IEEE 2005 Symposium on Computational Intelligence and Games

CIG'05

April 4-6 2005

Essex University, Colchester, Essex, UK

Graham Kendall and Simon Lucas (editors)

Contents

Poster Presentations

Preface

Welcome to the inaugural 2005 IEEE Symposium on Computational Intelligence and Games. This symposium marks a milestone in the development of machine learning, particularly using methods such as neural, fuzzy, and evolutionary computing. Let me start by thanking Dr. Simon Lucas and Dr. Graham Kendall for inviting me to write this preface. It's an honor to have this opportunity.

Games are a very general way of describing the interaction between agents acting in an environment. Although we usually think of games in terms of competition, games do not have to be competitive: Players can be cooperative, neutral, or even unaware that they are playing the same game. The broad framework of games encompasses many familiar favorites, such as chess, checkers (draughts), tic-tac-toe (naughts and crosses), go, reversi, backgammon, awari, poker, blackjack, arcade and video games, and so forth. It also encompasses economic, social, and evolutionary games, such as hawk-dove, the prisoner's dilemma, and the minority game. Any time one or more players must allocate resources to achieve a goal in light of an environment, those players are playing a game.

Artificial intelligence (AI) researchers have used games as test beds for their approaches for decades. Many of the seminal contributions to artificial intelligence stem from the early work of Alan Turing, Claude Shannon, Arthur Samuel, and others, who tackled the challenge of programming computers to play familiar games such as chess and checkers. The fundamental concepts of minimax, reinforcement learning, tree search, evaluation functions, each have roots in these early works.

In the 1940s and 1950s, when computer science and engineering was in its infancy, the prospects of successfully programming a computer to defeat a human master at any significant game of skill were dim, even if hopes were high. More recently, seeing a computer defeat even a human grand master at chess or checkers, or many other familiar games, is not quite commonplace, but not as awe-inspiring as it was only a decade ago. Computers are now so fast and programming environments are so easy to work with that brute force methods of traditional AI are sufficient to compete with or even defeat the best human players in the world at all but a few of our common board games.

Although it might be controversial, I believe that the success of Deep Blue (the chess program that defeated Garry Kasparov at chess), Chinook (the checkers program that earned the title of world champion in the mid-1990s), and other similar programs mark the end of a long journey - but not the journey started by Turing, Shannon, and Samuel - but rather a different journey.

The laudatory success of these traditional AI programs has once again pointed to the limitations of these programs. Everything they "know" is preprogrammed. They do not adapt to new players, they assume their opponent examines a position in a similar way as they do, they assume the other player will always seek to maximize damage, and most importantly, they do not teach themselves how to improve beyond some rudimentary learning that might be exhibited in completing a bigger lookup table of best moves or winning conditions. This is not what Samuel and others had in mind

when asking how we might make a computer do something without telling it how to do it, that is, to learn to do it for itself. Deep Blue, Chinook, and other superlative programs have closed the door on one era of AI. As one door closes, another opens.

Computational intelligence methods offer the possibility to open this new door. We have already seen examples of how neural, fuzzy, and evolutionary computing methods can allow a computer to learn how to play a game at a very high level of competency while relying initially on little more than primitive knowledge about the game. Some of those examples include my own efforts with Blondie24 and Blondie25 in checkers and chess, respectively, and perhaps that is in part why I was asked to contribute this preface, but there are many other examples to reflect on, and now many more examples that the reader can find in these proceedings. No doubt there will be many more in future proceedings.

Computational intelligence methods offer diverse advantages. One is the ability for a computer to teach itself how to play complex games using self-play. Another is the relatively easy manner in which these methods may be hybridized with human knowledge or other traditional AI methods, to leapfrog over what any one approach can do alone. Yet another is the ability to examine the emergent properties of evolutionary systems under diverse rules of engagement. It is possible to examine the conditions that are necessary to foster cooperation in different otherwise competitive situations, to foster maximum utilization of resources when they are limited, and when players might simply opt out of playing a game altogether. Computational intelligence offers a versatile suite of tools that will take us further on the journey to making machines intelligent.

If you can imagine the excitement that filled the minds of the people exploring AI and games in the 1940s and 1950s, I truly believe what we are doing now is even more exciting. We all play games, every day. We decide how to allocate our time or other assets to achieve our objectives. Life itself is a game, and the contributors to this symposium are players, just as are you, the reader. Not all games are fun to play, but this one is, and if you aren't already playing, I wholeheartedly encourage you to get in the game. I hope you'll find it as rewarding as I have, and I hope to see you at the next CIG symposium. There is a long journey ahead and it is ours to create.

David B. Fogel
Chief Executive Officer
Natural Selection, Inc.
La Jolla, CA, USA

Acknowledgements

This symposium could not have taken place without the help of a great many people and organisations.

We would like to thank the IEEE Computational Intelligence Society for supporting this symposium. They have provided both their knowledge and experience as well as providing the budgetary mechanisms to allow us to establish this inaugural event.

Smita Desai was one of our main contacts at IEEE and she guided us through the financial decisions that we had to make. As this was our first IEEE symposium as co-chairs, we found her help, advice and experience invaluable.

The UK Engineering and Physical Sciences Research Council (EPSRC) provided financial support. Without their support we would only have had one plenary speaker and their assistance has added significantly to the profile and quality of the symposium.

The on-line review system was developed by Tomasz Cholewo. This system made the organisation of the conference a lot easier than it might otherwise had been.

We would like to thank the four plenary speakers (Jordan B. Pollack, Risto Miikkulainen, Martin Mueller, Jaap van den Herik). Their willingness to present at this inaugural event has greatly helped raised the profile of the symposium.

We are deeply grateful to the three tutorial speakers (Andries P. Engelbrecht, Evan J. Hughes, Thomas P. Runarsson) who all gave up part of their weekend to allow others to learn from their knowledge and experience.

The program committee (see next page) did an excellent job in reviewing all the submitted papers. The legacy of a symposium is largely judged by the people represented on the program committee and we realise that we have been fortunate to have internationally recognised figures in computational intelligence and games represented at this symposium.

Lastly, but by no means least, we would like to acknowledge the contribution of David Fogel. We would to thank him for having the confidence in us to realise his idea for this symposium. David has also been a constant source of advice and support which we have found invaluable.

Graham Kendall and Simon Lucas
The University of Nottingham and the University of Essex

Program Committee

- Dan Ashlock, University of Guelph, Canada
- Ian Badcoe, ProFactum Software, UK
- Luigi Barone, The University of Western Australia, Australia
- Alan Blair, University of New South Wales, Australia
- Bruno Bouzy, Universite Rene Descartes, France
- Michael Buro, University of Alberta, Canada
- Murray Campbell, IBM T.J. Watson Research Center, USA
- Darryl Charles, University of Ulster, UK
- Ke Chen, University of Manchester, UK
- Sung-Bae Cho, Yonsei University, Korea
- Paul Darwen, Protagonist Pty Ltd, Australia
- Abdennour El Rhalibi, Liverpool John Moores University, UK
- Andries Engelbrecht, University of Pretoria, South Africa
- Thomas English, The Tom English Project, USA
- Maria Fasli, University of Essex, UK
- David Fogel, Natural Selection, Inc., USA
- Tim Hays, Natural Selection, Inc., USA
- Jaap van den Herik, Universiteit Maastricht, The Netherlands
- Phil Hingston, Edith Cowan University, Australia
- Howard A. Landman, Nanon, USA
- Huosheng Hu, University of Essex, UK
- Evan J. Hughes, Cranfield University, UK
- Doran Jim, University of Essex, UK
- Graham Kendall, University of Nottingham, UK (co-chair)
- Thiemo Krink, University of Aarhus, Denmark
- Sushil J. Louis , University of Nevada, Reno
- Simon Lucas, University of Essex, UK (co-chair)
- Stephen McGlinchey, University of Paisley, UK
- Risto Miikkulainen, University of Texas at Austin, USA
- Martin Muller, University of Alberta, Canada
- Jordan Pollack, Brandeis University, USA
- Thomas Philip Runarsson, University of Iceland, Iceland
- Jonathan Schaeffer, University of Alberta, Canada
- Lee Spector, Hampshire College, USA
- Rene Thomsen, University of Aarhus, Denmark
- Nikos Vlassis, Univ. of Amsterdam, The Netherlands
- Lyndon While, University of Western Australia, Australia
- Xin Yao, University of Birmingham, UK

A New Framework to Analyze Evolutionary 2×2 Symmetric Games

Umberto Cerruti
Mathematics Department
University of Torino
Torino, Italy
umberto.cerruti@unito.it

Mario Giacobini
Information Systems Department
University of Lausanne
Laussane, Switzerland
mario.giacobini@unil.ch

Ugo Merlone
Statistics and Applied Mathematics Department
University of Torino
Torino, Italy
merlone@econ.unito.it

***Abstract*— In this paper we present a new framework to analyze the behavior of evolutionary 2×2 symmetric games. The proposed approach enanbles us to predict the dynamics of the system using the parameters of the game matrix above, without dealing with the concepts of Nash equilibria and evolutionary stable strategies. The predictions are in complete accordance with those that can be made with these latter concepts. Simulations have been performed on populations with spatial structures, and show a good agreement with the model's predictions. We also analyze the dynamics of a particular system, showing how effectively the framework applies to it.**

I. Introduction

Harrald [1] used genetic algorithms as evolutionary dynamics and gives a representation of players with limited memory in repeated games. His approach is based on binary representation of mixed strategy players and is extended in order to use deterministic finite automata for these games. Starting from this approach we introduce a different and, in some sense, more natural representation for players and are able to give an elegant analysis of the game evolution. Finally, we implement this framework using both Matlab and C++ and compare the results. The structure of the paper is the following: in Section II we recall some fundamental notions of evolutionary game theory and present Harrald's contribution; in Section III we discuss some of Harrald's assumptions and present our approach. In Section IV we introduce a spatially structured evolutionary algorithm and give a formal description of the evolutionary system. Section V is devoted to the simulation analysis and conclusions are given in Section VI. The Appendix concerns some consequences of floating point arithmetic error we encountered in our implementations.

II. Probabilistic Players

Let's consider the general form of a 2×2 symmetric game where the two players always choose from the same action set, say $\{X, Y\}$, with the payoff matrix as depicted in Table I.

	X	Y
X	e	g
Y	h	f

TABLE I
The general payoff matrix for a symmetric 2×2 game.

In [1], Paul Harrald proposed an approach based on probabilistic strategies. A player's strategy is no longer deterministic, and becomes a probability of playing action X, regardless of the past actions played by both players. With the exception of the cases where the strategy probability is either 0 (i.e., always play action Y) or 1 (i.e., always play action X), the resulting strategies are mixed. A strategy is represented as a binary chromosome of fixed length L: the binary string is decoded into an integer value that is then divided by $2^L - 1$, so to obtain the actual value of the strategy. In a panmictic (i.e., not spatially distributed) population, each agent in the population plays against each other agent in a repeated game for a fixed number of iterations, obtaining a total payoff representing his fitness. During a game, each player determines his moves randomly choosing between the two actions X and Y with the probability encoded in his chromosome. By tournament selection, couples of parents are selected according to their fitness values. Two offspring are obtained from each couple of parents using one-point crossover, and each bit of their chromosome is mutated by standard binary mutation. The obtained offspring population is then considered as the new population of the next generation.

We consider symmetric bimatrix games $G(I, S, \pi)$, where $I = \{1, 2\}$ is the player set, consisting of two players, S is the pure strategies space and π is the combined payoff function fully represented by the associated payoff matrix pair (R, C), where $C = R^{\mathrm{T}}$ (see [2] for details). As usual, the set of mixed strategies for player i is denoted Δ_i and, since we restrict our attention to symmetric games, it holds $\Delta := \Delta_1 = \Delta_2$

In this class of games we define the *symmetric Nash equilibrium* as any strategy pair $(x, y) \in \Delta^2$ such that $x \in \beta(y)$ and $y \in \beta(x)$ where $\beta(\cdot)$ is the best reply correspondence, which maps each mixed strategy to the face of Δ which is spanned by the pure best reply to '$\cdot$'. Finally an *evolutionary stable strategy* (ESS) is a strategy $x \in \Delta$ such that for every strategy $y \neq x$ there exists some $\bar{\epsilon}_y \in (0, 1)$ such that for all $\epsilon \in (0, \bar{\epsilon}_y)$ it holds:

$$x \cdot R(\epsilon y + (1 - \epsilon) x) > y \cdot R(\epsilon y + (1 - \epsilon) x)$$

III. Probabilistic Payoffs

Two main criticisms can be raised to Harrald's evolutionary machinery:

- Since a player has no memory of the previous moves in a game, there is no need to make each couple of opponents play all the iterations of a game. In fact, given a big enough number of play iterations, if we denote the probabilities of the two players A and B with p_A and p_B respectively, then we can approximate the expected gain of player A according to the game described by the matrix in Table I by the expression:

$$ep_Ap_B+gp_A(1-p_B)+h(1-p_A)p_B+f(1-p_A)(1-p_B). \quad (1)$$

- The binary representation of the probability could not be the most suitable one (for a complete discussion on the representation choice see, for example, [3]). Other possible representation could be better suited, such as the real number one, as suggested by the author himself in the article.

Concerning the first criticism, let's consider a population $P(t)$ of N probabilistic players at generation t and denote p_i, the agent i's probability of playing action X. The expected gain of player i, when playing with agent j, follows from expression (1):

$$\begin{aligned} G(i,j,t) &= ep_i(t)p_j(t) + gp_i(t)(1-p_j(t)) + \\ &\quad + h(1-p_i(t))p_j(t) + f(1-p_i(t))(1-p_j(t)), \end{aligned}$$

therefore, the fitness (i.e., the sum of his payoffs against all other agents in the population) $f(i,t)$ of agent i at generation t is

$$f(i,t) = \sum_{j\neq i} G(i,j,t). \quad (2)$$

If we denote the sum of the probabilities of all players in the population at generation t with $U(t)$, equation (2) becomes

$$\begin{aligned} f(i,t) &= ep_i(t)(U(t)-p_i(t)) + \\ &\quad + gp_i(t)(N-1-U(t)+p_j(t)) + \\ &\quad + h(1-p_i(t))(U(t)-p_i(t)) + \\ &\quad + f(1-p_i(t))(N-1-U(t)+p_j(t)). \quad (3) \end{aligned}$$

Equation (3) gives an effective way of calculating the fitness of an agent of a given population without having to perform the actual games between the agent and all the other agents in the population.

It is well known (see for example [2]) that every 2×2 symmetric game can be normalized, and is equivalent to a doubly symmetric game, where the payoff matrix is symmetric. While this equivalence is proven in game theoretical context, it remains to be analyzed when considering dynamical evolutions. The new game, that is called reduced, has the payoff matrix displayed in Table II, where $a=e-h$ and $b=f-g$.

We decided to focus our attention on doubly symmetric games, given their relevance in evolutionary game theory. In

	X	Y
X	a	0
Y	0	b

TABLE II

The general payoff matrix for a reduced symmetric 2×2 game.

the case of a reduced symmetric game, the expected gain of player i playing against agent j at generation t is

$$G(i,j,t) = ap_i(t)p_j(t) + b(1-p_i(t))(1-p_j(t)),$$

thus, following the same reasoning done for equation (3), the fitness $f(i,t)$ of agent i at generation t is given by

$$\begin{aligned} f(i,t) &= ap_i(t)(U(t)-p_i(t) + \\ &\quad + b(1-p_i(t))(N-1-U(t)+p_i(t)). \quad (4) \end{aligned}$$

At generation t, let's define the mean agent $\bar{p}(t)$ of population $P(t)$ as the mean of the probabilities of the N agents of the population, i.e., $\bar{p}(t)=U(t)/N$. If we replace in equation (4) the value of $p_i(t)$ with the value of the mean agent $\bar{p}(t)$, and we divide by the constant factor $N-1$, we obtain the expression

$$F(\bar{p},t) = (a+b)\bar{p}^2(t) - 2b\bar{p}(t) + b, \quad (5)$$

proportional to the fitness of the mean agent at generation t. This equation determines a parabola with vertex V at abscissa $b/(a+b)$. Note that this value coincides with the value of the game.

The selection pressure of an evolutionary algorithm evolving this kind of agents' strategy will drive the mean agent of the population towards higher values on the parabola described by equation (5).

According to the possible values of the matrix parameters a and b in a reduced symmetric game we have the following four cases:

1) both a and b are positive: the parabola is concave and the evolution will depend from the mean agent $\bar{p}(0)$ of the initial population $P(0)$; if $\bar{p}(0) < b/(a+b)$ (the vertex of the parabola, i.e., the value of the game), then the evolution will be driven toward action Y; otherwise the evolution will be driven toward action X (see Figure 1(a));
2) a is negative and b is positive: the parabola is decreasing in the interval $[0,1]$. Therefore, whatever the initial population is, the evolution will be driven toward action Y (see Figure 1(b));
3) both a and b are negative: the parabola is convex, since its vertex is inside the interval $[0,1]$. Thus, whatever the initial population is, the evolution will be driven toward the vertex, i.e. the value of the game (see Figure 1(c));
4) a is positive and b is negative: the parabola is increasing in the interval $[0,1]$. Therefore whatever the initial population is, the evolution will be driven toward action X (see Figure 1(d)).

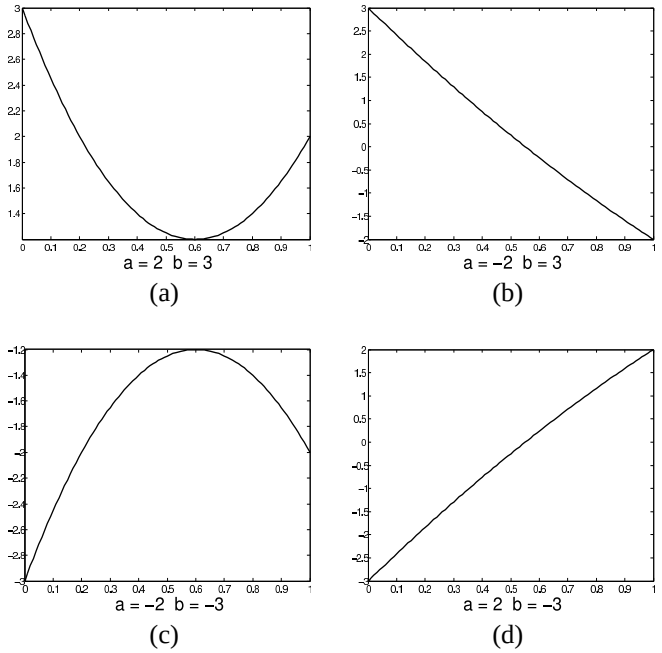


Fig. 1. The parabola defined by equation (5) when $a > 0$ and $b > 0$ (a), $a < 0$ and $b > 0$ (b), $a < 0$ and $b < 0$ (c), and $a > 0$ and $b < 0$ (d).

The results of this analysis completely agree with classical results of the evolutionary theory of 2×2 symmetric games (see [2] and [4]) obtained using the concepts of Nash equilibria and evolutionary stable strategies. Moreover, the model implies that the spatial structure of the population does not influence the mean behaviors of the evolved populations, as we will see in the next section.

IV. Spatial Artificial Evolution

The framework described in the previous section is independent of the spatial structure of the evolved population. To test whether its predictions are good when spatial constraints are introduced, we have decided to evolve populations on two-dimensional regular lattices: each agent is placed on a vertex of a rectangular grid with periodic boundary conditions (i.e., a toroidal structure), and is connected with the eight closest agents, thus defining a Moore neighborhood. While other neighborhoods are possible, the results remain qualitatively the same. Furthermore, one of the authors is running an experiment on human subjets; among the results, it is evident that individuals choose a partner in their physical Moore neighborhood. In further research we will consider the dynamical evolution on different networks such as small-world networks, fragmented networks and random networks.

The evolution is performed synchronously: at each generation, each agent selects the fittest agent in his neighborhood and produces an offspring whose associated probability is obtained by intermediate crossover (also known as arithmetical or guaranteed average crossover [3]) between the two probabilities associated with the agent itself and the selected agent. No mutation is used in this process, and the produced offspring replaces the considered agent in his location in the structure. The dynamical system is completely deterministic: different attractors can be found for each system, but, as we will show in the simulations in Section V, the mean agent will always tend to 0, 1 or the value of the game (the vertex of the parabola) depending on the cases described in the previous section.

Even if the algorithm is quite simple, we have noticed that using two different implementations in Matlab and C++ we obtained results qualitatively comparable but numerically different. This is probably due to the different internal representations of real numbers: for more details see the implementation note in the Appendix. While different approaches have been suggested for handling errors in floating point representations (e.g., interval arithmetic, see [5]), given the finite state structure of our model, we decided to use integer representation for state s. This approach is similar, in a certain sense, to that used by Harrald [1].

To each agent a_i is thus associated an integer state $s_i \in \{0, 1, \ldots, M\}$: the agent will then play action X with probability $p_i = s_i/M$. To calculate the gain (the fitness) of agent a_i, this probability is used in equation (4): if we denote with $W(t)$ the sum of the states of the agents of the population at time t ($W(t) = \sum_{i=1}^{N} s_i$), we have $U(t) = W(t)/N$. Multiplying by M^2 and simplifying, we obtain the following form of equation (4):

$$\begin{aligned} F(i,t) &= -(a+b)s_i^2(t) + \\ &\quad + ((a+b)W(t) + Mb(2-N))s_i(t) + \\ &\quad + Mb(M(N-1) - W(t)). \end{aligned} \tag{6}$$

This evolutionary system can be described in a more formal way: let's consider a discrete time t and a population $P(t)$ of N agents. To each agent are associated a state and a location: $P(t) = \{a_1(t), a_2(t), \ldots, a_N(t)\}$, with $a_i = \langle s_i(t), l_i \rangle$, where $s_i(t) \in S = \{s_1, s_2, \ldots, s_m\}$, the set of the possible states of the agents, and $l_i \in L = \{l_1, l_2, \ldots, l_N\}$, the set of the locations of the agents in the structure of the population. If we denote with T the product space between the space of the possible states and the space of the possible locations of the agents ($T = S \times L$), a population of N agents is an element of T^N.

A fitness function $F : T^N \rightarrow R^N$ (where R is the set of real numbers) is given, such that each population $P(t) = \{a_1(t), a_2(t), \ldots, a_N(t)\}$ is associated to a vector $f(t) = F(P(t)) = \langle f_1(t), f_2(t), \ldots, f_N(t) \rangle$ with $f_i(t)$ being the fitness value of agent $a_i(t)$.

The selection mechanism is described by a function $Sel : T^N \times R^N \rightarrow S^N$ such that $\langle s'_1(t), s'_2(t), \ldots, s'_N(t) \rangle = Sel(P(t), F(P(t)))$. For each agent in the population it selects the state of the agent in his neighborhood with the highest fitness value. Note that only the state of an agent is selected, since the location of the selected agent doesn't influence the successive crossover. On the contrary the location of the selecting agent influences the function Sel, since it determines the selection pool for each location in the structure. The topology of the structure thus affects the selection function, but not the successive reproduction operators.

The state of the agent in the considered location is then combined with the selected state by a function $Op : S \times S \rightarrow S$, producing the state of the agent in the next generation for the considered location. If only a crossover operator is used, as it is the case in our evolutionary algorithm, the function Op can be represented in the form of an $N \times N$ matrix of elements of S.

Given the topology of the structure, the set L of the possible locations of the agents, the set S of the possible states of the agents, the fitness function F, the selection function Sel, the recombination function Op, and the population $P(t)$, the population $P(t+1) = \{a_1(t+1), a_2(t+1), \ldots, a_N(t+1)\}$ at the next generation is formed by agents $a_i(t+1) = \langle s_i(t+1), l_i \rangle$ such that $s_i(t+1) = Op(s_i(t), s'_i(t))$.

V. Simulations Analysis

Two groups of simulation have been performed to test the exactness of the models' predictions: the first time, we let the system evolve starting from random populations. Then we created a particular initial population and the system dynamical behavior is observed, so as to show how the prediction of the model actually works.

At first, we let evolve a population of 2500 agents at 21 possible states disposed on a 50×50 toroidal grid with a Moore neighborhood (each agent's neighborhood is composed by the agent itself and the 8 agents directly surrounding him). The agents face a game whose matrix is the one depicted in Table III).

	X	Y
X	-2	0
Y	0	-3

TABLE III

THE PAYOFF MATRIX FOR THE SIMULATIONS.

Such a matrix falls under case 3) of Section II since both a and b are negative. The model in this case predicts that, whatever the initial population is, the mean agent will tend to the value of the game, which in this case is 0.6. This prediction is confirmed by the simulation results: in Figure 2 the evolution of the mean player over a generation is shown, when starting with a random population composed by 20% of agents playing action X with probability 1, and 80% playing action Y with probability 1.

The evolution over the generations of the fitness of the mean agent is shown in figure 3(a): it can be noticed how, even though the mean agent value oscillates between two different states, its fitness value (its payoff against all other members of the population) stabilizes. The fitness of the mean agent is clearly linked to the mean fitness of the population: the artificial evolution tends to populations of different agents who have very similar fitness values. In fact the difference between the maximal and the minimal fitnesses of the population through generations tends to 0, as it is shown in figure 3(b).

As we have previously pointed out, the model can predict the behavior of the mean agent, without taking into account

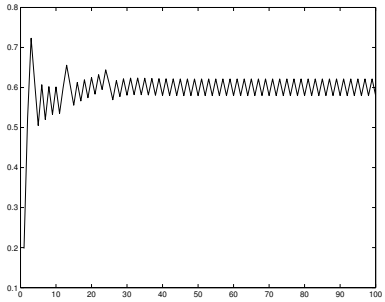
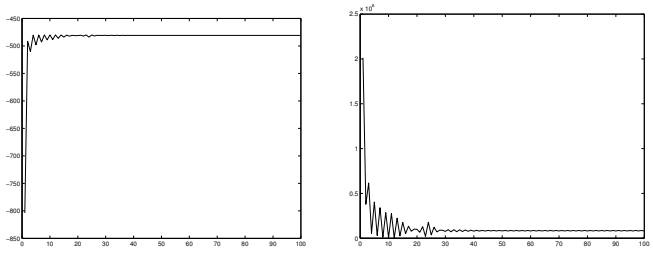


Fig. 2. Evolution of the value of the mean player over time of a population of 2500 agents at 21 possible states disposed on a 50×50 toroidal grid with a Moore neighborhood.



(a) (b)

Fig. 3. Evolution over the generations of the fitness of the mean agent (a) and of the difference between the maximal and the minimal fitness of a population of 2500 agents at 21 possible states disposed on a 50×50 toroidal grid with a Moore neighborhood.

the structure of the population and the initial disposition of the agents. In fact, starting with different populations, we will observe different attractors for the evolutionary process. For this simulation a period 2 attractor can be observed (see figure 4, where the two populations are shown): darker agents correspond to probabilities closer to 0 of playing action X, with black agent corresponding to probability 0, and white agents to probability 1.

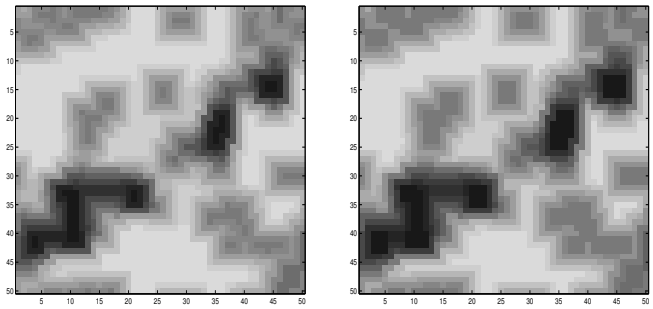


Fig. 4. Final populations of 2500 agents at 21 possible states disposed on a 50×50 toroidal grid with a Moore neighborhood. The two populations form a period 2 cyclic attractor of the evolutionary system.

If the payoff matrix is changed to the one depicted in Table IV, the game falls under case 1) of Section II since both a and b are positive.

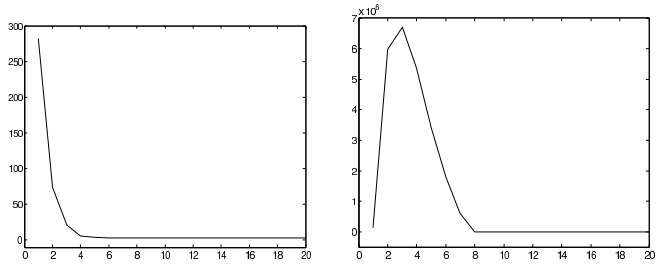
The model predicts that the evolution will depend on the mean agent $\bar{p}(0)$ of the initial population $P(0)$; if $\bar{p}(0) < 0.6$ (the value of the game), then the evolution will be driven to-

	X	Y
X	2	0
Y	0	3

TABLE IV

THE PAYOFF MATRIX FOR THE SIMULATIONS.

wards action Y; otherwise the evolution will be driven towards action X. The prediction is fully confirmed by the simulations shown in Figures 5 and 6, where the time evolution of the mean agent and of the difference between the maximal and the minimal population fitnesses are shown, in the case of initial random populations with $\bar{p}(0) = 0.5939$ and $\bar{p}(0) = 0.6047$ respectively. Note how the difference between the maximal and the minimal fitnesses in the population rapidly grows at the beginning of the evolution (the agents split towards opposite strategies), and then tends to 0.



(a) (b)

Fig. 5. Evolution over the generations of the mean agent (a) and of the difference between the maximal and the minimal fitnesses of a population of 2500 agents at 21 possible states disposed on a 50×50 toroidal grid with a Moore neighborhood. The initial population has a mean agent with associated probability $\bar{p}(0) = 0.5939 < 0.6$, the value of the game.

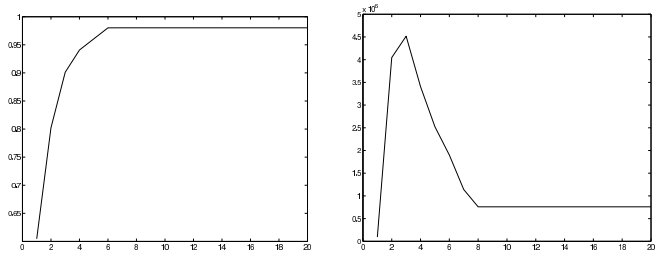


Fig. 6. Evolution over the generations of the mean agent (a) and of the difference between the maximal and the minimal fitnesses of a population of 2500 agents at 21 possible states disposed on a 50×50 toroidal grid with a Moore neighborhood. The initial population has a mean agent with associated probability $\bar{p}(0) = 0.6047 > 0.6$, the value of the game.

In the second part of the simulations we consider a small population of 121 agents disposed on a 11×11 toroidal grid. Each agent's neighborhood is composed by the agent itself and the 8 agents directly surrounding him (thus forming a Moore neighborhood). The payoff matrix of the game is the same as for the first group of simulations (see Table III).

The set of an agent's possible states is composed of 7 elements ($S = \{0, 1, \ldots, 6\}$). The probabilities of playing action X associated to the 7 states are respectively: 0, 0.1667, 0.3333, 0.5, 0.6667, 0.8333, and 1. To draw the populations during the evolution, we have associated to each state a color on a grey scale (see Figure 7).

Fig. 7. Color scale for 7 state agents: from state 0 (black) we pass through states corresponding to probabilities 0.1667, 0.3333, 0.5, 0.6667, 0.8333, to finally reach state 6 (white) that corresponds to probability 1 of playing action X.

The intermediate crossover between the integer states is performed according to the crossover matrix of Table V: recombining a state i agent with a state j agent, the state of the offspring agent will be the one at the intersection of row i and column j of the matrix.

	0	1	2	3	4	5	6
0	0	0	1	1	2	2	3
1	0	1	1	2	2	3	3
2	1	1	2	2	3	3	4
3	1	2	2	3	3	4	4
4	2	2	3	3	4	4	5
5	2	3	3	4	4	5	5
6	3	3	4	4	5	5	6

TABLE V

THE CROSSOVER MATRIX FOR AGENTS WITH 7 POSSIBLE STATES.

Figure 8 shows the evolution of the system starting from an initial population solely of all agents playing action X with probability 1 (agents' states 6), with the exception of the central individual who plays action Y with probability 1 (agent state 0). For each generation ($t = 0, 1, \ldots, 7$) the population is plotted on the left, and the parabola associated to the population is drawn on the right: the probabilities associated with the 7 possible states of the agents are on the x axis, and the corresponding fitness values, function of the sum $U(t)$ of the probabilities associated to the agents in the population, are on the y axis.

At time $t = 0$ the single agent at state 0 has the highest fitness value (the parabola is decreasing in the interval $[0, 1]$), and therefore it will be selected by its surrounding neighbors for recombination. Applying the crossover matrix (see Table V), at the next generation ($t = 1$) the population will be formed by one agent at state 0 surrounded by 8 agents at state 3, and all other agents at state 6. The behavior of the population from time 1 to time 4 is analogous to that of time 0: since smaller associated probabilities have higher fitness values, the agents will recombine with the agent in their neighborhood with smaller states. Note that the agents at state 6 disappear, because they have always the smallest fitness value. Since the crossover matrix allows the production of state 6 agents only when both the parents have state 6, that state will never appear once lost in the population.

At time $t = 5$ the parabola becomes increasing in the interval $[0, 1]$. The agents at state 0 have the lowest fitness

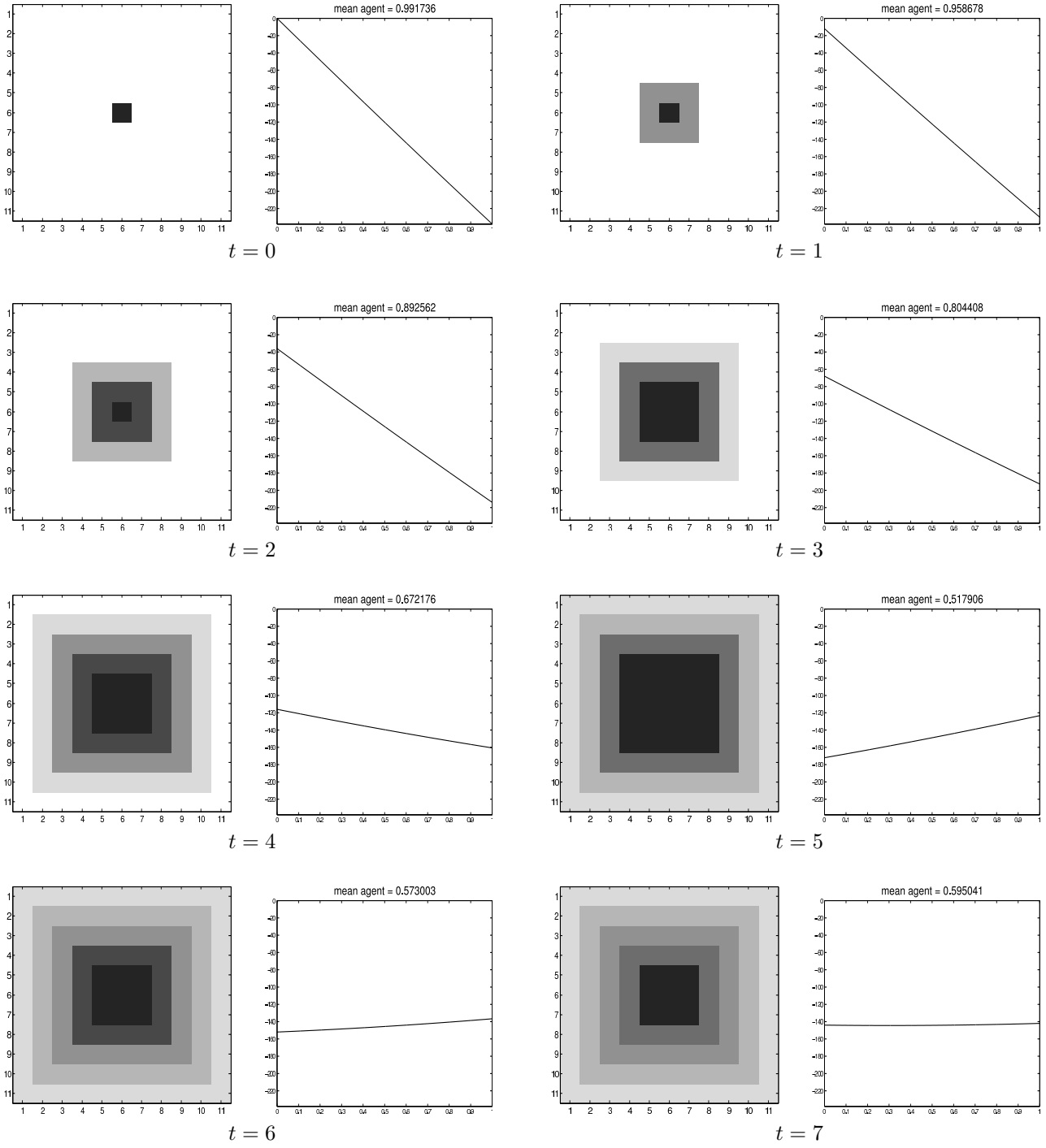


Fig. 8. Evolution of a population of 11×11 agents that can assume 7 possible states: the initial population is composed by all agents playing action X with probability 1 (agents' states 6), except the central individual that plays action Y with probability 1 (agent state 0). For each time step, the population (left) and the corresponding parabola (right) are shown. At time $t = 8$ the population will be the same as at time $t = 6$, resulting in an attractor of period 2 for the dynamic of the system.

value: those at the border of the region will select state-2 agents for recombination (since they have higher fitness), producing state-1 offspring agents. Agents at state 2 will select, for the same reason, agents at state 4, producing state-3 offspring agents. All other agents don't change state, since the crossover operator will produce offsprings with the same state (see Table V).

At time $t = 6$ (see the enlargement in Figure 9 left) the parabola is still increasing in the interval $[0, 1]$: only agents at state 1 will change state, since the only crossover producing

offsprings with a different state is the one between agents at state 1 and agents at state 3. At time $t = 7$ (see the enlargement in figure 9 right), parabola is such that state-0 agents have a higher fitness than state-1, -2, and -3 agents, and therefore only agents at state 2 selecting agents at state 0 will produce an offspring at a different state (1). A population equal to that of generation 6 is produced, and the system enters in a period-2 attractor oscillating between the two configurations. The system oscillates between states in which the mean agent has strategies 0.573003 and 0.595041. This result completely agrees with the model's prediction: since both a and b are negative, the mean agent shall tend to the value of the game, which in this case is 0.6.

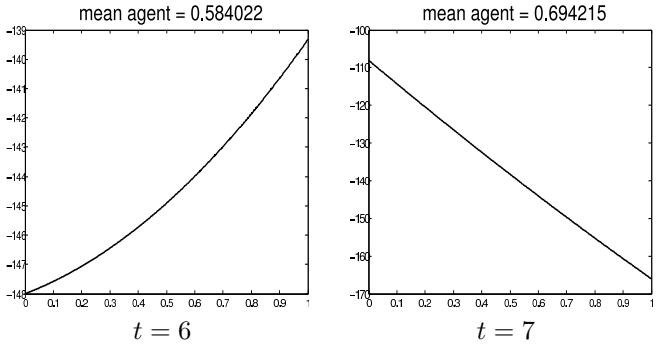


Fig. 9. Enlargement, enlarging y-axis, of the parabola of figure 8 at generations 6 and 7: from increasing in the probability interval $[0, 1]$ at generation 6, it becomes decreasing at generation 7.

VI. Conclusions and Future Work

We have introduced a new framework to analyze and predict the behavior of evolutionary 2×2 symmetric games. This approach only uses the parameters of the payoff matrix of the game, and leads to behavior predictions that are in perfect agreement with classical evolutionary theory, without dealing with Nash equilibria or evolutionary stables strategies. The proposed model is not influenced by the spatial structure of the evolving population of agents.

The evolutionary algorithm used to experimentally validate the model is then described, introducing a new formalism for the evolution of spatially structured populations. The experiments fully confirm the predicted behaviors, and a complete analysis of a simple dynamical system is presented, in order to exemplify the model.

In the future we intend to investigate the different dynamics induced by different crossover matrices, and the possibility of the co-evolution of strategies and crossover operators. We also want to study and model the introduction of random mutation in the evolutionary algorithm.

Acknowledgments

Mario Giacobini gratefully acknowledge financial support by the Fonds National Suisse pour la Recherche Scientifique under contract 200021-103732/1.

Appendix

The consequences of floating points arithmetic error are well known in the simulation literature (see for instance [6]). In order to avoid this common pitfall we decided to implement our framework using both Matlab and C++. With continuous states the results obtained by two implementations were qualitatively the same even if numerically different.

Since exact replication of the experiments is obviously desirable we decided to have quantized states in order to obtain crossover results that were consistent between the two implementations.

While usually such effects are thought to be arising from accumulated floating point errors, in our case we found such discrepancies almost immediately.

In fact even with the seven-state simulation described in Section V our Matlab implementation incurred in some problems, when performing the arithmetical crossover.

For example, consider crossover between two agents with probabilities $1/2$ (state 3) and 0 (state 0). With the arithmetic crossover the offspring is $1/4$ and, in deciding the state of the offspring, two quantities must be compared: $1/4 - 1/6$ and $2/6 - 1/4$. Note that obviously in this case the two quantities are identical and a rule should be implemented for deciding the state of the offspring. The problem we encountered is that Matlab considers $1/4 - 1/6$ greater than $2/6 - 1/4$. With the C++ implementation we used long double i.e., floating-point data type with 80 bits of precision for our variables and the problem did not occur. Yet, since the accumulated floating point errors could not be ruled out completely, we decided to consider integer representation for the states.

Nevertheless a further step was in order. Since, due to the internal representation, the same fitness could be considered different depending on the implementation, we resolved to consider integer values for the fitness as well. This, to the best of our knowledge, solved our problems with the only drawback of imposing an upper bound on the number of states to be considered.

References

[1] Paul Harrald, "Evolving behaviour in repeated 2-player games," in *Practical Handbook of Genetic algorithms: Applications, Volume I*, Lance Chambers, Ed., pp. 459–496. CRC Press, Boca Raton, FL, 1995.
[2] J. Weibull, *Evolutionary Game Theory*, MIT Press, 1997.
[3] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Heidelberg, third edition, 1996.
[4] J. Maynard Smith, *Evolution and the Theory of Games*, Cambridge University Press, 1982.
[5] G. Alefeld and J. Herzberger, *Introduction to interval computation*, Academic Press, New York, New Jersey, 1983.
[6] J.G. Polhill, L.R. Izquierdo, and N.M. Gotts, "The ghost in the model (and other effects of floating point arithmetic)," *Journal of Artificial Societies and Social Simulation*, 2004, (to appear).