The Influence of Asynchronous Dynamics in the Spatial Prisoner's Dilemma Game

Carlos Grilo[1,2] and Luís Correia[2]

[1] Dep. Eng. Informática, Escola Superior de Tecnologia e Gestão
Instituto Politécnico de Leiria
grilo@estg.ipleiria.pt
[2] LabMag, Dep. Informática, Faculdade de Ciências da Universidade de Lisboa
Luis.Correia@di.fc.ul.pt

Abstract. We examine the influence of asynchronism in the Spatial Prisoner's Dilemma game. Previous studies reported that less cooperation is achieved with the asynchronous version of the game than with the synchronous one. Here, we show that, in general, the opposite is the most common outcome. This conclusion is only possible because a larger number of scenarios was tested, namely, different interaction topologies, a transition rule that can be tuned to emulate different levels of determinism in the choice of the next strategy to be adopted and different rates of asynchronism. The influence of stochastic and deterministic periodic updating in the outcome of the system is also compared. We found that these two update disciplines lead basically to the same result. This is an important issue in the simulation of social and biological behavior.

1 Introduction

Spatial evolutionary games are used in the area of evolutionary game theory as models to study, for example, how could cooperation ever emerge in nature and human societies [11]. They are also used as models to study how can cooperation be promoted and sustained in artificial societies [9]. In these models, a structured population of agents interacts during several time steps through a given game which is used as a metaphor for the type of interaction that is being studied. The population is structured in the sense that each agent can only interact with its neighbors. After each interaction session, some or all the agents, depending on the update method used, have the possibility to change their strategies. This is done using a so called transition rule that models the fact that agents tend to adapt their behavior to the context in which they live by imitating the most successful agents they know. It can also be interpreted as the selection step of an evolutionary process in which the least successful strategies tend to be replaced by the most successful ones.

The final outcome of these models, that is, the proportion of cooperating agents eventually achieved, can be influenced by, for example, the game that is being used, the interaction topology, the transition rule or the update method. The most used game in this area is the Prisoner's Dilemma game (see section 2.1).

M. Asada et al. (Eds.): SAB 2008, LNAI 5040, pp. 362–371, 2008.

There are some works where the influence of some of those aspects on the spatial version of this game is studied. For example, in [10] the influence of the interaction topology is analyzed. Also, in [7] the influence of the interaction topology, the transition rule and the update method are studied.

In this area, the discussion about using synchronous or asynchronous update methods started with a paper by Huberman and Glance [5]. Synchronous updating means that, at each time step, the revision of strategies happens for all agents simultaneously, while this is not the case for asynchronous updating. In that paper the authors contested the results achieved in [8] by Nowak and May who showed that cooperation can be maintained when the game is played in a regular 2-dimensional grid by agents which do not remember their neighbors' past actions. Huberman and Glance criticized the fact that the model used in [8] was a synchronous one, which is an artificial feature. They also presented the results of simulations where cooperation was no longer sustainable when an asynchronous updating was used. After this work, in [7] Nowak *et al* tested their model under several conditions, including synchronous and asynchronous updating and showed that cooperation can be maintained for many different conditions, including asynchronism. However, the results are presented through system snapshot images, which render difficult to measure the way they are affected by the modification from synchronous to asynchronous updating. Recently, in [6], a similar scenario was studied using various asynchronous update methods besides synchronous updating. The authors found that *"The most notable difference between the synchronous and asynchronous schemes, is that the synchronous updating scheme supports more cooperators than the other updating schemes"*.

In this paper, we show that, in general, more cooperation is achieved with the asynchronous version of the Spatial Prisoner's Dilemma game than with the synchronous one. The conclusions derived in [5] and [6] result from the fact that a limited number of conditions were tested, namely, the utilization of only the best-neighbor transition rule, according to which an agent always imitates the strategy of its most successful neighbor. Here, we use the transition rule used in [7] (see Section 2.3). This rule can be tuned to cover the spectrum between proportional updating, with which agents can imitate strategies other than the one used by their most successful neighbor, and the best neighbor rule. Different types of interaction topologies were also used so that the conclusions derived can have a more general character.

In what concerns to the update methods, we first used an asynchronous stochastic update method [2] that allows us to cover the spectrum from synchronous to sequential updating. Usually, asynchronous updating is understood as sequential updating, which means that, at each time step, only one agent updates its strategy after interacting with its neighbors. But, reality seems to lie somewhere between these two extremes and, so, sequential updating can be considered as artificial as synchronous updating. In a population of interacting agents, many interaction and decision processes can be occurring at the same time but not necessarily involving all the agents. If both were instantaneous

phenomena we could model the dynamics of the system as if they occurred one after another but that is not usually the case. These processes can take some time, which means that their output is not available to other ongoing decision processes. Even if we consider them as being instantaneous, the time that information takes to be transmitted and perceived implies that their consequences are not immediately available to other agents.

We also compare the results achieved using the stochastic update method with the ones achieved with a deterministic periodic one. There are numerous examples of periodic behaviors where not all the population is necessarily synchronized [3][6]. This comparison can help us understand to what extent the results depend on this type of updating.

The paper is structured as follows: in Sec. 2 we describe the model we used in our simulations and in Sec. 3 we present and discuss the results. Finally, in Sec. 4 some conclusions are drawn and future work is advanced.

2 The Model

2.1 The Prisoner's Dilemma Game

In the Prisoner's Dilemma game (PD), players can cooperate (C) or defect (D). The payoffs are the following: R to each player if they both play C; P to each if they both play D; T and S if one plays D and the other C, respectively. These values must obey $T > R > P > S$ and $2R > T + S$. It follows that there is a strong temptation to play D. But, if both play D, which is the rational choice or the Nash equilibrium of the game, both get a smaller payoff than if they both play C, hence the dilemma. For practical reasons, the payoffs are usually defined as $R = 1$, $T = b > 1$ and $S = P = 0$, where b represents the advantage of D players over C ones when they play the game with each other. This has the advantage that the game can be described by only one parameter without loosing its essence [7].

2.2 Population Topologies

We used two types of topologies: *small-world networks* (SWNs) [13] and *scale-free networks* (SFNs) [1]. We build SWNs as in [12]: first, a toroidal regular 2-dimensional grid is built so that each node is linked to its 8 surrounding neighbors by undirected links; then, with probability ϕ, each link is replaced by another one linking two randomly selected nodes. Parameter ϕ is called the *rewiring probability*. In some works [7] self-links are allowed because it is considered that each node can represent not a single agent but a set of similar agents that may interact with each other. Here, we do not allow self-interaction since we are interested in modeling nodes as individual agents. Repeated links and disconnected graphs are also avoided. The rewiring process may create long range links connecting distant agents. For simplicity, we will call neighbors to all interconnected agents, even if they are not located at adjacent nodes. By varying ϕ from 0 to 1 we are able to build from completely regular networks to random

ones. SWNs have the property that, even for very small values of the rewiring probability, the mean path length between any two nodes is much smaller than in a regular network, maintaining however a high clustering coefficient observed in many real systems including social ones.

Although frequently used to model real networks, SWNs do not have a power law degree distribution that is frequently observed on such networks [10]. SFNs are network models that have this property since their degrees follow the power law distribution $P(k) \sim k^{-\gamma}$. We build SFNs in the following way: the network is initialized with m_0 fully connected nodes. Then we add nodes, one at a time, until the network has the desired size. Each added node is linked to m_0 already existing nodes so that the probability of creating a link with some existing node i is equal to $\frac{k_i}{\sum_j k_j}$, where k_i is the degree of i, that is, the number of nodes to which it is connected. This method of link creation is called *preferential attachment*, since the more links a node has, the greater is the probability of creating links to it. This has the effect that a little proportion of nodes has a big connectivity while the most part has a very low connectivity.

2.3 Interaction and Strategy Update

On each time step, agents first play a one round PD game with all their neighbors. Agents can only play C or D and the only way they can change their strategy is by way of the application of the transition rule, after the interaction process. This rule is used to model the fact that agents tend to imitate the most successful agents they know. In order to be able to model intermediate levels of asynchronism in the strategy update process, we use an update method called a*synchronous stochastic dynamics* (ASD)[2]. When ASD is used, at each time step, each agent has a given probability $0 < \alpha \leq 1$ of applying the transition rule in order to decide which strategy to use next. The α parameter is called the *synchrony rate* and is the same for all agents. When $\alpha = 1$ we have synchronous updating and as $\alpha \to \frac{1}{n}$, where n is the population size, ASD approaches sequential updating.

To model the strategy update process, we used a generalization of the *proportional* transition rule [7]. Let G_i be the average payoff earned by agent i in the second stage, N_i be the set of neighbors of agent i, s_i be equal to 1 if i's strategy is C and 0 otherwise, and d a positive number. According to this rule, the probability that an agent i adopts C as its next strategy is

$$p_C(i) = \frac{\sum_{l \in N_i \cup i} s_l (G_l)^d}{\sum_{l \in N_i \cup i} (G_l)^d}. \tag{1}$$

The d parameter acts as a weight that favors the most successful neighbor's strategy B in the update process: the bigger d, the larger is the probability that i adopts B. When $d \to +\infty$ we have a deterministic best-neighbor rule such that i always adopts B as its next strategy. When $d = 1$ we have the proportional update rule. It can be viewed, as well, as the *deterministic degree* of the transition rule. We use average payoffs instead of total payoffs because agents may have a different number of neighbors.

3 Simulations and Results

All the simulations were done with populations of $50 \times 50 = 2500$ agents, randomly initialized with 50% of Cs and 50% of Ds. When the system is running synchronously, i.e., when $\alpha = 1$, we let it run during a transient period of 900 iterations. After this, we let the system run during 100 more iterations and, at the end, we take as output the average proportion of cooperators during this period, which is called the *sampling period*. When $\alpha \neq 1$ the number of selected agents at each time step may not be equal to the size of the population and it may vary between two consecutive time steps. In order to guarantee that these runs are equivalent to the synchronous ones in what concerns to the total number of individual updates, we let the system first run until 900×2500 individual updates have been done. After this, we sample the proportion of cooperators during 100×2500 individual updates and we average it by the number of time steps needed to do these updates. Each simulation is a combination of the ϕ/m_0, α, b and d parameters, and all the possible combinations of the values shown in Table 1 were tested. For each combination, 30 runs were made and the average of these runs is taken as the output.

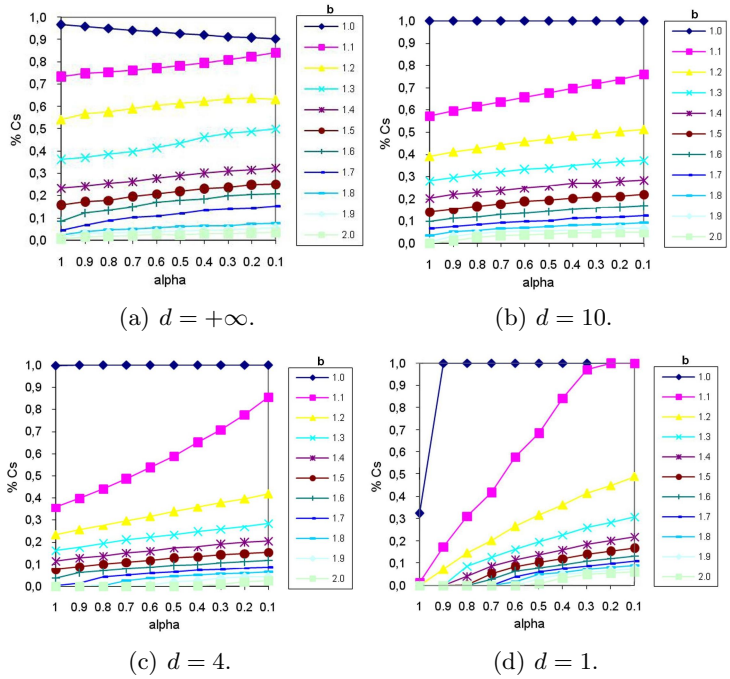
Table 1. Parameter values used in the simulations

Parameter	**Values**
ϕ (SWNs)	0 (regular), 0.01, 0.05, 0.1, 1 (random)
m_0 (SFNs)	4, 8, 12
α (ASD)	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1 (synchronous)
b (PD game)	1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2
d (transition rule)	$+\infty$ (best neighbor), 100, 10, 8, 6, 4, 2, 1 (proportional)

In [4] we presented results of simulations done with the ASD update method where only SWNs were used. In the next section, we present an analysis of the results achieved based also on simulations done with SFNs. Most of the conclusions apply to both types of topologies but we will point the differences where they exist. In order to help the understanding of the conclusions, in Fig. 1 we show four typical charts produced by the simulations. In Sec. 3.2 we then compare the results achieved using ASD with the ones achieved using a periodic deterministic update method.

3.1 ASD Results

We can conclude that, for most conditions, the system is not very sensitive to small changes in the α value (we consider that the system is sensitive when its outcome changes by more than 0.1 when the α value is changed by 0.1). There are, however, some situations of big sensitivity (Ex: Fig. 1d). Also, the results show that, in general, the system responds monotonically to changes in the α value. The few situations of non-monotonicity happen almost only for SWNs



(a) $d = +\infty$. (b) $d = 10$.

(c) $d = 4$. (d) $d = 1$.

Fig. 1. % of cooperators for $m_0 = 4$ (SFNs) and different combinations of d, b and α

and mainly for large values of d, that is, when the probability that an agent imitates its most successful neighbor is high. As to the influence of the interaction topology on these two aspects, the results show that the system becomes more robust and monotonous as ϕ and m_0 are increased. We hypothesize that the real reason for this coincidence of results can be due to the fact that the mean path length between the nodes decreases as we increase ϕ and m_0, respectively, on SWNs and SFNs. Another possibility would be the clustering degree of the network. However, while the clustering degree decreases on SWNs as we increase the ϕ value, it can not decrease on SFNs as we increase the value of m_0. So, this result can not be ascribed to this property. More work must be done, however, in order to confirm this.

There is a somewhat unexpected result that can be phrased like this: the lower the value of the d parameter, the more is cooperation favored when we decrease the value of α. That is, as we decrease the value of d, the slope of the curves increases. This also means that the system becomes more sensitive to α changes as the d parameter is decreased. We call this effect the *small determinism degree and small synchrony rate* effect.

Possibly, the most relevant result is that, as we decrease the α value, the proportion of cooperators increases for the big majority of the simulations. That is, in general, asynchronism supports more cooperators than synchronism. This conclusion can only be derived because several scenarios were tested, namely, different interaction topologies and different values for the determinism degree

of the transition rule. Indeed, this allowed us to verify that the situations where this conclusion does not apply are the ones where big values of the determinism degree of the transition rule, d, are used. Putting it another way, asynchronism is detrimental to the emergence of cooperation only when the probability of committing errors in the choice of the next strategy to use is very low. But reality is far from such perfection and phenomena like noise, be it deliberate or not, perception errors, or simply bad evaluations, often prevent agents from choosing what would seem the best choice.

Finally, we would like to stress that using an update method able to cover all the spectrum from synchronous to near sequential updating, as the ASD method is, allows a deeper analysis of the system being studied. It allows, for example, the identification of existing phase transitions: often the level of cooperation is 0 for $\alpha = 1$ and it remains there until a given $\alpha = c$ value is reached. Then, suddenly, the level of cooperation starts to increase, sometimes in a significative way, as α decreases from c to 0.1. For many of such situations c is very near to 1 or even equal to 1 (see, for example, Fig. 1d). This may suggest that in these cases the existence of some degree of cooperation is the most probable outcome in the system being modeled since it exists for almost the entire α domain.

3.2 Stochastic *Versus* Deterministic Updating

In [3] a classification of *random boolean networks* (RBNs) is given based mainly on the update methods used. The authors found that the behavior of asynchronous deterministic RBNs is much closer to the behavior of synchronous RBNs than to the one shown by asynchronous stochastic RBNs. In spite that our intention in this paper is not to make a comparison between the model studied here and RBNs, a question arises: is this also the case for spatial evolutionary games? An affirmative answer could help in the formulation of a general explanation for the influence of asynchronism on dynamical systems, if there is any. A negative answer is also important because, not only it takes this hypothesis off the path, but also because it can help us understanding the influence of deterministic updating on the modeled systems.

In order to model asynchronous deterministic dynamics, the authors of [3] used a sequential update method and another one (DGARBN)[1] that allows more than one agent to be updated at the same time. DGARBN is also a periodic update method: each agent has two parameters, p and q that, respectively, determine the period of the updating, and the phase (the translation of the update). Agents updated at the same time step are updated synchronously. The problem with this method is that it doesn't allow us to control the synchrony rate so that a direct comparison with the results achieved with ASD can be made. Besides, it doesn't provide the same number of updates for all agents when time grows. This means that agents can not be considered homogeneous anymore, which is a significant modification generating a different problem. Therefore, we changed the method so that the p parameter is the same for all agents. We call *asynchronous deterministic dynamics* (ADD) to this update method. As in the original method,

[1] DGARBN: Deterministic Generalized Asynchronous Random Boolean Networks.

$q < p$ and is defined randomly for each agent. This way, when $p = 2$, $p = 5$, or $p = 10$, respectively, 50%, 20% and 10% of the population is updated each time step, on average. These are the values we used in our simulations. They allow us to compare the results achieved with the ASD method for $\alpha = 0.5$, $\alpha = 0.2$ and $\alpha = 0.1$. It would be desirable that we could compare results for other α values, namely for $0.5 < \alpha < 1.0$, but we don't know about any deterministic and periodic update method that allows us to cover that interval.

Let us consider Res_{Sync}, Res_{ASD} and Res_{ADD} as the results achieved for a given combination of parameters with synchronous updating, ASD and ADD, respectively (each point in the charts of Fig. 1 corresponds to a given combination of parameters). In order to see which one of the two methods is closer to synchronous updating and how different they are, we first compute the quantities $dif_{ASD} = |Res_{ASD} - Res_{Sync}|$ and $dif_{ADD} = |Res_{ADD} - Res_{Sync}|$ for each combination of parameters. dif_{ASD} and dif_{ADD} are not considered in the following when $Res_{Sync} = Res_{ASD} = Res_{ADD}$ since in these cases it makes no sense to talk about which method is closer to synchronous updating. After this, we separately compute the averages of dif_{ASD} and dif_{ADD} values. Let us call these two quantities Avg_{ASD} and Avg_{ADD}, respectively. Finally, we compute the quantities $s = Avg_{ASD} - Avg_{ADD}$ and $r = x/y$, where $x = min(Avg_{ADD}, Avg_{ASD})$ and $y = max(Avg_{ADD}, Avg_{ASD})$. The s value tells us two things: which method is closer to synchronous updating and how different they are from each other in absolute terms. If $s > 0$, ADD is closer to synchronous updating than ASD and vice-versa. Also, as $s \to 0$ the two methods become closer. The r meaning is the following: if F and C are, respectively, the method that is farther and closer to synchronous updating, then $r < 0.5$ means that C is closer to synchronous updating than to F. If $r > 0.5$, C is closer to F than to synchronous updating.

The results are ($s = 0.011$, $r = 0.928$) for SWNs and ($s = 0.010$, $r = 0.922$) for SFNs. This means that ADD is closer to synchronous updating than ASD. However, we are talking about s values very close to 0. Furthermore, the r values are very close to 1, which means that the two methods are much closer to each other than to synchronous updating. Given that these values result from all the combinations simulated for each type of interaction topology, we later separated the results along the different values of each parameter in order to enquire if there are certain types of combinations for which there is a clear difference between the two methods. We made separations for all the parameters and, as Table 2 shows, the same conclusions derived above can be applied no matter how we separate the results. This means that, although ADD is almost always closer to synchronous updating than ASD, it is much closer to ASD than to synchronous updating. This result indicates that the conclusions derived for ASD also apply to ADD and that possible different levels of cooperation observed in the system being modeled are not due to the deterministic or stochastic nature of the updating scheme used by the agents. Also, this means that if we are willing to explain the influence of update methods on the level of cooperation we must look somewhere else away from their deterministic *versus* stochastic nature.

Table 2. s and r values for the different values of each parameter

SWNs						SFNs					
$\phi = 0.0$		$\phi = 0.1$		$\phi = 1.0$		$m_0 = 4$		$m_0 = 8$		$m_0 = 12$	
s	r	s	r	s	r	s	r	s	r	s	r
0.020	0.877	0.006	0.952	0.011	0.940	0.013	0.916	0.008	0.928	0.011	0.900

$m = +\infty$		$m = 100$		$m = 2$		$m = 1$	
s	r	s	r	s	r	s	r
0.009	0.876	0.007	0.919	0.012	0.939	0.017	0.932

$\alpha = 0.5 \equiv p = 2$		$\alpha = 0.2 \equiv p = 5$		$\alpha = 0.1 \equiv p = 10$	
s	r	s	r	s	r
0.025	0.792	0.011	0.933	−0.002	0.985

4 Conclusion and Future Work

In this work we examined the influence of asynchronism on the evolution of cooperation in models where agents try to adapt their behavior to the context in which they live. We showed that, in general, asynchronism supports more cooperators in the Spatial Prisoner's Dilemma game than synchronism. This conclusion contradicts previous ones but it results from a more general analysis, based on a bigger number of tested conditions, namely, different types of topologies, various values for the determinism degree of the transition rule and different levels of the synchrony rate. Besides this conclusion, the asynchronous update method used allowed us to derive some conclusions concerning, for example, the sensitivity and monotonicity of the model to changes in the synchrony rate. Finally, we found that the outcome of the studied model is approximately the same whether a stochastic or a deterministic asynchronous updating is used.

Future extensions to this work will explore the ASD update method with other games in order to verify if the results achieved with the PD game as, for example, the *small determinism degree and small synchrony rate* are also present. The results achieved in [12] with the Snowdrift game, where the best-neighbor ($+\infty$) and the proportional ($m = 1$) transition rules, as well as synchronous and sequential updating were used, seem to indicate that this is the case. However, only by exploring intermediate levels of asynchronism and intermediate levels of determinism of the transition rule we can confirm this. We will also try to answer the question "Is there an explanation for the influence of asynchronism on the emergence of cooperation?" and, if it turns out that there is a positive answer "Is this explanation extensible to other dynamical systems?".

Acknowledgments

We thank the GruVA members Pedro Santana, Carlos Cândido, Vasco Santos and Pedro Mariano for useful discussions and comments. This work was partially supported by FCT/MCTES grant No. SFRH/BD/37650/2007.

References

1. Barabasi, A.-L., Albert, R.: Emergence of scaling in random networks. Science 286, 509 (1999)
2. Fatès, N., Morvan, M.: An experimental study of robustness to asynchronism for elementary cellular automata. Complex Systems 16(1), 1–27 (2005)
3. Gershenson, C.: Classification of random boolean networks. In: Proceedings of the Eight International Conference on Artificial Life, pp. 1–8. MIT Press, Cambridge (2002)
4. Grilo, C., Correia, L.: Asynchronous stochastic dynamics and the spatial prisoner's dilemma game. In: Neves, J., Santos, M.F., Machado, J.M. (eds.) EPIA 2007. LNCS (LNAI), vol. 4874, pp. 235–246. Springer, Heidelberg (2007)
5. Huberman, B., Glance, N.: Evolutionary games and computer simulations. Proceedings of the National Academy of Sciences 90, 7716–7718 (1993)
6. Newth, D., Cornforth, D.: Asynchronous spatial evolutionary games: spatial patterns, diversity and chaos. In: Proceeding of the 2007 IEEE Congress on Evolutionary Computation, pp. 2463–2470 (2007)
7. Nowak, M., Bonhoeffer, S., May, R.M.: More spatial games. International Journal of Bifurcation and Chaos 4(1), 33–56 (1994)
8. Nowak, M., May, R.M.: Evolutionary games and spatial chaos. Nature 359, 826–829 (1992)
9. Oh, J.C.: Cooperating search agents explore more than defecting search agents in the internet information access. In: Proceedings of the 2001 Congress on Evolutionary Computation, CEC2001, pp. 1261–1268. IEEE Press, Los Alamitos (2001)
10. Pacheco, J.M., Santos, F.C.: Network dependence of the dilemmas of cooperation. In: Science of Complex Networks: From Biology to the Internet and WWW, CNET 2004, vol. 776, pp. 90–100 (2005)
11. Smith, J.M.: Evolution and the Theory of Games. Cambridge University Press, Cambridge (1982)
12. Tomassini, M., Luthi, L., Giacobini, M.: Hawks and doves on small-world networks. Physical Review E 73(1), 016132 (2006)
13. Watts, D., Strogatz, S.H.: Collective dynamics of small-world networks. Nature 393, 440–442 (1998)