by changing the polarization of membrane P. Rule $RS_{1,10}$ extracts from P objects p_l, that now represent $p_l(t) = p_i^k(t)$, as objects $\langle a,k,i,l\rangle$.

$RS_{1,8} \equiv [y_2 \rightarrow y_3]^+_P$

$RS_{1,9} \equiv [y_3]^+_P \rightarrow [y_4]^-_P rem$

$RS_{1,10} \equiv [p_l]^-_P \rightarrow [\ ]^-_P\langle a,k,i,l\rangle$

$RS_{1,11} \equiv [y_4 \rightarrow y_5]^-_P$

Rule $RS_{1,12}$ moves each object $\langle a,k,i,l\rangle$ to its corresponding membrane k. These objects will later be used in Stage 2. Rules $RS_{1,13}$ to $RS_{1,15}$ are used to coordinate the beginning of Stage 2. Rule $RS_{1,16}$ is a cleaning rule used to remove objects *rem* that are now useless.

$RS_{1,12} \equiv \langle a,k,i,l\rangle[\ ]^0_k \rightarrow [\langle a,k,i,l\rangle]^0_k$

$RS_{1,13} \equiv [y_5 \rightarrow y_6]^-_P$

$RS_{1,14} \equiv [y_6]^-_P \rightarrow [\ ]^0_P\ y_{7,1}\ y_{7,2}\ldots y_{7,N}$

$RS_{1,15} \equiv y_{7,k}[\ ]^0_k \rightarrow [mult_0^{|S^k|}]^-_k,\ \forall k \in \mathcal{P}.$

$RS_{1,16} \equiv [rem \rightarrow \lambda]^s_m,\ \forall m \in H,\ \forall s \in \{+,-,0\}$ (Cleaning rule).

Stage 2 (Compute sums $\sum_{j\in S^k} z_j^k(t)p_j^k(t)$)

To compute the products of $z_j^k(t)$ and $p_j^k(t)$, we use the membranes $MULT_{i,k}$, which work as multiplication modules. These modules compute the product of two numbers and return the result in a maximum of 43 transition steps.

Each membrane $MULT_{i,k}$ contains two membranes, $M1$ and $M2$. When objects a are placed in $M1$, objects b are placed in $MULT_{i,k}$, an object k_1 is placed in $M1$, and the polarization of the three membranes is 0, the multiplication module will compute the product of the numbers represented by objects a and b. Membranes $MULT_{i,k}$ expel objects d, representing the product, and an object f representing that the multiplication process is finished. For the sake of simplicity, we left the details about the multiplication process in Appendix A, including a computation analysis.

The intuition behind the rules of this stage is that objects $\langle Prod,k,i,l\rangle$ were produced in Stage 1 to represent $z_i^k(t)$. These objects are transformed into objects *prod*. Together with objects $\langle a,k,i,l\rangle$, that represent $p_i^k(t)$, and the multiplication modules $MULT_{i,k}$, we obtain the result of multiplying $z_i^k(t)$ by $p_i^k(t)$. The sum $\sum_{j\in S^k} z_j^k(t)p_j^k(t)$ is obtained by grouping all the resulting objects in the membranes $ACUM_{i,k}$. The rest of the rules are for coordination, rounding, or producing objects necessary for later stages.

Rules $RS_{2,1}$ to $RS_{2,9}$ are used to coordinate the generation of objects a, b, and k_1 in membranes $M1$, $MULT_{i,k}$, and $M1$, respectively. Once this is done, the multiplication process will begin, and the product of $z_i^k(t)$ (represented by objects a) and $p_i^k(t)$ (represented by objects b) will be computed. Objects $\langle Prod2,k,i,l\rangle$ and pos_i are also generated for later stages.

$RS_{2,1} \equiv [\langle Prod,k,i,l\rangle \rightarrow prod\ \langle Prod2,k,i,l\rangle]^-_k$

$RS_{2,2} \equiv [\langle a,k,i,l\rangle \rightarrow e\ pos_i]^-_k$

$RS_{2,3} \equiv mult_0[\ ]^0_{MULT_{i,k}} \rightarrow [mult_0]^+_{MULT_{i,k}}$

$RS_{2,4} \equiv prod[\ ]^+_{MULT_{i,k}} \rightarrow [prod]^+_{MULT_{i,k}}$

$RS_{2,5} \equiv e[\ ]^+_{MULT_{i,k}} \rightarrow [e]^+_{MULT_{i,k}}$

$RS_{2,6} \equiv [mult_0]^+_{MULT_{i,k}} \rightarrow [k_0]^0_{MULT_{i,k}} rem$

$RS_{2,7} \equiv prod \rightarrow [a]^0_{M1}$

$RS_{2,8} \equiv [e \rightarrow b]^0_{MULT_{i,k}}$

$RS_{2,9} \equiv k_0[\ ]^0_{M1} \rightarrow [k_1]^0_{M1}$

Rule $RS_{2,10}$ is used to send objects pos_i to membranes $UPD_{i,k}$, which will be used in later stages.

$RS_{2,10} \equiv pos_i[\ ]^0_{UPD_{i,k}} \rightarrow [pos_i]^0_{UPD_{i,k}}$

Rule $RS_{2,12}$ takes the multiplication results, represented by objects d, and sends them to membrane $ACUM_k$ transformed into objects *neg*. When all multiplications are finished, the sum $\sum_{j\in S^k} z_j^k(t)p_j^k(t)$ is given inside $ACUM_k$, represented by objects *neg*. Then, rule $RS_{2,11}$ changes the polarization of $ACUM_k$ to indicate that the sum is computed. Because $z_j^k(t)$ and $p_j^k(t)$ were rounded by multiplying by 100, it is necessary to use rules $RS_{2,13}$, $RS_{2,14}$, and $RS_{2,15}$ to round the product again and eject them from membrane $ACUM_k$.

$RS_{2,11} \equiv f^{|S^k|}[\ ]^0_{ACUM_k} \rightarrow [y_{2,0}]^+_{ACUM_k}$

$RS_{2,12} \equiv d[\ ]^0_{ACUM_k} \rightarrow [neg]^0_{ACUM_k}$

$RS_{2,13} \equiv [neg^{100}]^+_{ACUM_k} \rightarrow [\ ]^+_{ACUM_k} neg$

$RS_{2,14} \equiv [neg^{51}]^+_{ACUM_k} \rightarrow [\ ]^+_{ACUM_k} neg$, with $\rho_{2,13} > \rho_{2,14}$.

$RS_{2,15} \equiv [neg \rightarrow \lambda]^+_{ACUM_k}$, with $\rho_{2,14} > \rho_{2,15}$.

Rules $RS_{2,16}$, $RS_{2,17}$, and $RS_{2,18}$ coordinate the beginning of the next stage.

$RS_{2,16} \equiv [y_{2,0} \rightarrow y_{2,1}]^+_{ACUM_k}$

$RS_{2,17} \equiv [y_{2,1}]^+_{ACUM_k} \rightarrow [\ ]^0_{ACUM_k} y_{2,2}$

$RS_{2,18} \equiv [y_{2,2}]^-_k \rightarrow [y_{3,0}]^0_k rem$

Stage 3 (Compute $[\hat{p}^k_i]_+$ and $\sum_{j \in S^k} [\hat{p}^k_j]_+$)

The goal of this rules is, because of Eq. (8), to compute $\hat{p}^k_i$ by computing the difference of two numbers: $p^k_i(t)$ and $\sum_{l \in S^k} z^k_l(t) \cdot p^k_l(t)$ In this stage, we compute $[\hat{p}^k_i]_+$ and $\sum_{j \in S^k} [\hat{p}^k_j]_+$.

From now on, in this stage, let $S^k = \{i_1, \ldots, i_{|S^k|}\}$.

Rules $RS_{3,1}$ and $RS_{3,2}$ are used for coordination.

$RS_{3,1} \equiv [y_{3,0} \rightarrow y_{3,1,i_1} \ \ldots \ y_{3,1,i_{|S^k|}}]^0_k$

$RS_{3,2} \equiv y_{3,1,i}[\ ]^0_{UPD_{i,k}} \rightarrow [rem]^+_{UPD_{i,k}} y_{3,2,i}$

Because we need to compute $\hat{p}^k_i(t)$ for each i, we use rule $RS_{3,3}$ to create $|S^k|$ copies of $\sum_{l \in S^k} z^k_l(t) \cdot p^k_l(t)$, each represented by neg_i. These copies are then sent into membranes $UPD_{i,k}$ by using rule $RS_{3,4}$.

$RS_{3,3} \equiv [neg \rightarrow neg_{i_1} \ \ldots \ neg_{i_{|S^k|}}]^0_k$

$RS_{3,4} \equiv neg_i[\ ]^+_{UPD_{i,k}} \rightarrow [neg_i]^+_{UPD_{i,k}}$

Rules $RS_{3,5}$ and $RS_{3,6}$ are used to coordinate the computation of $p^k_i(t) - \sum_{l \in S^k} z^k_l(t) \cdot p^k_l(t)$ by changing the polarization of membranes $UPD_{i,k}$ to $-$.

$RS_{3,5} \equiv [y_{3,2,i} \rightarrow y_{3,3,i}]^0_k$

$RS_{3,6} \equiv [y_{3,3,i}[\ ]^+_{UPD_{i,k}} \rightarrow [y_{3,4,i}]^-_{UPD_{i,k}}]^0_k$

Rules $RS_{3,7}$, $RS_{3,8}$, and $RS_{3,9}$ compute $p^k_i(t) - \sum_{l \in S^k} z^k_l(t) \cdot p^k_l(t)$. If the difference is positive, objects q_i remain. Otherwise, because we are interested in using this difference to compute $[\hat{p}^k_i]_+$, objects neg_i are eliminated. $[\hat{p}^k_i]_+$ is then represented by objects q_i.

$RS_{3,7} \equiv [neg_i\ pos_i \rightarrow \lambda]^-_{UPD_{i,k}}$

$RS_{3,8} \equiv [neg_i \rightarrow \lambda]^-_{UPD_{i,k}}$, with $\rho_{3,7} > \rho_{3,8}$.

$RS_{3,9} \equiv [pos_i \rightarrow q_i]^-_{UPD_{i,k}}$, with $\rho_{3,7} > \rho_{3,9}$.

Rules $RS_{3,10}$ and $RS_{3,11}$ are used for coordination. Rule $RS_{3,12}$ generates objects q and ejects them from membranes $UPD_{i,k}$. By doing this, we compute the sum $\sum_{j \in S^k} [\hat{p}^k_j]_+$, while $[\hat{p}^k_i]_+$ is still represented by objects q_i. Rule $RS_{3,13}$ coordinates the beginning of the next stage.

$RS_{3,10} \equiv [y_{3,4,i} \rightarrow y_{3,5,i}]^-_{UPD_{i,k}}$

$RS_{3,11} \equiv [y_{3,5,i}]^-_{UPD_{i,k}} \rightarrow [y_{3,6,i}]^0_{UPD_{i,k}} rem$

$RS_{3,12} \equiv [q_i]^0_{UPD_{i,k}} \rightarrow [\ ]^0_{UPD_{i,k}} q\ q_i$

$RS_{3,13} \equiv [y_{3,6,i}]^0_{UPD_{i,k}} \rightarrow [\ ]^0_{UPD_{i,k}} y_{3,7,i}$

Stage 4 (Compute $z^k_i(t)$)

Because of Eq. (7), to compute $z^k_i(t)$ we need to compute first $z^k_i \cdot \sum_{j \in S^k} [\hat{p}^k_j]_+$. From Stage 3, we have objects q that represent $\sum_{j \in S^k} [\hat{p}^k_j]_+$, and from Stage 2 we have objects $\langle Prod2, k, i, l \rangle$ that represent z^k_i. As in Stage 2, we can use membranes $MULT2_{i,k}$ as multiplication modules. These modules work the same as $MULT_{i,k}$ from Stage 2, computing products in a maximum of 43 transition steps, and with the only difference that $MULT2_{i,k}$ returns objects d_i instead of d and objects f_1 instead of f (see Appendix A for more details).

When the multiplication process is finished, we will have objects q_i from Stage 3 that represent $[\hat{p}^k_i]_+$, and we will have obtained objects d_i that represent $z^k_i \cdot \sum_{j \in S^k} [\hat{p}^k_j]_+$. We can then compute $z^k_i(t)$ (see Eq. (7)), resulting in objects *zvarp* if it is positive, or *zvarn* if it is negative.

From now on, in this stage, let $S^k = \{i_1, \ldots, i_{|S^k|}\}$.

Rules from $RS_{4,1}$ to $RS_{4,12}$ coordinate the beginning of the multiplication process inside membranes $MULT2_{i,k}$, multiplying z^k_i, represented by objects $\langle Prod2, k, i, l \rangle$, and $\sum_{j \in S^k} [\hat{p}^k_j]_+$, represented by objects q.

$RS_{4,1} \equiv [y_{3,7,i_1} \ldots y_{3,7,i_{|S^k|}} \rightarrow multz_{i_1,0} \ldots multz_{i_{|S^k|},0}]^0_k$

$RS_{4,2} \equiv [multz_{i,0} \rightarrow multz_{i,1}]^0_k$

$RS_{4,3} \equiv [q \rightarrow \langle q, i_1 \rangle \ldots \langle q, i_{|S^k|} \rangle]^0_k$

$RS_{4,4} \equiv \langle Prod2, k, i, l \rangle [\ ]^0_{MULT2_{i,k}} \rightarrow [prod]^0_{MULT2_{i,k}}$

$RS_{4,5} \equiv \langle q, i \rangle [\ ]^0_{MULT2_{i,k}} \rightarrow [e]^0_{MULT2_{i,k}}$

$RS_{4,6} \equiv multz_{i,1}[\ ]^0_{MULT2_{i,k}} \rightarrow [mult_0]^+_{MULT2_{i,k}}$

$RS_{4,7} \equiv [prod \rightarrow prod_0]^+_{MULT2_{i,k}}$

$RS_{4,8} \equiv [e \rightarrow e_0]^+_{MULT2_{i,k}}$

$RS_{4,9} \equiv [mult_0]^+_{MULT2_{i,k}} \rightarrow [mult_1]^0_{MULT2_{i,k}} rem$

$RS_{4,10} \equiv prod_0 \rightarrow [a]^0_{M1'}$

$RS_{4,11} \equiv [e_0 \rightarrow b]^0_{MULT2_{i,k}}$

$RS_{4,12} \equiv mult_1[\ ]^0_{M1'} \rightarrow [k_1]^0_{M1'}$

Rules $RS_{4,13}$ and $RS_{4,14}$ coordinate the beginning of the computation of $[\hat{p}^k_i(t)]_+ - z^k_i(t) \cdot \sum_{j \in S^k} [\hat{p}^k_j(t)]_+$ in membranes $S_{i,k}$.

$RS_{4,13} \equiv f_1^{|S^k|} \rightarrow y^{|S^k|}_{4,0}$

$RS_{4,14} \equiv y_{4,0}[\ ]^0_{S_{i,k}} \rightarrow [y_{4,1}]^+_{S_{i,k}}$

Rules $RS_{4,15}$ to $RS_{4,23}$ compute such difference. If the difference is positive (negative), then objects *zvarp* (*zvarn*) are produced.

$RS_{4,15} \equiv q_i[\ ]^+_{S_{i,k}} \rightarrow [s_0]^+_{S_{i,k}}$

$RS_{4,16} \equiv d_i[\ ]^+_{S_{i,k}} \rightarrow [d_i]^+_{S_{i,k}}$

$RS_{4,17} \equiv [s_0 \rightarrow s_1]^+_{S_{i,k}}$

$RS_{4,18} \equiv [d_i^{100} \rightarrow zneg]^+_{S_{i,k}}$

$RS_{4,19} \equiv [d_i^{51} \rightarrow zneg]^+_{S_{i,k}}$, with $\rho_{4,18} > \rho_{4,19}$.

$RS_{4,20} \equiv [d_i \rightarrow \lambda]^+_{S_{i,k}}$, with $\rho_{4,19} > \rho_{4,20}$.

$RS_{4,21} \equiv [s_1 zneg \rightarrow \lambda]^+_{S_{i,k}}$

$RS_{4,22} \equiv [s_1 \rightarrow zvarp]^+_{S_{i,k}}$, with $\rho_{4,21} > \rho_{4,22}$.

$RS_{4,23} \equiv [zneg \rightarrow zvarn]^+_{S_{i,k}}$, with $\rho_{4,21} > \rho_{4,23}$.

Rules $RS_{4,24}$, $RS_{4,25}$, and $RS_{4,26}$ are just for coordination, preparing the system for the next and final stage.

$RS_{4,24} \equiv [y_{4,1} \rightarrow y_{4,2}]^+_{S_{i,k}}$

$RS_{4,25} \equiv [y_{4,2} \rightarrow y_{4,3}]^+_{S_{i,k}}$

$RS_{4,26} \equiv [y_{4,3}]^+_{S_{i,k}} \rightarrow [y_{5,0}]^-_{S_{i,k}} rem$

Stage 5 (Update $z(t)$ and output results)

Since Stage 1, we have objects c that represent z_i^k. In this stage, we combine them with objects *zvarp* or *zvarn*, representing $\dot{z}(t)$, to update $z(t)$ using Euler's method: $z(t + t_{step}) = z(t) + t_{step} \cdot \dot{z}(t)$.

Rule $RS_{5,1}$ is used for coordination. Because we take $t_{step} = 0.01$, rules $RS_{5,2}$ to $RS_{5,10}$ are used to compute $z(t) + t_{step} \cdot \dot{z}(t)$. Because nothing guarantees that the new values satisfy $0 \leq z_i^k(t) \leq 100$, there are some special cases to consider. Most of the rules from $RS_{5,11}$ to $RS_{5,38}$ are used to deal with these cases, and the rest are for coordination. These cases are further developed in section 4.2, Stage 5.

$RS_{5,1} \equiv [y_{5,0} \rightarrow y_{5,1}]^-_{S_{i,k}}$

$RS_{5,2} \equiv [zvarn^{100}\ c \rightarrow \lambda]^-_{S_{i,k}}$

$RS_{5,3} \equiv [zvarn^{51}\ c \rightarrow \lambda]^-_{S_{i,k}}$, with $\rho_{5,2} > \rho_{5,3}$.

$RS_{5,4} \equiv [zvarp^{100} \rightarrow p]^-_{S_{i,k}}$

$RS_{5,5} \equiv [c \rightarrow p]^-_{S_{i,k}}$, with $\rho_{5,3} > \rho_{5,5}$.

$RS_{5,6} \equiv [zvarn^{100} \rightarrow n]^-_{S_{i,k}}$, with $\rho_{5,3} > \rho_{5,6}$.

$RS_{5,7} \equiv [zvarn^{51} \rightarrow n]^-_{S_{i,k}}$, with $\rho_{5,6} > \rho_{5,7}$.

$RS_{5,8} \equiv [zvarn \rightarrow \lambda]^-_{S_{i,k}}$, with $\rho_{5,7} > \rho_{5,8}$.

$RS_{5,9} \equiv [zvarp^{51} \rightarrow p]^-_{S_{i,k}}$, with $\rho_{5,4} > \rho_{5,9}$.

$RS_{5,10} \equiv [zvarp \rightarrow \lambda]^-_{S_{i,k}}$, with $\rho_{5,9} > \rho_{5,10}$.

$RS_{5,11} \equiv [y_{5,1} \rightarrow y_{5,2}\ comp^{100}]^-_{S_{i,k}}$

$RS_{5,12} \equiv [p^{100}]^-_{S_{i,k}} \rightarrow [over]^-_{S_{i,k}} w_i^{100}$

$RS_{5,13} \equiv [y_{5,2}]^-_{S_{i,k}} \rightarrow [y_{5,3}]^0_{S_{i,k}} y_{5,3,i}$

$RS_{5,14} \equiv [p]^0_{S_{i,k}} \rightarrow [\ ]^0_{S_{i,k}} p$

$RS_{5,15} \equiv [over\ comp^{100} \rightarrow \lambda\ ]^-_{S_{i,k}}$

$RS_{5,16} \equiv [n]^0_{S_{i,k}} \rightarrow [\ ]^0_{S_{i,k}} n$

$RS_{5,17} \equiv [comp]^0_{S_{i,k}} \rightarrow [\ ]^0_{S_{i,k}} compw_i$

$RS_{5,18} \equiv [p\ comp \rightarrow p_1]^-_{S_{i,k}}$, with $\rho_{15} > \rho_{18}$.

$RS_{5,19} \equiv [p_1]^0_{S_{i,k}} \rightarrow [\ ]^0_{S_{i,k}} w_i$

$RS_{5,20} \equiv [p\ n \rightarrow \lambda]^0_k$

$RS_{5,21} \equiv [p\ compw_i \rightarrow w_i]^0_k$, with $\rho_{20} > \rho_{21}$.

$RS_{5,22} \equiv [n\ w_i \rightarrow compw_i]^0_k$, with $\rho_{20} > \rho_{22}$.

$RS_{5,23} \equiv [p \rightarrow err]^0_k$, with $\rho_{21} > \rho_{23}$.

$RS_{5,24} \equiv [n \rightarrow err]^0_k$, with $\rho_{22} > \rho_{24}$.

$RS_{5,25} \equiv [y_{5,3} \rightarrow y_{5,4}]^0_{S_{i,k}}$

$RS_{5,26} \equiv [y_{5,4} \rightarrow y_{5,5}]^0_{S_{i,k}}$

$RS_{5,27} \equiv [y_{5,5}]^0_{S_{i,k}} \rightarrow [\ ]^+_{S_{i,k}} y_{5,6}$

$RS_{5,28} \equiv [y_{5,3,i_1} \ldots y_{5,3,i_{|S^k|}} \rightarrow y_{5,4}]^0_k$

$RS_{5,29} \equiv [y_{5,4}]^0_k \rightarrow [y_{5,5}\ v^{100}]^+_k rem$

$RS_{5,30} \equiv [w_i\ v \rightarrow z_i]^+_k$

$RS_{5,31} \equiv [w_i \rightarrow \lambda]^+_k$, with $\rho_{5,30} > \rho_{5,31}$.

$RS_{5,32} \equiv [compw_i \rightarrow \lambda]^+_k$

$RS_{5,33} \equiv [v \rightarrow z_i]^+_k$, with $\rho_{5,30} > \rho_{5,33}$.

$RS_{5,34} \equiv [y_{5,5}]^+_k \rightarrow [\ ]^0_k rem$

$RS_{5,35} \equiv z_i[\ ]^+_{S_{i,k}} \rightarrow [z_i]^+_{S_{i,k}}$

$RS_{5,36} \equiv y_{5,6}[\ ]^+_{S_{i,k}} \rightarrow [y_{5,7}]^0_{S_{i,k}}$

$RS_{5,37} \equiv y_{5,7}[\ ]^0_{RES_{i,k}} \rightarrow [y_{5,8}]^+_{RES_{i,k}}$

$RS_{5,38} \equiv z_i[\ ]^+_{RES_{i,k}} \rightarrow [EXIT_i]^+_{RES_{i,k}}$

Rules from $RS_{5,39}$ to $RS_{5,47}$ are used for coordination and for preparing the output of the P system. Objects $\langle AUX, n\rangle$ are used to count how many iterations of Algorithm 1 have been completed. Objects $\langle EXIT, k, i, l, n\rangle$ represent the values $z_i^k(t)$ at iteration n. Objects $\langle INIT, k, i, l\rangle$ are used to reset the P system, preparing Stage 1 for a new iteration of the algorithm.

$RS_{5,39} \equiv [\langle AUX, n\rangle \rightarrow \langle CLK, n+1\rangle^{100} \langle AUX1, n+1\rangle]^+_{RES_{i,k}}$

$RS_{5,40} \equiv [y_{5,8}]^+_{RES_{i,k}} \rightarrow [y_{5,9}]^-_{RES_{i,k}} rem$

$RS_{5,41} \equiv [EXIT_i\ \langle CLK, n\rangle]^-_{RES_{i,k}} \rightarrow [\ ]^-_{RES_{i,k}} \langle EXIT, k, i, l, n\rangle$, with $n \geq 1$

$RS_{5,42} \equiv [\langle CLK, n\rangle \rightarrow \lambda]^-_{RES_{i,k}}$, with $n \geq 1$ and $\rho_{5,41} > \rho_{5,42}$.

$RS_{5,43} \equiv [y_{5,9}]^-_{RES_{i,k}} \rightarrow [\ ]^0_{RES_{i,k}} y_{5,10}$

$RS_{5,44} \equiv [\langle AUX1, n\rangle \rightarrow \langle AUX, n\rangle]^0_{RES_{i,k}}$, with $n \geq 1$

$$RS_{5,45} \equiv [\langle EXIT, k, i, l, n\rangle]^0_{S_{i,k}} \rightarrow \ [\langle INIT, k, i, l\rangle]^0_{S_{i,k}} \langle EXIT, k, i, l, n\rangle, \text{ with } n \geq 1$$

$$RS_{5,46} \equiv [y_{5,10}]^0_{S_{i,k}} \rightarrow [\ ]^0_{S_{i,k}} y_{5,11,i}$$

$$RS_{5,47} \equiv [\langle EXIT, k, i, l, n\rangle]^0_k \rightarrow \ [\ ]^0_k \langle EXIT, k, i, l, n\rangle, \text{ with } n \geq 1$$

Rules $RS_{5,48}$ to $RS_{5,60}$ coordinate a new iteration of Algorithm 1, sending an object y_0 to membrane P and initializing the new objects $\langle k, i, l\rangle$ for Stage 1.

$RS_{5,48} \equiv [y_{5,11,i_1} \ldots y_{5,11,i_{|S^k|}}]^0_k \rightarrow [\ ]^0_k \ y_{5,12,k}$

$RS_{5,49} \equiv [y_{5,12,1} \ldots y_{5,12,N} \rightarrow y_0]^0_0$

$RS_{5,50} \equiv [err]^0_k \rightarrow [\ ]^0_k err$

$RS_{5,51} \equiv [y_0 \rightarrow y_{0,0} \ y_{0,1} \ldots y_{0,N}]^0_0$

$RS_{5,52} \equiv y_{0,0}[\ ]^0_P \rightarrow [y_{0,0}]^0_P$

$RS_{5,53} \equiv y_{0,k}[\ ]^0_k \rightarrow [y_{0,0}]^0_k$

$RS_{5,54} \equiv [y_{0,0} \rightarrow y_{0,1}]^0_P$

$RS_{5,55} \equiv [y_{0,0}]^0_k \rightarrow [y_{0,i_1} \ldots y_{0,i_{|S^k|}}]^0_k$

$RS_{5,56} \equiv [y_{0,1} \rightarrow y_{0,2}]^0_P$

$RS_{5,57} \equiv y_{0,i}[\ ]^0_{S_{i,k}} \rightarrow [y_{0,2}]^+_{S_{i,k}}$

$RS_{5,58} \equiv [y_{0,2} \rightarrow y_0]^0_P$

$RS_{5,59} \equiv [\langle INIT, k, i, l\rangle \rightarrow \langle k, i, l\rangle]^+_{S_{i,k}}$

$RS_{5,60} \equiv [y_{0,2}]^+_{S_{i,k}} \rightarrow [\ ]^0_{S_{i,k}} rem$

To stop after L iterations of Algorithm 1

$RS_{5,61} \equiv [\langle AUX1, L\rangle \ y_{5,9} \rightarrow \lambda]^-_{RES_{i,k}}$

4.2 Overview of the computation

This subsection provides an analysis of the computation for each stage of Algorithm 1. The result of the computation analysis is that for each time step t, the number of transition steps is upper bound by a constant. This means that the computation time complexity of the global computation only grows linearly with t.

Stage 1: Computation of payoff $p(t)$ In this stage, the payoff of the current iteration is computed. To do this, it is necessary to consider the function f involved in Eq. (6). The computation starts with the initial configuration. If we only consider the membranes involved in this stage, we have the following configuration:

$$\mathbb{C}_0 = [[[\langle k, i, l\rangle^{z_i^k}]^0_{S_{i,k}}]^0_k [y_0]^0_P]^0_0$$

After 8 transitions steps, applying rules from $RS_{1,1}$ to $RS_{1,13}$, we have that each value $p_l(t)$ of $p(t)$ is computed:

$$\mathbb{C}_8 = [[\langle a, k, i, l\rangle^{p_l(t)}]^0_k [y_6]^-_P]^0_0$$

We computed then $p(t)$, as intended. Some rules in Stage 2 will transform the objects $\langle a, k, i, l\rangle$ that are in the membranes $[\ ]_k$, but for now, we are interested in seeing when Stage 1 will finish. Two iterations later, after applying rules $RS_{1,14}$ and $RS_{1,15}$:

$$\mathbb{C}_{10} = [[\ldots mult_0^{|S^k|}]^-_k \ [\ ]^0_P]^0_0$$

Stage 2: Computation of the sums $\sum_{j \in S^k} z_j^k(t) p_j^k(t)$

Stage 2 starts with the last configuration of Stage 1: $\mathbb{C}_{10}$. After $\mathbb{C}_2$ and $\mathbb{C}_8$, because of the effects of rules $RS_{1,3}$ and $RS_{1,12}$, there are some elements $\langle Prod, k, i, l\rangle$ and $\langle a, k, i, l\rangle$ in $[\ ]_k$. These membranes k changed their polarization from 0 to $-$ in $\mathbb{C}_{10}$, initiating Stage 2. After only three transition steps, and applying rules from $RS_{2,1}$ to $RS_{2,9}$:

$$\mathbb{C}_{13} = [[[b^{p_l(t)}[a^{z_i^k} k_1]^0_{M1}[\ ]^0_{M2}]^0_{MULT_{i,k}} \& [pos_i^{p_l(t)}]^0_{UPD_{i,k}} \ [\ ]^0_{ACUM_k} rem^{|S^k|}]^-_k]^0_0$$

In this configuration, $p_l = p_i^k$, so $z_i^k \cdot p_i^k$ is computed in each membrane $MULT_{i,k}$. Because the round function multiplies by 100, the (rounded) result of the multiplication will be $100 \cdot z_i^k \cdot p_i^k$. Furthermore, each product is computed after (at most) 43 transition steps, as proven in Appendix A. Some objects d may be introduced as objects pos into membranes $ACUM_k$ before finishing these 43 transition steps. After (at most) 44 steps, having applied rule $RS_{2,10}$, and ignoring membranes $MULT_{i,k}$:

$$\mathbb{C}_{\leq 57} = [[[pos_i^{p_l(t)}]^0_{UPD_{i,k}} \ [neg^{100 \cdot z_i^k \cdot p_i^k} y_{2,0}]^+_{ACUM_k}]^-_k]^0_0$$

After three transition steps, applying rules $RS_{2,11}$ to $RS_{2,18}$:

$$\mathbb{C}_{\leq 60} = [[[pos_i^{p_l(t)}]^0_{UPD_{i,k}} \ [\ ]^0_{ACUM_k} \ neg^{\pi_k} \ y_{3,0} \ rem]^0_k]^0_0$$

where $\pi_k = \sum_{i \in S^k} z_i^k(t) \cdot p_i^k(t)$. By the end of this stage, we computed then π_k, as intended.

Stage 3: Computation of $[\hat{p}_i^k]_+$ and $\sum_{j \in S^k} [\hat{p}_j^k]_+$

This stage starts with the last configuration of Stage 2: $\mathbb{C}_{\leq 60}$. The change of polarization of membranes $[\ ]_k$ from $-$ to 0 and the presence of objects $y_{3,0}$ initiate Stage 3. Following all rules of Stage 3 ($RS_{3,1}$ to $RS_{3,13}$), we can see that after 7 steps we have the configuration:

$$\mathbb{C}_{\leq 67} = [[[\ ]^0_{UPD_{i,k}} q^{\sigma_k} \ q_i^{[\hat{p}_i^k]_+} \ y_{3,7,i}]^0_k]^0_0$$

where $\sigma_k = \sum_{j \in S^k} [\hat{p}_j^k]_+$.

Stage 4: Computation of $z_i^k(t)$

Stage 4 starts with the last configuration of Stage 3: $\mathbb{C}_{\leq 67}$. After objects $\langle Prod2, k, i, l\rangle$ were created in computation $\mathbb{C}_{11}$, they are processed and introduced in membrane

$MULT2_{i,k}$ in computation $\mathbb{C}_{12}$, but the product is not initiated until all objects $y_{3,7,i}$ are generated using rule $RS_{3,13}$:

$$\mathbb{C}_{\leq 67} = [[\ [\ ]^0_{M1'}[\ ]^0_{M2'} prod^{z_i^k}]^0_{MULT2_{i,k}}[\ ]^0_{S_{i,k}} q^{\sigma_k}\ q_i^{[\hat{p}_i^k]_+}\ y_{3,7,i}]^0_k]^0_0$$

Five iterations later, after applying rules from $RS_{4,1}$ to $RS_{4,12}$, the multiplication in $MULT2_{i,k}$ can start:

$$\mathbb{C}_{\leq 72} = [[\ [a^{z_i^k} k_1]^0_{M1'}[\ ]^0_{M2'} b^{\sigma_k}]^0_{MULT2_{i,k}}[\ ]^0_{S_{i,k}} rem\ q_i^{[\hat{p}_i^k]_+}]^0_k]^0_0$$

After at most 43 iterations, as proven in Appendix A:

$$\mathbb{C}_{\leq 115} = [[\ ]^0_{S_{i,k}} d_i^{100\cdot z_i^k\cdot\sigma_k}\ f_1^{|S^k|}\ q_i^{[\hat{p}_i^k]_+}]^0_k]^0_0$$

After four more iterations, using rules from $RS_{4,13}$ to $RS_{4,20}$, $RS_{4,24}$, and $RS_{4,25}$:

$$\mathbb{C}_{\leq 119} = [[y_{4,3}\ zneg^{z_i^k\cdot\sigma_k} s_1^{[\hat{p}_i^k]_+}]^+_{S_{i,k}}]^0_k]^0_0$$

Now, depending on the result, the objects generated will be different. We represent by $zvar(p||n)$ the possibility of having objects $zvarp$ or $zvarn$.

Considering that $z_i^k = [\hat{p}_i^k]_+ - z_i^k \cdot \sum_{j\in S^k}[\hat{p}_j^k]_+$, after applying rules from RS_{21} to $RS_{4,23}$ depending on the case, and $RS_{4,26}$, we have:

$$\mathbb{C}_{\leq 120} = [[y_{5,0} zvar(p||n)^{z_i^k}]^-_{S_{i,k}}]^0_k]^0_0$$

Stage 5: Update of $z(t)$, coordination for next iteration and results output

This stage starts with the last configuration of Stage 4: $\mathbb{C}_{\leq 120}$. In the initial multiset, there were some copies of $\langle AUX, 0\rangle$ that have not been processed yet. Since configuration $\mathbb{C}_1$, there are also objects c in membranes $S_{i,k}$ that have not been processed because the polarization has changed from 0 to $-$ in the last configuration for the first time since:

$$\mathbb{C}_{\leq 120} = [[[\langle AUX, 0\rangle]^0_{RES_{i,k}} y_{5,0}\ zvar(p||n)^{z_i^k}\ c^{z_i^k}]^-_{S_{i,k}}]^0_k]^0_0$$

To update $z(t)$, we use Euler's method with $z(t+0.01) = z(t) + 0.01\cdot\dot{z}(t)$. Depending on the objects existing at the end of Stage 4, there are three possible cases. Let $m_i = z_i^k + 0.01\cdot\dot{z}_i^k$. Then:

- **Case 1**: If $0 \leq m_i < 100$, then three transition steps later using rules from $RS_{5,1}$ to $RS_{5,11}$, $RS_{5,13}$, and $RS_{5,18}$:

  $$\mathbb{C}_{\leq 123} = [[[\langle AUX, 0\rangle]^0_{RES_{i,k}} y_{5,3}\ comp^{100-m_i}\ p_1^{m_i}]^0_{S_{i,k}}\ y_{5,3,i_1}\cdots y_{5,3,i_{|S^k|}}]^0_k]^0_0$$

  And finally, using rules $RS_{5,17}$, $RS_{5,19}$, $RS_{5,25}$, and $RS_{5,28}$:

  $$\mathbb{C}_{\leq 124} = [[[\langle AUX, 0\rangle]^0_{RES_{i,k}} y_{5,4}]^0_{S_{i,k}}\ compw_i^{100-m_i}\ w_i^{m_i} y_{5,4}]^0_k]^0_0$$

- **Case 2**: If $m_i < 0$, then three transition steps later using rules from $RS_{5,1}$ to $RS_{5,3}$, from $RS_{5,6}$ to $RS_{5,8}$, $RS_{5,11}$, and $RS_{5,13}$:

  $$\mathbb{C}_{\leq 123} = [[[\langle AUX, 0\rangle]^0_{RES_{i,k}} y_{5,3}\ comp^{100}\ n^{m_i}]^0_{S_{i,k}} y_{5,3,i_1}\cdots y_{5,3,i_{|S^k|}}]^0_k]^0_0$$

  And finally, using rules $RS_{5,16}$, $RS_{5,17}$, $RS_{5,25}$, and $RS_{5,28}$:

  $$\mathbb{C}_{\leq 124} = [[[\langle AUX, 0\rangle]^0_{RES_{i,k}} y_{5,4}]^0_{S_{i,k}}\ compw_i^{100}\ n^{m_i}\ y_{5,4}]^0_k]^0_0$$

- **Case 3**: If $m_i \geq 100$, then three transition steps later using rules $RS_{5,1}$, $RS_{5,4}$, $RS_{5,5}$, from $RS_{5,9}$ to $RS_{5,13}$, and $RS_{5,15}$:

  $$\mathbb{C}_{\leq 123} = [[[\langle AUX, 0\rangle]^0_{RES_{i,k}} y_{5,3}\ p^{m_i-100}]^0_{S_{i,k}} y_{5,3,i_1}\cdots y_{5,3,i_{|S^k|}} w_i^{100}]^0_k]^0_0$$

  Notice that, in this stage, objects $y_{5,3,i}$ are always created in the same configuration in each of the three cases. Finally, using rules $RS_{5,14}$, $RS_{5,25}$, and $RS_{5,28}$:

  $$\mathbb{C}_{\leq 124} = [[[\langle AUX, 0\rangle]^0_{RES_{i,k}} y_{5,4}]^0_{S_{i,k}}\ w_i^{100}\ p^{m_i-100}\ y_{5,4}]^0_k]^0_0$$

Regardless of the result of each case, objects w_i represent the new z_i^k, objects $compw_i$ represent the difference between z_i^k and 100, and objects p and n represent positive and negative overflow respectively. If both overflows exist, they will cancel each other for the next configuration. These objects are introduced because, while using Euler's method, there is nothing that assures that $0 \leq z_i^k(t) \leq 100$ $\forall t \geq 1$. The next configuration, using rules from $RS_{5,20}$ to $RS_{5,24}$, $RS_{5,26}$, and $RS_{5,29}$, is then:

$$\mathbb{C}_{\leq 125} = [[[\langle AUX, 0\rangle]^0_{RES_{i,k}} y_{5,5}]^0_{S_{i,k}}\ err^{\mu}\ w_i^{z_i^k}\ compw_i^{100-z_i^k}\ y_{5,5}\ v^{100}]^+_k rem]^0_0$$

Objects err are created only in case some overflow appears. Seven transition steps later, using rules $RS_{5,27}$ and from $RS_{5,30}$ to $RS_{5,48}$, the objects $EXIT$ will be in the skin. These objects will represent the final result of $z(t)$ for each t:

$$\mathbb{C}_{\leq 132} = [[[\langle AUX, 1\rangle\ ]^0_{RES_{i,k}}\ \langle INIT, k, i, l\rangle^{z_i^k}]^0_{S_{i,k}}]^0_k \langle EXIT, k, i, l, 1\rangle^{z_i^k}\ y_{5,12,1}\cdots y_{5,12,N}]^0_0$$

Finally, after six more iterations, using rules from $RS_{5,49}$ to $RS_{5,60}$:

$$\mathbb{C}_{\leq 138} = [[[\langle AUX, 1\rangle\ ]^0_{RES_{i,k}}\ \langle k, i, l\rangle^{z_i^k}]^0_{S_{i,k}} rem]^0_k \langle EXIT, k, i, l, 1\rangle^{z_i^k} [y_0]^0_P]^0_0$$

It is important to notice that, in this last configuration, object y_0 is in membrane P and the objects $\langle k, i, l\rangle$ are in S_i^k. This configuration is then analogous to configuration $\mathbb{C}_0$, so we know that the system can run another iteration of Algorithm 1. When rule $RS_{5,61}$ is eventually triggered, object $y_{5,9}$ will be consumed, and this will prevent the generation of new objects y_0 and $\langle k, i, l\rangle$. This will eventually make the P system stop evolving (see rules $RS_{5,43}$, $RS_{5,46}$, $RS_{5,48}$, $RS_{5,49}$, and from $RS_{5,51}$ to $RS_{5,59}$).

In conclusion, the transition from $z(t)$ to $z(t + t_{step})$ has a constant upper bound of 138 steps, and it does not depend on the size of the problem or the t_{step} chosen. If we had to compute GNE by using Euler's method iteratively to compute the evolution of $z(t)$, we would have to update the values $z_i^k(t)$ in each iteration using Eqs. (6), (7), and (8) for each $k \in \mathcal{P}$ and each $i \in S^k$. This would require a computation time proportional to $\sum_{k\in\mathcal{P}} |S^k|$ for each iteration. If t iterations of Euler's method are required to compute the GNE, then the computation time complexity is proportional to $\mathcal{O}(t \cdot \sum_{k\in\mathcal{P}} |S^k|)$. In contrast, the computation time our P system requires to update $z(t)$ is upper bound by a constant that does not depend on $\sum_{k\in\mathcal{P}} |S^k|$. This implies that the computation time complexity is proportional to $\mathcal{O}(t)$. Because of the massive parallelism inherent to P systems, the computation time complexity is reduced by a factor of $\sum_{k\in\mathcal{P}} |S^k|$.

To define this system, it was required to select a specific $f(z(t))$ to compute $p(t)$ (see Eq. (6)). In our case, $f(z(t))$ was given by the Energy Market Game (Sect. 3.1) and involves real numbers. We also used objects that represent 1% of the agents $z(t)$ as explained at the beginning of Sect. 4. Because of these decisions, we rounded every number to two decimals. We can take smaller values of t_{step} to get better approximations if we round to more decimals, but then Euler's method will require more time steps to compute the GNE. However, this does not affect the difference in computation time complexity stated above.

4.3 Experimental results

4.3.1 Setup

Researchers have developed different P system simulators that are available, like P-lingua (MeCoSim) Gutiérrez-Naranjo et al. (2008); García-Quismondo et al. (2009); Pérez-Hurtado et al. (2010) or UPSimulator Guo et al. (2019, 2018). To perform experiments, we use MeCoSim for its simplicity. This simulator allows the design of P systems with particular properties, in this case, transition P systems with membrane polarization.

One limitation of every P system simulator is that it still runs on a computer, so the parallelism inherent to P systems is lost. However, we can use MeCoSim to perform experiments and test the P system designed in Sect. 4.1.

Although some implementations of P systems on GPU devices can be found in the literature (see, e.g., Martínez-del-Amor et al. (2015); Zhang et al. (2021)), our experiment was conducted on a machine with a 3.59GHz 6-core CPU and with 16GB of RAM. We consider the possibility of adapting the implementation to GPU devices, and empirically studying the improvements in time, as future work.

4.3.2 Experiment

To test the correct behavior of our P system, we have considered a small example, where $N = 3$, $T = 5$, $S^1 = \{3, 5\}$, $S^2 = \{1, 3, 5\}$, $S^3 = (S^1)^c = \{1, 2, 4\}$, and we randomly sample the nonzero elements of D, $\bar{J}$, α, β and m^k from [0, 1], [2, 4], [1, 10], [0, 1] and Debreu (1952); Facchinei and Kanzow (2007) respectively. The parameters are sampled from the same distributions described in Martinez-Piazuelo et al. (2022).

In Fig. 2, we can see how the trajectories of the values z_i^k initialize as equally distributed for each player k as possible, and they evolve until reaching a stable distribution after only 10 time steps: $\dot{z} = 0$ after ten steps. At that moment, a GNE is found. It was expected that player 3 would concentrate more agents over time slot 4 (z_4^3) or 2 (z_2^3) since it is the only player with access to those resources. The difference is a consequence of the cost difference. As expected, the results coincide with those obtained by applying Euler's method independently.

Notice that, as explained in Sect. 3, z_i^k represents the proportion of agents in population k that follow strategy i. In the Energy Market Game that we are solving, z_k^i is the proportion of energy purchased in time slot i by player k.

5 Conclusions

We propose a P system based on Euler's method that computes approximations of GNE for population games under BNN dynamics. If Euler's method requires t iterations to compute GNE, we have proved that the computation time complexity of the P system is linear with respect to t, independently of the size of the problem.

Our P system presents some limitations. First, if the function f representing the game is complex enough, it is possible that P systems are not powerful enough to compute it in constant time, increasing the system's time complexity. Second, the system presented only works for population games under BNN dynamics.

This work can be extended in some directions. First, in this paper, some of the evolution rules of the P system are defined specifically for instances of the Energy Market Game as a representative of population games under BNN dynamics. As future work, we propose modifying this P system to compute GNE for other population games under BNN dynamics by changing the payoff function and the corresponding rules. Second, our P system is also easily generalizable for studying the evolution of other EDM-

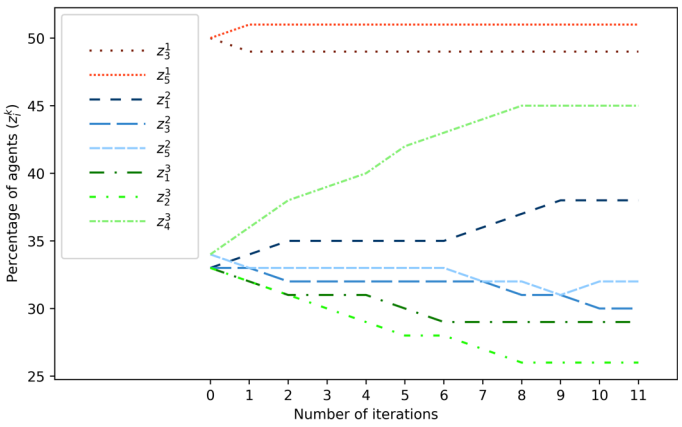

Fig. 2 Small experiment with three players (populations) and five strategies, distributed such that $S^1 = \{3,5\}$, $S^2 = \{1,3,5\}$, $S^3 = (S^1)^c = \{1,2,4\}$. Players 1, 2, and 3 are represented by lines z_i^1, z_i^2, and z_i^3, respectively

PDM systems besides population games under BNN dynamics, even if equilibria are not reached.

Finally, as pointed out in sect. 4.3, the current study can also be simulated in a GPU device to explore further time improvements.

Appendix A Russian peasant multiplication in membrane computing

A P system that computes the product of two numbers $m, n \in \mathbb{N}_0$, inspired by the Russian peasant multiplication algorithm Cameron (1994), is defined in this appendix. The goal is to use it as a multiplication module for the P system defined in Sect. 4.1.

The algorithm is described in Algorithm 2.

Algorithm 2 Russian Peasant Multiplication

Require: $m, n \in \mathbb{Z}_{\geq 1}; res := 0$
  if $m = 2q + 1$ for some $q \in \mathbb{Z}$ **then**
    1. $res := n$
  end if
  while $m \neq 1$ **do**
    2. $m := \lfloor m/2 \rfloor$, $n := 2n$.
    if $m = 2q + 1$ for some $q \in \mathbb{Z}$ **then**
      3. $res := res + n$
    end if
  end while
  4. Return res

For example, if we want to multiply $m = 37$ by $n = 16$ using Algorithm 2, we get the result of Table 1, where t is the iteration of the while loop.

The structure of the P system has three membranes: $\mu = [[]_1[]_2]_0$ and the initial multisets are $w_1 = \{a^m k_1\}$, $w_2 = \emptyset$ and $w_0 = \{b^n\}$. The output region i_{out} is the environment.

The alphabet of objects is given by:

$$\Gamma = \{a, b, c, d, f, m, y_0, y_1, rem\} \cup \{k_i : 1 \leq i \leq 6\} \cup \{a_i : 1 \leq i \leq 5\} \cup \{b_i : 1 \leq i \leq 4\} \cup \{m_i : 1 \leq i \leq 3\} \cup \{f_i : 1 \leq i \leq 4\}$$

The ruleset of this system is the following:

$RS_1 = [k_1 \to k_2\ y_0]_1^0$
$RS_2 = [k_2 \to k_3]_1^0$
$RS_3 = [k_3 \to k_4]_1^0$
$RS_4 = [k_4 \to k_5]_1^0$
$RS_5 = [k_5 \to k_6]_1^0$
$RS_6 = [k_6 \to k_1]_1^0$
$RS_7 = [a^2 \to a_1\ y_1^2]_1^0$
$RS_8 = [a \to m\ y_1]_1^0$, with $\rho_7 > \rho_8$
$RS_9 = [a_1 \to a_2]_1^0$
$RS_{10} = [a_2 \to a_3]_1^0$
$RS_{11} = [a_3 \to a_4]_1^0$
$RS_{12} = [a_4 \to a_5]_1^0$
$RS_{13} = [a_5 \to a]_1^0$
$RS_{14} = [y_1^2\ y_0 \to \lambda]_1^0$

Table 1 Multiplication of $m = 37$ and $n = 16$ using the Russian Peasant Multiplication algorithm (Algorithm 2)

t	m	n	res
0	37	16	16
1	18	32	16
2	9	64	80
3	4	128	80
4	2	256	80
5	1	512	**592**

$RS_{15} = [y_1\ y_0\ m]_1^0 \rightarrow [f]_1^0 f$, with $\rho_{14} > \rho_{15}$
$RS_{16} = [y_1 \rightarrow \lambda]_1^0$, with $\rho_{15} > \rho_{16}$
$RS_{17} = [m]_1^0 \rightarrow [\ ]_1^0 m$, with $\rho_{15} > \rho_{17}$
$RS_{18} = [k_2 y_0]_1^0 \rightarrow [\ ]_1^0 y_0$, with $\rho_{15} > \rho_{18}$ and $\rho_{18} > \rho_2$
$RS_{19} = [b \rightarrow b_1]_0^0$
$RS_{20} = [b_1 \rightarrow b_2]_0^0$
$RS_{21} = [b_2 \rightarrow b_3]_0^0$
$RS_{22} = [b_3 \rightarrow b_4]_0^0$
$RS_{23} = [b_4 \rightarrow c^2]_0^0$
$RS_{24} = [c \rightarrow b]_0^0$
$RS_{25} = [b_3[\ ]_2^0 \rightarrow [d]_2^0 c^2]_0^+$
$RS_{26} = [c \rightarrow b]_0^+$
$RS_{27} = [m]_0^0 \rightarrow [m_1]_0^+ rem$
$RS_{28} = [m_1 \rightarrow m_2]_0^+$
$RS_{29} = [m_2 \rightarrow m_3]_0^+$
$RS_{30} = [m_3]_0^+ \rightarrow [\ ]_0^0 rem$
$RS_{31} = [f\ k_3 \rightarrow \lambda]_1^0$, with $\rho_{31} > \rho_3$
$RS_{32} = f[\ ]_2^0 \rightarrow f_1\ [f_1]_2^0$
$RS_{33} = [f_1]_2^0 \rightarrow [\ ]_2^- rem$
$RS_{34} = [f_1]_0^0 \rightarrow [f_2]_0^- rem$
$RS_{35} = f_2[\ ]_2^- \rightarrow f_3\ [f_3]_2^-$
$RS_{36} = [f_3 \rightarrow f_4]_2^-$
$RS_{37} = [f_3 \rightarrow f_4]_0^-$
$RS_{38} = [f_4]_2^- \rightarrow [\ ]_2^0 rem$
$RS_{39} = [f_4]_0^- \rightarrow [\ ]_0^0 f$
$RS_{40} = [d]_2^- \rightarrow [\ ]_2^- d$
$RS_{41} = [d]_0^- \rightarrow [\ ]_0^- d$
$RS_{42} = [b_4 \rightarrow d]_0^-$
$RS_{43} = [y_0]_0^0 \rightarrow [f_3]_0^- rem$
$RS_{44} = [b_3 \rightarrow \lambda]_0^-$
$RS_{45} \equiv [rem \rightarrow \lambda]_m^s, \quad \forall m \in \{0,1,2\}, \quad \forall s \in \{+,-,0\}$
(Cleaning rule).

A.1Computation analysis

The behavior of this P system is correct. It is easy to prove this by analyzing the following four cases: $m = 0$, $m = 1$, $m = 2p$ with $p \geq 1$ and $m = 2p + 1$ with $p \geq 1$. Notice that, because the rules are applied in a parallel and maximal way as described in Sect. 2, the behavior of this P system is deterministic, and no other sets of rules can be applied in each configuration.

- If $m = 0$, then $m \cdot n = 0$, and the following computation takes place:

$\mathbb{C}_0 = [[k_1]_1^0[\ ]_2^0 b^n]_0^0$
$\mathbb{C}_1 = [[k_2 y_0]_1^0[\ ]_2^0 b_1^n]_0^0$
$\mathbb{C}_2 = [[\ ]_1^0[\ ]_2^0 b_2^n y_0]_0^0$
$\mathbb{C}_3 = [[\ ]_1^0[\ ]_2^0 b_3^n f_3]_0^- rem$
$\mathbb{C}_4 = [[\ ]_1^0[\ ]_2^0 f_4]_0^-$
$\mathbb{C}_5 = [[\ ]_1^0[\ ]_2^0]_0^0 f$

The computation is then done in 5 iterations, and because the output is zero copies of object d, the result is correct.

- If $m = 1$, then $m \cdot n = n$ and:

$\mathbb{C}_0 = [[ak_1]_1^0[\ ]_2^0 b^n]_0^0$
$\mathbb{C}_1 = [[my_1 y_0 k_2]_1^0[\ ]_2^0 b_1^n]_0^0$
$\mathbb{C}_2 = [[fk_3]_1^0[\ ]_2^0 b_2^n f]_0^0$
$\mathbb{C}_3 = [[\ ]_1^0[f_1]_2^0 b_3^n f_1]_0^0$
$\mathbb{C}_4 = [[\ ]_1^0[\ ]_2^- b_4^n f_2 rem]_0^- rem$
$\mathbb{C}_5 = [[\ ]_1^0[f_3]_2^- d^n f_3]_0^-$
$\mathbb{C}_6 = [[\ ]_1^0[f_4]_2^- f_4]_0^- d^n$
$\mathbb{C}_7 = [[\ ]_1^0[\ ]_2^0 rem]_0^0 f d^n$

The computation is then done in 7 iterations, and because the output is n copies of object d, the result is correct.

- If $m = 2p$ with $p \geq 1$ and we follow the Russian peasant multiplication algorithm, then the result of the partial computation doesn't contribute directly to the final result. In our system, this means that no objects d will be saved in membrane $[\ ]_2$:

$\mathbb{C}_0 = [[a^{2p} k_1]_1^0[\ ]_2^0 b^n]_0^0$
$\mathbb{C}_1 = [[a_1^p y_1^{2p} y_0 k_2]_1^0[\ ]_2^0 b_1^n]_0^0$
$\mathbb{C}_2 = [[a_2^p k_3]_1^0[\ ]_2^0 b_2^n]_0^0$
$\mathbb{C}_3 = [[a_3^p k_4]_1^0[\ ]_2^0 b_3^n]_0^0$
$\mathbb{C}_4 = [[a_4^p k_5]_1^0[\ ]_2^0 b_4^n]_0^0$
$\mathbb{C}_5 = [[a_5^p k_6]_1^0[\ ]_2^0 c^{2n}]_0^0$
$\mathbb{C}_6 = [[a^p k_1]_1^0[\ ]_2^0 b^{2n}]_0^0$

The computation goes from multiplying by $m = 2p$, represented by objects a, to multiplying by $m = p$ in 6 iterations. We can see also that no objects d were created, and that n, represented by objects b, is raised to $2n$ as in the Russian peasant multiplication algorithm.

- If $m = 2p + 1$ with $p \geq 1$ and we follow the Russian peasant multiplication algorithm, then in this case the result of the partial computation contributes directly to the final result. In our system, this means that n objects d will be saved in membrane $[\ ]_2$:

$$\mathbb{C}_0 = [[a^{2p+1}k_1]_1^0[\ ]_2^0 b^n]_0^0$$
$$\mathbb{C}_1 = [[a_1^p\ m\ y_1^{2p+1}\ y_0\ k_2]_1^0[\ ]_2^0 b_1^n]_0^0$$
$$\mathbb{C}_2 = [[a_2^p k_3]_1^0[\ ]_2^0 b_2^n\ m]_0^0$$
$$\mathbb{C}_3 = [[a_3^p k_4]_1^0[\ ]_2^0 b_3^n\ m_1]_0^+ rem$$
$$\mathbb{C}_4 = [[a_4^p k_5]_1^0[d^n]_2^0 c^{2n}\ m_2]_0^+$$
$$\mathbb{C}_5 = [[a_5^p k_6]_1^0[d^n]_2^0 b^{2n}\ m_3]_0^+$$
$$\mathbb{C}_6 = [[a^p k_1]_1^0[d^n]_2^0 b^{2n}]_0^0 rem$$

The computation goes from multiplying by $m = 2p + 1$, represented by objects a, to multiplying by $m = p$ in 6 iterations. We can see also that n objects d were created in membrane $[\ ]_2^0$, and that n, represented by objects b, is raised to $2n$ as in the Russian peasant multiplication algorithm.

Finally, if we observe rules RS_{40} and RS_{41}, we see that the objects d will be extracted from $[\ ]_2^0$ when the whole computation is finished, giving the result as output.

It takes 6 iterations to reduce the problem of multiplying by m to the problem of multiplying by $\lfloor m/2 \rfloor$, this is, to complete an iteration of the while loop in Algorithm 2. Because n doesn't influence the number of operations, we have an upper bound over the number of transition steps of $1 + 6 \cdot \lceil log_2(m) \rceil$.

For the P system defined in Sect. 4.1, we take advantage of the previous bound by always doing $m = z_i^k$ and $n = p_i^k$ or $n = \sum_{j \in S^k} \lceil \hat{p}_j^k \rceil_+$, depending on the multiplication we have to compute. Because $0 \leq z_i^k \leq 100$, we have that if we multiply m by any number, the number of iterations will be bounded by $1 + 6 \cdot \lceil log_2(100) \rceil = 43$.

In the P system defined in Sect. 4.1, the membranes $MULT_{i,k}$ will be almost exactly as described, with the structure changing from $\mu = [[\ ]_1^0[\ ]_2^0]_0^0$ to $\mu' = [[\ ]_{M1}^0[\ ]_{M2}^0]_{MULT_{i,k}}^0$. Membranes $MULT2_{i,k}$ will have $\mu'' = [[\ ]_{M1'}^0[\ ]_{M2'}^0]_{MULT2_{i,k}}^0$ as membrane structure, and will return objects d_i instead of d (rules RS_{40}, RS_{41}) and objects f_1 instead of f (rule RS_{39}).

Acknowledgements M.A.G.N. acknowledges the support by the European Union HORIZON-CL4-2021-HUMAN-01-01 under grant agreement 101070028 (REXASI-PRO) and by TED2021-129438B-I00 / AEI/10.13039/501100011033 / Unión Europea NextGeneration EU/PRTR.

References

Brown GW, von Neumann J (1951) 6. SOLUTIONS OF GAMES BY DIFFERENTIAL EQUATIONS, Princeton University Press, Princeton 73–80. https://doi.org/10.1515/9781400881727-007

Butcher J (2016). Numerical methods for ordinary differential equations. https://doi.org/10.1002/9781119121534

Cameron PJ (1994) Combinatorics: topics techniques, algorithms. Cambridge University Press, Cambridge

Cardona M, Colomer MÀ, Margalida A, Palau A, Pérez-Hurtado I, Pérez-Jiménez MJ, Sanuy D (2011) A computational modeling for real ecosystems based on P systems. Nat Comput 10(1):39–53. https://doi.org/10.1007/S11047-010-9191-3

Colomer M, Lavin S, Marco I, Margalida A, Pérez-Hurtado I, Pérez-Jiménez M, Sanuy D, Serrano E, Valencia-Cabrera L (2010) Modeling population growth of Pyrenean Chamois (Rupicapra P. Pyrenaica) by using P-systems, Membrane Computing 11th International Conference. Lect Notes Computer Sci 6501:144–159. https://doi.org/10.1007/978-3-642-18123-813

Colomer MA, Margalida A, Sanuy D, Pérez-Jiménez MJ (2011) A bio-inspired computing model as a new tool for modeling ecosystems: the avian scavengers as a case study. Ecol Model 222(1):33–47. https://doi.org/10.1016/j.ecolmodel.2010.09.012

Debreu G (1952) A social equilibrium existence theorem. Proc Natl Acad Sci United States of America 38(10):886–893

Facchinei F, Kanzow C (2007) Generalized Nash equilibrium problems. 4OR - A Quart J Oper Res 5(3):173–210. https://doi.org/10.1007/S10288-007-0054-4

García-Quismondo M, Reed JM, Chew FS, del Amor MAM, Pérez-Jiménez MJ (2017) Evolutionary response of a native butterfly to concurrent plant invasions: simulation of population dynamics. Ecol Model 360:410–424. https://doi.org/10.1016/j.ecolmodel.2017.06.030

García-Quismondo M, Gutiérrez-Escudero R, Pérez-Hurtado I, Pérez-Jiménez MJ, Riscos-Núñez A (2009) An overview of P-lingua 2.0, in: G. Păun, M. J. Pérez-Jiménez, A. Riscos-Núñez, G. Rozenberg, A. Salomaa (Eds.), Membrane computing 10th international workshop 264–288. https://doi.org/10.1007/978-3-642-11467-020

García-Victoria P, Cavaliere M, Gutiérrez-Naranjo MA, Cárdenas-Montes M (2022) Evolutionary game theory in a cell: a membrane computing approach. Inform Sci 589:580–594. https://doi.org/10.1016/J.INS.2021.12.109

Guo P, Quan C, Ye L (2019) Upsimulator: a eneral P system simulator. Knowl-Based Syst 170:20–25. https://doi.org/10.1016/j.knosys.2019.01.013

Guo P, Quan C, Ye L (2018) A simulator for cell-like P system. In: Qiao J, Zhao X, Pan L, Zuo X, Zhang X, Zhang Q, Huang S (eds) Bio-inspired computing: theories and applications. Springer Singapore, Singapore. https://doi.org/10.1007/978-981-13-2826-8_20

Gutiérrez-Naranjo MA, Pérez-Jiménez MJ, Ramírez-Martínez D (2008) A software tool for verification of spiking neural P systems. Nat Comput 7(4):485–497. https://doi.org/10.1007/S11047-008-9083-Y

Hofbauer J, Sigmund K (2000) Evolutionary games and population dynamics. J Amer Stat Assoc. https://doi.org/10.2307/2669431

Martinez-Piazuelo J, Ocampo-Martinez C, Quijano N (2022) Generalized Nash equilibrium seeking in population games under the Brown-von Neumann-Nash dynamics, 2022 European Control Conference 2161–2166. https://doi.org/10.23919/ECC55457.2022.9838437

Martínez-del-Amor MA, García-Quismondo M, Macías-Ramos LF, Valencia-Cabrera L, Riscos-Núñez A, Pérez-Jiménez MJ (2015) Simulating P systems on GPU devices: a survey. Fundam Informaticae 136(3):269–284. https://doi.org/10.3233/FI-2015-1157

Nash J (1951) Non-cooperative games. Annals Math 54(2):286–295

Pérez-Hurtado I, Valencia-Cabrera L, Pérez-Jiménez M, Colomer M, Riscos-Núñez A (2010) Mecosim: A general purpose software tool for simulating biological phenomena by means of P systems, IEEE Fifth international conference on bio-inspired computing: theories and applications (BIC-TA) 637–643. https://doi.org/10.1109/BICTA.2010.5645199

Păun Gh (2000) Computing with membranes. J Computer Syst Sci 61(1):108–143. https://doi.org/10.1006/jcss.1999.1693

Păun Gh (2002) Membrane Computing. Springer-Verlag, Germany

Păun Gh (2001) P systems with active membranes: attacking NP-complete problems, J Automata, LangCombin 6(1): 75-90. https://doi.org/10.25596/jalc-2001-075

Păun Gh, Rozenberg G, Salomaa A (eds) (2010) The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford, England

Sandholm WH (2010) Population games and evolutionary dynamics. MIT Press

Song B, Li K, Orellana-Martín D, Pérez-Jiménez MJ, Pérez-Hurtado I (2021) A survey of nature-inspired computing: membrane computing. Assoc Comput Mach Comput Surv 54(1):1–31. https://doi.org/10.1145/3431234

Zhang G, Pérez-Jiménez MJ, Riscos-Núñez A, Verlan S, Konur S, Hinze T, Gheorghe M (2021) P systems implementation on GPUs. In: Membrane Computing models: implementations. Springer, Singapore, 163-215. https://doi.org/10.1007/978-981-16-1566-56