P Systems with Replicator Dynamics: A Proposal

Matteo Cavaliere[1], Miguel A. Gutiérrez-Naranjo[2]

[1] Spanish National Biotechnology Centre
(CNB-CSIC), Madrid 28049, Spain
mcavaliere@cnb.csic.es
[2] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012, Sevilla, Spain
magutier@us.es

Summary. This short note proposes some ideas for considering evolutionary game theory in the area of membrane computing.

1 Introduction

Evolutionary game theory is a field started by J. Maynard Smith [4] with the aim of modelling the evolution of animal behavior by using game theory. Replicator dynamics [2] is a specific type of evolutionary dynamics where individuals, called replicators, exist in several different types. Each type of individual uses a pre-programmed strategy and passes this behavior to its descendants without modification. Replicator dynamics is one of the is one of most used approach to define the evolutionary dynamics of a population.

The main idea of the mechanism the following one. One assumes a population of players (individuals/organisms) interacting in a game composed by several possible strategies. Individuals have fixed strategies. The players randomly interact with other individuals (if space is considered, then the interactions are done according to the defined structure). Each of these encounters produces a payoff for the two individuals that depend on their strategies and on the payoff matrix that defines the game. The payoff of all these encounters are added up. Payoff is interpreted as fitness (reproductive success). Strategies that do well reproduce faster. Strategies that do poorly are outcompeted.

In this note we propose the possibilities of consider replicator dynamics in the framework of Membrane Computing (P Systems), [3].

We imagine two possibilities. The first one is using replication dynamics as "evolution" rules of a membrane system. A second possibility consists in "simulating" replication dynamics by using the tools and the notions provided by the membrane computing paradigms.

We believe that both possibilities could be sources of new kinds of problems for the area.

2 Using Replicator Dynamics in P Systems

As a simple example of replication dynamics let us consider the following payoff matrix of a well-known game, the *prisoner's dilemma*, [2].

	cooperate	defect
cooperate	$4, 4$	$1, 5$
defect	$5, 1$	$2, 2$

This is read in the following way. When a cooperator meets another cooperator, they both gets 4. If a cooperator meets a defector, the cooperator gets 1 and the defector 5. If two defectors meet, they both gets 2. If we have a population of n individuals, k of them being cooperators (symbol c) and $n - k$ being defectors (symbol d) then the population is updated in the following manner (*one step of the evolutionary dynamics*).

Each object c receives a payoff that is the sum of all the payoffs obtained by considering the meetings with all other players. In this case the payoff accumulated by each single c is: $4(k - 1) + 1(n - k)$. In the same manner, each d receives $2(n - k - 1) + 5k$. The replication dynamics impose that each object (c or d) replicates (produce off-springs) with a rate function of the obtained payoff (in other words, the payoff is interpreted as fitness, [4]).

The simplest approach could then assume that each object c divides in $4(k - 1) + 1(n - k)$ copies (off-springs), while each d divides in $2(n - k - 1) + 5k$ copies. This means that each c produces $4(k-1)+1(n-k)$ copies of c and each d produces $2(n - k - 1) + 5k$ copies of d.

Moreover, one can also assume that, at each step, a certain number of objects is removed from the population. The simplest scenario is to assume a death/removal rate that indicates the number of objects (constant) removed at each step. In a more complex scenario, the removal, as the replication, could depend on the accumulated payoff (e.g., the players with worst fitness are removed). Many variants have been considered [2].

In P systems, there is the notion of compartment that has been shown to be relevant for the evolutionary dynamics of a population [2]. In this respect, there are many examples that show that the evolutionary dynamics can be very different when observed in structured populations and in homogeneous populations (e.g., [2]). One could then consider a P system where the objects in the compartments represent the individuals (players) of a population. Each object indicates (is associated to) the strategy of a certain chosen game (for instance, in the case of the prisoner's dilemma (PD), we have objects c and d).

The population (e.g, a multiset over the alphabet c and d in the PD game) evolves, in parallel, in the compartments, according to the replicator dynamics.

Specifically, the payoff matrix is used to calculate the payoff for each individual object (as described above), by considering all other objects present in the same compartment. Then, based on these obtained payoffs, one decides which objects to replicate and which objects to remove. For instance, this could be done using thresholds (e.g., *if payoff* $>$ *..then replicates*, *if payoff* $<$ *..then the object is removed*). Each object is then replicated (e.g., a c creates more copies of c, a d creates more copies of d) or is removed based on such threshold and on its obtained payoff. Target indications could be used to move the created objects across compartments. The number of objects in a certain compartment could be naturally interpreted as output produced. However, the programmability of such device remains an open issue. In fact, notice that, differently from standard P systems, the rules here cannot be programmed – they are "naturally" assigned by the evolutionary dynamics.

3 Simulating Replicator Dynamics

The second possibility is to program the replication dynamics using the tools available in the membrane computing area. The task is non-trivial, in particular to implement the payoff-based replication that is naturally present in the replicator dynamics.

We propose a first solution where any individual produces a new set of individuals identical to the original, at each time unit according to a discrete global clock. The number of off-spring depends on the number of encounters with defectors and cooperators and their corresponding payoffs.

We suggest a family of P systems for dealing with prisoner's dilemmas in its most general form (however, the approach proposed here can be generalized to different games). The family of P systems considers the following initial situation: A population of n individuals, k of them being cooperators (c) and $n-k$ being defectors (d). Let us consider four non negative integers R, S, T and P and the following general payoff matrix for the prisoner's dilemma.

	cooperate	defect
cooperate	R,R	S,T
defect	T,S	P,P

As standard in the area, [2], we use the terms R, *reward*, P, *punishment*, T, *temptation* and S, *sucker's payoff*. Hence, the 4-uple $PD \equiv \langle R, S, T, P\rangle$ can encode the game.

We assume the simplest replication mechanism where each individual c or d is substituted in the next *stage* (by using *mitosis* or whatever replication mode) by as many objects of the same type as its *payoff*. In other words, if c_n and d_n is the number of individuals of type c and d in the stage n, then

$$\begin{aligned} c_0 &= k \\ d_0 &= n - k \\ c_{n+1} &= R\,(c_n - 1) + S\,d_n \\ d_{n+1} &= T\,c_n + P\,(d_n - 1) \end{aligned}$$

In the membrane computing framework one can consider rules of type $c \to c^{\alpha}$ and $d \to d^{\beta}$. This reproduces the idea of replication of the original individuals. The drawback is, of course, than α and β depends on the number of individuals of the current configuration. This idea leads us to consider a set of rules $c \to c^{\alpha_1}$, $c \to c^{\alpha_2}$, $c \to c^{\alpha_3}$, ..., but even in case of having an *oracle* which decides the right rule in each configuration, we will need a potentially infinite amount of rules.

We propose an alternative solution that uses a P systems family (a P system for each 4-uple $\langle R, S, T, P\rangle$ in the framework of P systems with active membranes, [3], that computes $\{c_n, d_n\}_{n\in\mathbb{N}}$). The proposed systems have been checked with the SCPS simulator [1]. As usual in this P system model, each membrane can be crossed out by a unique object (at most) in each computation step. This feature will be used to control the flow of objects between regions.

Given a 4-uple $PD \equiv \langle R, S, T, P\rangle$ encoding a prisoner's dilemma, let us consider the following P system

$$\Pi_{PD} = \langle \Gamma, H, EC, \mu, w_e, w_s, R\rangle$$

where

- The alphabet of objects is $\Gamma = \{c, c_*, c_a, c_1, c_2, c_3, d, d_*, d_a, d_1, d_2, d_3, z, z_1, z_2, z_3, z_4\}$;
- $H = \{e, s\}$ is the set of labels;
- $EC = \{q0, q1, q2, q3, q4, qc, qd, qcc, qdd\}$ is the set of electrical charges;
- the membrane structure has only two membranes, the skin with label s an an elementary membrane with label e, $\mu = [\,[\,]_e^{q0}\,]_s^{q0}$;
- the initial multisets are $w_e = z$ and $w_s = \varnothing$. We also consider as *input*, the *population* of objects c^k and d^{n-k}. They will be placed in membrane e in the initial configuration.

We will also consider the following sets of rules

$$\begin{aligned} R_1 &\equiv [\,z\,]_e^{q0} \to [\,z_1]_e^{q1}\,[\,z_1\,]_e^{q2} \\ R_2 &\equiv [\,z_1 \to \lambda\,]_e^{q1} \\ R_3 &\equiv [\,z_1 \to \lambda\,]_e^{q2} \\ R_4 &\equiv [\,c]_e^{q1} \to c\,[\,]_e^{q3} \\ R_5 &\equiv [\,d]_e^{q1} \to d\,[\,]_e^{q3} \end{aligned}$$

In the initial configuration we have only one membrane e with the population of objects c and d and one extra object z. This extra object z produces the division (R_1) of the membrane. We have two copies of the population: one with charge $q1$ and the second one with charge $q2$.

The main idea is that all the objects in the membrane e with charge $q1$ will pass sequentially to membrane with charge $q2$. In this second membrane the *payoffs* will

be computed. The charges will be used as traffic-lights in order to control the flow of objects.

$$
\begin{aligned}
R_6 &\equiv c\,[\,]_e^{q2} \to [\,c_1\,]_e^{qc} \\
R_7 &\equiv d\,[\,]_e^{q2} \to [\,d_1\,]_e^{qd} \\
R_8 &\equiv [\,c\,]_e^{qc} \to z_4\,[\,]_e^{qcc} \\
R_9 &\equiv [\,d\,]_e^{qd} \to z_4\,[\,]_e^{qdd}
\end{aligned}
$$

When an object c or d arrives to the membrane with label $q2$ with R_6 or R_7, the calculation of the *payoff* starts. Since an individual does not meet itself in order to get a payoff, an object c or d is sent out of the membrane (R_8 or R_9).

$$
\begin{aligned}
R_{10} &\equiv [\,c_1 \to c_2 c_3\,]_e^{qc} \\
R_{11} &\equiv [\,d_1 \to d_2 d_3\,]_e^{qd} \\
R_{12} &\equiv [\,z_4 \to \lambda\,]_s^{q0}
\end{aligned}
$$

These rules $R_{10} - R_{12}$ are technical rules in order to adjust the proposed P system to the model of active membranes, where rules $c\,[\,]_e^{q2} \to [\,c_2 c_3\,]_e^{qc}$ or $[\,c\,]_e^{qc} \to \lambda\,[\,]_e^{qcc}$ are not allowed. The computation of the payoff is performed by the following rules:

$$
\begin{aligned}
R_{13} &\equiv [\,c \to c_*^R\,c\,]_e^{qcc} \\
R_{14} &\equiv [\,d \to c_*^S\,d\,]_e^{qcc} \\
R_{15} &\equiv [\,c \to d_*^T\,c\,]_e^{qdd} \\
R_{16} &\equiv [\,d \to d_*^P\,d\,]_e^{qdd}
\end{aligned}
$$

The charge qcc can be interpreted as the visit of an individual c. The objects c in the membrane produce R copies of c_* and all the objects d produce S copies of d_*. Analogously, the charge qdd can be interpreted as the visit of an individual d. In this case, the objects c in the membrane produce T copies of c_* and all the objects d produce P copies of d_*.

The path to complete the cycle and to start again begins with the following rules. An object z_2 is sent to the first membrane labeled with e in order to get a new individual for the calculation of the payoff.

$$
\begin{aligned}
R_{17} &\equiv [\,c_2\,]_e^{qcc} \to z_2\,[\,]_e^{q2} \\
R_{18} &\equiv [\,d_2\,]_e^{qdd} \to z_2\,[\,]_e^{q2} \\
R_{19} &\equiv z_2\,[\,]_e^{q3} \to [\,z_2\,]_e^{q1}
\end{aligned}
$$

The object c or d sent out by the rule R_8 or R_9 is placed again on the corresponding membrane by rule R_{20} or R_{21}.

$$
\begin{aligned}
R_{20} &\equiv [\,c_3 \to c\,]_e^{q2} \\
R_{21} &\equiv [\,d_3 \to d\,]_e^{q2}
\end{aligned}
$$

Sending z_2 into the corresponding membrane opens the traffic light by changing the charge to q_1. The cycle starts again and rules R_4 and R_5 can be triggered again, if any object c or d remains in the membrane. In order to control the behavior of the membrane when all the objects c and d have been sent out, we add some technical rules.

$R_{22} \equiv [\, z_2 \rightarrow z_3 \,]_e^{q1}$
$R_{23} \equiv [\, z_3 \rightarrow \lambda \,]_e^{q3}$

If z_3 appears in a membrane, it means that all objects c or d have been sent out in previous steps. In this case, the membrane can be dissolved and the cycle of computing the payoffs is completed.

$R_{24} \equiv [\, z_3 \,]_e^{q1} \rightarrow z_3$
$R_{25} \equiv z_3 \, [\,]_e^{q2} \rightarrow [\, z_3 \,]_e^{q4}$

When an object z_3 goes into the membrane with label e, the *old* objects c and d are removed and the objects c_* and d_* become the new population.

$R_{26} \equiv [\, c_* \rightarrow c_a \,]_e^{q4}$
$R_{27} \equiv [\, d_* \rightarrow d_a \,]_e^{q4}$
$R_{28} \equiv [\, z_3 \rightarrow z \, z_4 \,]_e^{q4}$
$R_{29} \equiv [\, c_a \rightarrow c \,]_e^{q4}$
$R_{30} \equiv [\, d_a \rightarrow d \,]_e^{q4}$
$R_{31} \equiv [\, c \rightarrow \lambda \,]_e^{q4}$
$R_{32} \equiv [\, d \rightarrow \lambda \,]_e^{q4}$

Finally, we change the charge of the membrane e and a new *stage* can start

$R_{33} \equiv [\, z_4 \,]_e^{q4} \rightarrow z_4 \, [\,]_e^{q0}$

3.1 Overview of the Computation

The main idea of the design is to consider two copies of the population. The first copy (which acts as a counter) sends individuals to the second one: when all the objects have been sent, the computation of all payoffs is completed and we finish a *cycle*. In the second copy, the payoffs are computed and stored. For each object, the P system takes five computational steps in order to calculate its payoff.

We start with the initial configuration $C_0 = [\, [\, z c^k d^{n-k} \,]_e^{q0} \,]_s^{q0}$. Initially, the two copies of the population are created by applying the rule R_1, $C_1 = [\, [\, z_1 c^k d^{n-k} \,]_e^{q1} \, [\, z_1 c^k d^{n-k} \,]_e^{q2}]_s^{q0}$. The first new membrane, with label $q1$ will send objects to the second one with label $q2$. At this moment, rules R_4 and R_5 can be non-deterministically applied, but, due to the semantics of active membranes, only one of them is chosen. Let us suppose that R_3 is taken (the other case is analogous) and we reach $C_2 = [\, [\, c^{k-1} d^{n-k} \,]_e^{q3} \, [\, c^k d^{n-k} \,]_e^{q2} \, c]_s^{q0}$. Notice that the label in

the first membrane has been changed to $q3$. Intuitively, this membrane is closed till the arrival of the object z_2 at step 6. Objects z_1 are removed.

In the next step, the object c in the skin is sent as c_1 into the second elementary membrane and changes the polarization, $C_3 = [\,[\,c^{k-1}d^{n-k}\,]_e^{q3}\,[\,c^k d^{n-k}c_1\,]_e^{qc}]_s^{q0}$. The process of computing the payoff of this object c_1 starts: c_1 is replaced by c_2c_3 and one object c is sent to the skin, changing again the polarization, $C_4 = [\,[\,c^{k-1}d^{n-k}\,]_e^{q3}\,[\,c^{k-1}d^{n-k}c_2c_3\,]_e^{qcc}\,z_4\,]_s^{q0}$. The computation of the payoff is made now by application in parallel of the rules R_{13} and R_{14}. In order to avoid that this rule can be applied in the next step, the object c_2 is sent out (as z_2) and the polarization changes again. According to R_{13} and R_{14}, R objects c_* are produced for each c and S objects c_* for each d, $C_5 = [\,[\,c^{k-1}d^{n-k}\,]_e^{q3}\,[\,c^{k-1}d^{n-k}c_3c_*^{R(k-1)+S(n-k)}\,]_e^{q2}\,z_2]_s^{q0}$. Finally, c_3 goes to c in the second elementary membrane and z_2 goes into the first one, changes the polarization and *opens* the membrane, $C_6 = [\,[\,c^{k-1}d^{n-k}z_2\,]_e^{q1}\,[\,c^k d^{n-k}c_*^{R(k-1)+S(n-k)}\,]_e^{q2}\,z_2]_s^{q0}$. Notice that we have again two membranes, one of them with charge $q1$ and the other one with charge $q2$ as in the configuration C_1. In the next steps the process goes on by sending all the objects from the first membrane to the second one, where the payoffs will be stored.

After $5n+1$ steps we arrive at the configuration

$$C_{5n+1} = [\,[\,z_2\,]_e^{q1}\,[\,c^k d^{n-k}c_*^{k(R(k-1)+S(n-k))}d_*^{(n-k)(Tk+P(n-k-1))}\,]_e^{q2}\,]_s^{q0}$$

No more objects are left in the first membrane. In C_{5n+2} we have an object z_3 inside a membrane with label e and charge q_1. In the next step, the rule r_{24} is applied and the membrane is dissolved,

$$C_{5n+3} = [\,[\,c^k d^{n-k}c_*^{k(R(k-1)+S(n-k))}d_*^{(n-k)(Tk+P(n-k-1))}\,]_e^{q2}\,z_3]_s^{q0}$$

The object z_3 goes into the elementary membrane and changes the polarization to q_4,

$$C_{5n+4} = [\,[\,c^k d^{n-k}c_*^{k(R(k-1)+S(n-k))}d_*^{(n-k)(Tk+P(n-k-1))}z_3\,]_e^{q4}\,]_s^{q0}$$

This new polarization produces the updating of the payoff to a new population in two steps, so

$$C_{5n+6} = [\,[\,c^{k(R(k-1)+S(n-k))}d^{(n-k)(Tk+P(n-k-1))}\,]_e^{q4}\,z_4\,]_s^{q0}$$

which is analogous to the configuration C_0, so a new stage starts (the object z_4 disappears in the next step by using the rule R_{12}).

4 Conclusions and Future Work

Replicator dynamics in one of most used mechanisms in evolutionary game theory. In this context, several papers have shown the relevance of compartments and structures. On the other hand, membrane computing explicitly investigates the

relevances of compartments for computation. A natural possibility, proposed in this note, is to bridge these two areas. We have sketched two possibilities. The first one consists in using rules inspired from the replicator dynamics. A second one consists in programming the replicator dynamics using the tools of membrane computing. In this case, we have presented a possible solution using membrane systems with active membranes. However several other solutions can be imagined, in particular replacing the cell-like membrane structure by a tissue-like structure could allow a simpler version of the simulations.

Acknowledgments

MAGN acknowledges the support of the projects TIN-2009-13192 of the Ministerio de Ciencia e Innovación of Spain and the support of the Project of Excellence of the Junta de Andalucía, grant P08-TIC-04200. M. C. acknowledges the support of the program JAEDoc15 ("Programa junta para la ampliacion de estudios") and of the Research Group on Natural Computing of the University of Sevilla.

References

1. Gutiérrez-Naranjo, M.A., Riscos-Núñez, A., Pérez-Jiménez, M.J.: A simulator for confluent P systems. In: Gutiérrez-Naranjo, M.A., Riscos-Núñez, A., Romero-Campero, F.J., Sburlan, D. (eds.) Third Brainstorming Week on Membrane Computing. pp. 169–184. Fénix Editora, Sevilla, Spain (2005)
2. Hofbauer, J., Sigmund, K.: *Evolutionary Games and Population Dynamics*. Cambridge University Press (Jun 1998)
3. Păun Gh., Rozenberg G., Salomaa A. Eds.: *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.
4. Smith, J.M.: *Evolution and the Theory of Games*. Cambridge University Press, 1st edition edn. (Dec 1982)