http://www.e-jemed.org/

ISSN : 1298-0137

e - JEMED
The Electronic Journal of Evolutionary Modeling and Economic Dynamics

Article number: 1021

Please cite this article as following:

Marco Valente, 2002, “Comments on the paper "Equilibrium Selection via Adaptation: Using Genetic Programming to Model Learning in a Coordination", by Chen, Duffy and Yeh”, *The Electronic Journal of Evolutionary Modeling and Economic Dynamics*, n° 1021, http://www.e-jemed.org/1021/index.php

---

Comments on the paper "Equilibrium Selection via Adaptation: Using Genetic Programming to Model Learning in a Coordination", by Chen, Duffy and Yeh

Marco Valente
University of L’Aquila

Abstract

This short note comments upon the paper mentioned in the title. The original work analyses a coordination game replicating some results observed in experiments by using Genetic Programming as artificial agents. My note suggests that the same result can be obtained by a much simpler structure, that is randomly generated agents.

Comments on the paper "Equilibrium Selection via Adaptation: Using Genetic Programming to Model Learning in a Coordination", by Chen, Duffy and Yeh

Marco Valente
Università dell'Aquila
mv@business.auc.dk

January 2, 2002

1 Introduction

The paper deals with a coordination game, where an individual agent's pay-off depends on the simultaneous strategies of all other fellow agents. The authors have the interesting goal of replicating the game (already explored via the usual analytical tools and also via experiments with real agents) using computer simulations. The system is represented by means of Genetic Programming, a powerful tool for problem solving searching in the space of the functions.

In these brief notes I try to sustain that whereas the authors claim to show that GP learns to behave human-like, I think that their results come mostly from the game set up, that is the environment where GP works, and not from the specific use of GP[1].

We can represent the game as an evolving population of strategies, where the GP "engine" produces the innovative elements in the population and the game set-up provides the selection process on the population. In my opinion, the model presented provides results depending exclusively on the "selection" (that is, the game's rewarding system) while the role of GP is just to add "noise" to the system.

Before presenting the results with which I substantiate my opinion, I think it is better to clarify briefly what is my approach to the use of simulations. I consider a simulation model useful when it provides insights on the behaviour of the system modelled. Normally, the results provided by simulation models are extremely general

[1] I will not comment upon the actual implementation of the simulation. The GP system (a very complicated object) seems to be implemented correctly, although the interface with the game structure may be improved.

and may have been obtained in many different ways. That is, we have many different potential computational structures that give rise to the same identical results. How to choose the computational structure mostly fitting with the results we want to present? I think that a concept of "parsimony" should be adopted, that is, use the simplest possible computational structure providing the results of interest. This rule may not be applied in certain cases. For example, when a model tries to represent a specific real system. In this latter case the focus is on the representational power of the model.

I think that the paper fails in not choosing one of the above principle. That is, definitely GP is never the simplest tool, and the results showed may be obtained in other ways. Nor the authors are able to substantiate that GP has much deeper similitude to human thinking other than providing the same numerical results. Concerning the first point above I present in the following an alternative version of the coordination game, providing, in my opinion, totally identical results in a much simpler way [2].

2 A simpler model

Concerning the paper, the model used consider a population of simulation agents each of which adopts a function to determine the "strategy" to play in the coordination game (that is, a value in the range [0,1]). The functions used by the agents are obtained in a quite complex way by means of a Genetic Programming structure, which generates endogenously the most fitting functions to solve a given problem. Eventually, the agents all play similar strategies, ending up in one of the two equilibria of the game.

The question, that I answer negatively, is the following: is GP the simplest possible computational structure giving rise to those results?

In order to test whether the use of GP is actually crucial for the results, I built a model with the same structure of the game as the one described in the paper (i.e. same definition of the fitness function for agents' strategies). Then I endowed a set of agents with randomly chosen initial strategies. At each cycle of the simulation agents are ranked according to the fitness values (function of their strategy and of the population average). The worst R agents are removed from the population and replaced with new ones, whose strategies are randomly selected. The remaining N-R agents continue to play the same strategy[3].

In other terms, I replicated the paper's model replacing the GP with a uniform random function. The results can barely be distinguished from the ones described in the paper. The average strategy played by agents converge to the expected

[2]Technical details are documented in the Lsd implementation of the model. See also the appendix below.

[3]For details on the test model see the *Coordination Game* model in the Lsd distribution, http://www.business.auc.dk/lsd.

equilibrium. The speed for reaching the equilibrium and the width of the oscillation around the average is determined by the number of agents selected at each time step. For example, Figure 1 reports two series with the mean values generated by two populations of 500 agents. In the first series 2 agents are replaced (i.e. switch randomly strategy) at each time step, while in the second series the agents replaced are 20. Note that the higher volatility in the second series provides also a slightly lower mean value, since the newly randomly chosen values for the replaced agents are more likely to be below the mean rather than above.

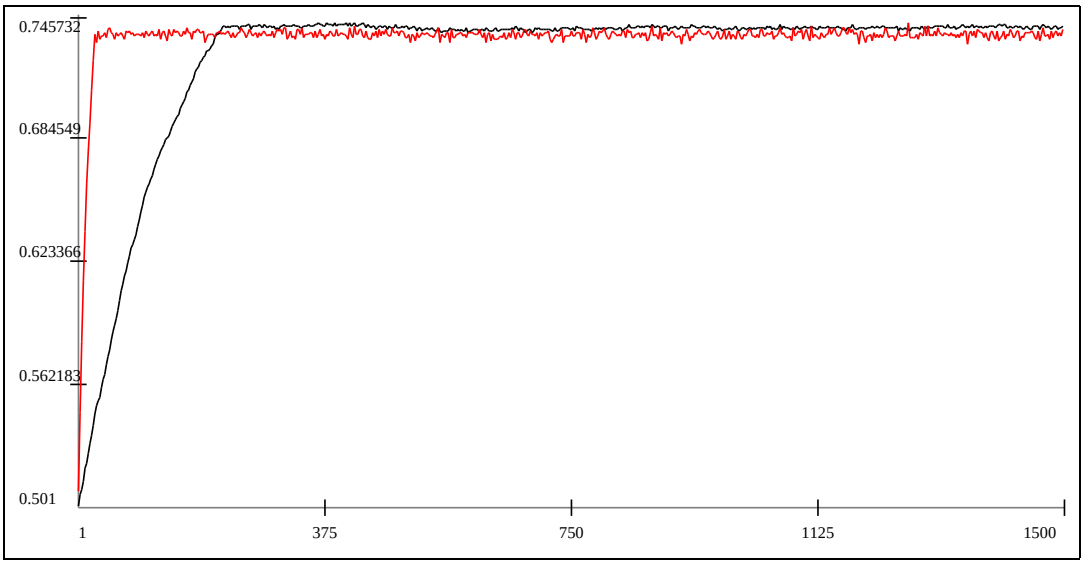

Figure 1: Time series of the average strategy played by the agents in two populations of 500 agents each with 2 replaced agents (black) and 20 (red).

A more evident effect of the number of replaced agents is given by the variance in the strategy played. In Figure 2 is reported the variance level as a function of the number of replaced agents (a test produced over 50 independent populations with replaced agents ranging from 2 to 100, and variance measured after 1500 time steps). Clearly, the variance increases with the number of replaced agents, since more noise it introduced when more agents are replaced.

In my opinion, the results mentioned above show that the dynamics of the model is governed by the game set up, and that the GP is merely a provider of "noise" so that enough variety is subjected to the selection process in order to find the "stable" solution.

3 Further considerations

One aspect of the use of GP in the paper does not seem very convincing. The GP is normally intended as a learning system made by several rules, generated and tested according specific procedures (basically, the usual genetic tools derived

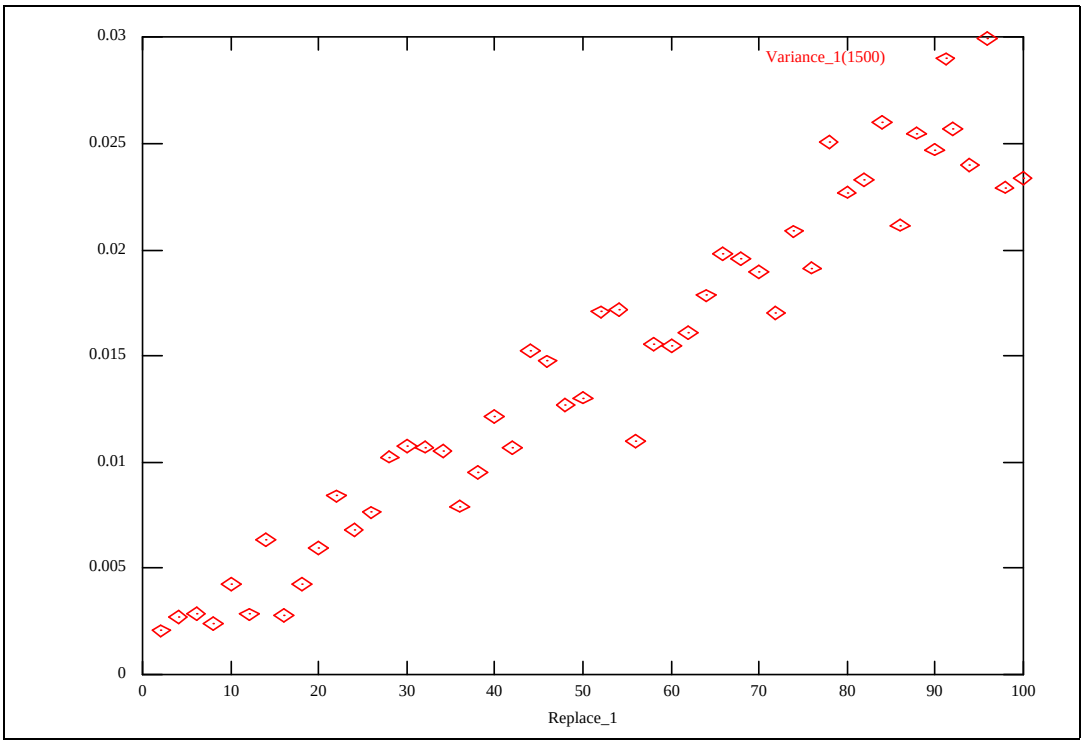

Figure 2: Variance around the mean strategy as a function of the number of replaced agents (measured after 1500 time steps).

from Genetic Algorithms). However, the authors seem to interpret a single rule in the GP system as a metaphor for an individual agent. In this case, the "learning entity" becomes the individual rule, with the whole system (generation of rules and selection mechanism) as the environment pushing for individuals to develop their skills. I don't have conclusive opinions on this aspects, although I think it may be interesting to develop: where does the learning springs from? in which cases we can consider a collective entity as a learning one?

Does the discussion above suggest that GP is never useful? I think the answer is negative, but it should be applied more appropriately. The game described in the paper is compared with other results obtained with human subjects. It may therefore be possible to use GP as a means of a model for real people's thinking. Saying that the GP and the people provide the same result, in a simple problem like this one, it is not enough. But a potential line of research would be to use, for example, interviews to the experimentees to be compared with the functional forms produced by the GP[4]. In this case, if similarities can be found, it may be possible to support the adequacy of the model not only on the result obtained (which would violate the principle of parsimony in reproduving given results mentioned before),

[4]This line of research is potentially open given the fact that GP manages symbolic representations of functions.

but on the basis of its capacity to reproduce the mental pattern triggered in the mind of people by the coordination game (satisfying the second principle).

As a totally different line of research, it may be possible to explore more extensively the "space" of the computational structures giving rise to the equilibrium results. That is, to try other types of (simple) models for agents such that the whole population converges (or not) to the equilibrium. For example, it may be possible to use, instead of the uniform random function, a normal random function centered on the previous period average, or other elaborations of this type[5]. It may also be interesting to distort the game in such a way that there is no equilibrium, for example suddenly changing, from time to time, the fitness function parameters. In other terms, a different approach to the problem discussed in the paper is to see "what happen if..." different computational structures are tested. In this case, it is necessary to provide clear and convincing explanations of why the model behaves in a certain way.

Appendix - The Coordination Game model distributed in Lsd

The test model is implemented in Lsd a language, based on C++, which provides the power and flexibility of C++ to write the computation part of the model, while being endowed with a complete set of interfaces to manage the model.

The installation of Lsd includes several example models, including the coordination game mentioned above, and a developing environment to create new models. Lsd can be downloaded from:

http://www.business.auc.dk/lsd

To run a model in Lsd after the installation run the *Lsd Model Manager* program. Choose to select a model and open the Coordination Game model. Open menu **Model** and choose **Run**. This will compile the relevant code and will launch the actual model program. This must be loaded with a configuration by means of the entry **Load** in menu **File**. The simulation run is launched with the entry **Run** in menu **Run**. After the simulation the data produced can be observed with the *Data Analysis* package (menu **Data/Analysis of Results**). Extensive documentation on the model is located in the menu **Help** of the Lsd model program. The use of Lsd models, and their creation, is extensively documented in the manual pages for Lsd (how to manage the simulation for a model) and LMM (how to create and modify the equations of a model). In LMM are also available three tutorials on Lsd introduction, model use and model creation respectively.

The configurations available concern one or more populations (independently simulated for comparison), with different parameterizations. To change the param-

[5]I implemented a preliminary version of this, and it converges much more rapidly to the equilibrium, as it may be expected

eters (with a after loading a fresh configuration) select the Lsd Browser on the object *Population* and choose menu **Data/Init. Value**.