ABCDE: Agent Based Chaotic Dynamic Emergence*

Pietro Terna

Università di Torino, Dipartimento di scienze economiche e finanziarie G.Prato, corso Unione Sovietica 218bis, 10134 Torino, Italia
terna@econ.unito.it

Abstract. This paper concerns agent based experiments in the field of negotiation and exchange simulation. A computer simulation environment is built, showing the emergence of chaotic price sequences in a simple model of interacting consumers and vendors, both equipped with minimal rules. "Swarm" is the framework of the model (www.santafe.edu/projects/swarm), a simulation tool with a strong object oriented structure, also very useful to separate in a clear way the model level from the level of the observer. Swarm if fully programmable in Objective C, with many powerful libraries, aimed at modeling the objects and the schedules of our experiments, with lists and arrays where necessary. Finally we introduce a tool (Cross Target method: CT), useful in building artificial laboratories, for experiments with learning, self-developed consistency and interaction of agents in artificial worlds, in order to observe the emergence of complexity without *a priori* behavioral rules: The perspective of our work is that of developing CT within the Swarm framework to replicate the ABCDE experiment in this light-rules or no-rules context.

1 Introduction

This paper starts with the definition of agent based models and then introduces the main problems arising in their construction, mainly focusing on software problems. Section 2 "Agent based models" underlines the usefulness of agent based models in the social science perspective, also focusing on the main computational problems (memory management and time management in simulation); Section 3, "The Agent Based Chaotic Dynamic Emergence (ABCDE)", introduces a small specific application of those techniques, modeling a consumer-vendor interaction experiment where chaos emerges as a side effect of the agents' behavior, in a spontaneous way, i.e. avoiding the use of equations to produce apparently non-deterministic data series when parameters lie in a particular range; Section 4, "The Self-development of Consistency

* This research has been supported by grants from the Italian Ministero dell'Università e della Ricerca scientifica e tecnologica, inside the project "Intermediazione finanziaria, funzionamento dei mercati ed economia reale".

in Agents' Behavior" introduces artificially intelligent agents founded upon algorithms that can be modified by a trial and error process without *a priori* behavioral rules.

The project is that of developing, in a parallel way, models built upon simple agents interacting in the Swarm context with light rules (without the CT technique) and with soft rules (mainly, the quest for consistency in the agent's behavior) always in the Swarm context with the CT technique.

2 Agent based models

The starting point is the choice on model foundations: If we choose the agent based model paradigm we enter a wide unexplored world where methodology and techniques are largely "under construction."

2.1 An Overview

An interesting overview comes from the following Web sites: Syllabus of Readings for Artificial Life and Agent-Based Economics[1]; Web Site for Agent-Based Computational Economics[2]; Agent-Based Economics and Artificial Life: A Brief Intro[3]; Complex Systems at the University of Buenos Aires[4]; Computational Economic Modeling[5]; Individual-Based Models[6]. With respect to social simulation we refer also to Conte et al. [1] and, in economics, to Beltratti et al. [2]; also of interest is The Complexity Research Project[7] of the London School of Economics and Political Science.

At present, the best plain introduction to agent based modeling techniques is in Epstein and Axtell [3]. For a thoughtful review of the book, see Gessler [4]. As Epstein and Axtell (Cap. 1) note:

"Herbert Simon is fond of arguing that the social sciences are, in fact, the hard sciences. For one, many crucially important social processes are complex. They are not neatly decomposable into separate subprocesses--economic, demographic, cultural, spatial--whose isolated analyses can be aggregated to give an adequate analysis of the social process as a whole. And yet, this is exactly how social science is organized, into more or less insular departments and journals of economics, demography, political science, and so forth (...)"

"The social sciences are also hard because certain kinds of controlled experimentation are hard. In particular, it is difficult to test hypotheses concerning the

[1] http://www.econ.iastate.edu/tesfatsi/sylalife.htm
[2] http://www.econ.iastate.edu/tesfatsi/ace.htm
[3] http://www.econ.iastate.edu/tesfatsi/getalife.htm
[4] http://www.cea.uba.ar/aschu/complex.html
[5] http://zia.hss.cmu.edu/econ/
[6] http://hmt.com/cwr/ibm.html
[7] http://www.lse.ac.uk/lse/complex/

relationship of individual behaviors to macroscopic regularities, hypotheses of the form: If individuals behave in thus and such a way--that is, follow certain specific rules--then society as a whole will exhibit some particular property. How does the heterogeneous micro-world of individual behaviors generate the global macroscopic regularities of the society?"

"Another fundamental concern of most social scientists is that the rational actor--a perfectly informed individual with infinite computing capacity who maximizes a fixed (nonevolving) exogenous utility function--bears little relation to a human being. Yet, there has been no natural methodology for relaxing these assumptions about the individual."

"Relatedly, it is standard practice in the social sciences to suppress real-world agent heterogeneity in model-building. This is done either explicitly, as in representative agent models in macroeconomics (Kirman, [5]), or implicitly, as when highly aggregate models are used to represent social processes. While such models can offer powerful insights, they "filter out" all consequences of heterogeneity. Few social scientists would deny that these consequences can be crucially important, but there has been no natural methodology for systematically studying highly heterogeneous populations."

"Finally, it is fair to say that, by and large, social science, especially game theory and general equilibrium theory, has been preoccupied with static equilibria, and has essentially ignored time dynamics. Again, while granting the point, many social scientists would claim that there has been no natural methodology for studying nonequilibrium dynamics in social systems."

The response to this long, but exemplary, quotation, is social simulation and agent based artificial experiments. According to the Swarm documentation[8] we introduce a general sketch about how one might implement an experiment in the agent based modeling field.

2.2 Tools useful in Social Sciences: Design and Characteristics

An idealized experiment requires, first, the definition of: (i) the computer based experimental procedure and (ii) the software implementation of the problem.

The first step is that of translating the real base (the physical system) of our problem into a set of agents and events. From a computational point of view, agents become objects and events become steps activated by loops in our program. In addition, in a full object oriented environment, time steps are also organized as objects.

We can now consider three different levels of completeness in the structure of our software tools.

– At the lowest level (i.e., using plain C) we have to manage both the agent memory structures (commonly with a lot of arrays) and the time steps, with loops (such as "for" structures) driving the events; this is obviously feasible, but it is costly (a lot of software has to be written; many "bugs" have to be discovered);

[8] http://www.santafe.edu/projects/swarm/

– At a more sophisticated level, employing object oriented techniques (C++, Objective C, etc.), we avoid the memory management problem, but we have nevertheless to run time steps via the activation of loops;
– Finally, using a high level tool such as Swarm, we can dismiss both the memory management problems and the time simulation ones; in high level tools, also the events are treated as objects, scheduling them in time-sensitive widgets (such as action-groups). The ABCDE model (see Sec. 3) introduced below is built according to these design techniques.

Obviously, there are many alternatives to a C or Objective C environment. More generally, we have to consider the multiplicity of tools aimed at developing agent based software applications. For an idea of these tools, visit the following "no name" Web sites: www.ececs.uc.edu/~abaker/JAFMAS/compare.html and the highly interesting Linux AI/Alife mini-HOWTO[9].

For a more complete discussion of this subject and for a discussion about agent based models and intelligent agents, see Terna [6].

In Appendix we report a direct reference to the Swarm environment, used in this work for the ABCDE experiment; the CT technique also (see Sec. 4) is introduced here in the perspective of rewriting it with Swarm.

3 The Agent Based Chaotic Dynamic Emergence (ABCDE)

This example concerns an agent based experiment in the field of negotiation and exchange simulation. One can replicate or modify the experiment applying Swarm (1.0.5 or above) to "make" an executable file from the content of the archive compressed file abcde-#_of_experiment_version-#_of_Swarm_version.tgz[10].

3.1 Chaos from Agents

The experiment shows the emergence of chaotic price sequences in a simple model of interacting consumers and vendors, both equipped with minimal rules.

Mainstream chaos supporters look for sets of equations that produce apparently non-deterministic data series when parameters lie in a particular range. But what about the plausibility of these synthetic constructions? In the ABCDE model we are not seeking to produce chaos: it emerges as a side effect of the agents' behavior.

[9] http://www.ai.uga.edu/~jae/ai.html

[10] This archive file can be obtained via email form the author or downloaded from the link contained in Ref. [6]; finally, in the future, may be from the "anarchy" archive in the Swarm site.

3.2 ABCDE Structure

There are ten consumers and ten vendors; in other words, we have twenty agents of two types. Each agent is built upon an object, i.e. a small Objective C program, capable of reacting to messages, for example deciding whether it should buy at a specific offer price. Both agents-objects-consumers and agents-objects-vendors are included in lists; the simulation environment runs the time, applying at each step the actions included in a temporal object (an action-group) and operates with the agents sending messages to their lists. There is a shuffler mechanism to change the order in which the agents operate and to establish random meetings of the members of the two populations.

At every simulation step (i.e., a tick of the simulation clock), artificial consumers look for a vendor; all the consumers and vendors are randomly matched at each step. An exchange occurs if the price asked by the vendor is lower than the level fixed by the consumer. If a consumer has not been buying for one or more than one step, it raises its price level by a fixed amount according to the counter rule and the sensitivity parameter introduced below. It acts in the opposite way if it has been buying and its inventory is greater than one unit.

A simulated vendor behaves in a symmetric way (but without a sensitivity parameter): it chooses the offer price randomly within a fixed range. If the number of steps for which it has not been selling is greater than one, it decreases the minimum and maximum boundaries of this range, and vice versa if it has been selling.

In detail, we have the following steps:

- At each time step *t*, each artificial consumer (an agent) meets an artificial vendor (another agent), randomly chosen.
- The vendor fixes its selling price, randomly chosen within a small range.
- The consumer accepts the offer only if the selling price falls below its buying price level.
- At each time step *t* each agent (consumer or vendor) increases its transaction counter by 1 unit, if it makes a transaction; it decreases it by 1 unit in the opposite case.
- When their counters are less than -1 (sensitivity = 0) or less equal than -1 (sensitivity = 1), *consumers* change their internal status: they raise their buying price by a fixed amount.
- When their counters are greater than 1 (sensitivity = 0) or greater equal than 1 (sensitivity = 1), *consumers* change their internal status in the opposite direction: they reduce their buying price by a fixed amount.
- When their counters are less than -1, *vendors* change their internal status: they reduce, by a random amount (from 0 to a fixed value) both the limits of the range within which they choose the selling price.
- When their counters are greater than 1, *vendors* change their internal status in the opposite direction: they raise, by a random amount (from 0 to a fixed value) both the limits of the range within which they choose the selling price.

3.3 Running the ABCDE Experiment

In all the experiments, the result is that the mean price behavior emerges as cyclical, with chaotic transitions from one cyclical phase to another. From a methodological point of view there are two kinds of emergence.

- Unforeseen emergence: While building the simulation experiment, we were only looking for the simulated time required to obtain an equilibrium state of the model with all the agents exchanging nearly at each time: The appearance of a sort of cyclical behavior was unexpected;
- Unpredictable emergence: Chaos is obviously observable in true social science phenomena, but it is not easy to make a reverse engineering process leading to it as a result of an agent based simulation.

We have now to define the parameters used in the eight experiments reported in the Figures of this paragraph. Parameters: "theLevel" is the initial price below which consumers buy (it changes independently for each consumer while the simulation evolves); "agentNumber" is the number of consumers and vendors; "minStartPrice" and "maxStartPrice" are the initial limits within which vendors choose their selling price (they change independently for each vendor while the simulation evolves); "use_printf" is a technical parameter to print internal values; the "reactivityFactor" is a multiplying factor that enhances the fixed values used by consumers and vendors to modify their prices or range of prices; "sensitivity" is zero or one, with the meanings introduced above.

In Table 1 we introduce the parameters used in the experiments presented in the following Figures.

Parameters and Experiments	Fig. 1	Fig. 2	Fig. 3	Fig. 4	Fig. 5	Fig. 6	Fig. 7	Fig. 8
theLevel	50	50	50	50	25	25	25	25
agentNumber	10	10	10	10	10	10	10	10
minStartPrice	45	45	45	45	70	70	70	70
maxStartPrice	55	55	55	55	90	90	90	90
use_printf	0	0	0	0	0	0	0	0
reactivityFactor	1	1	3	3	1	1	3	3
sensitivity	0	1	0	1	0	1	0	1
standard Swam random seed	yes	yes	yes	yes	yes	yes	yes	yes

Table 1. Parameters of the experiments reported in Fig. 1 to 8.

In the experiments reported in Fig.1 to Fig.4, the starting points are a buying price level of 50 (on a scale from 0 to 100) and a selling price range from 45 to 55 on the same scale. Initially, all the consumers and vendors have the same parameters, but during the simulation, they evolve on an individual basis. One could say that the memory of the system lies in the consumers/vendors random interaction.

The series reported in the Figures are the mean of global prices (all prices offered in each day or cycle) or the min. or max. within global prices. (The "alternative way" of the title is related to an internal calculus problem) [11].

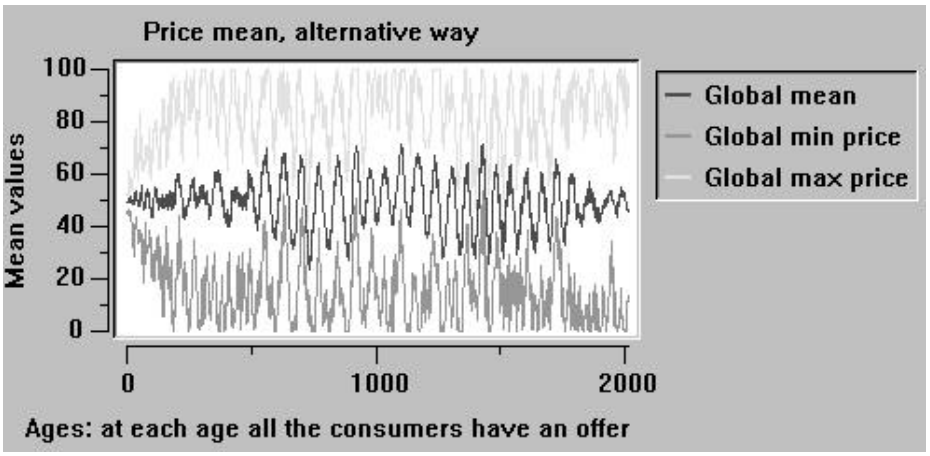

Fig. 1. Starting with a balanced situation, low reactivity and low sensitivity.

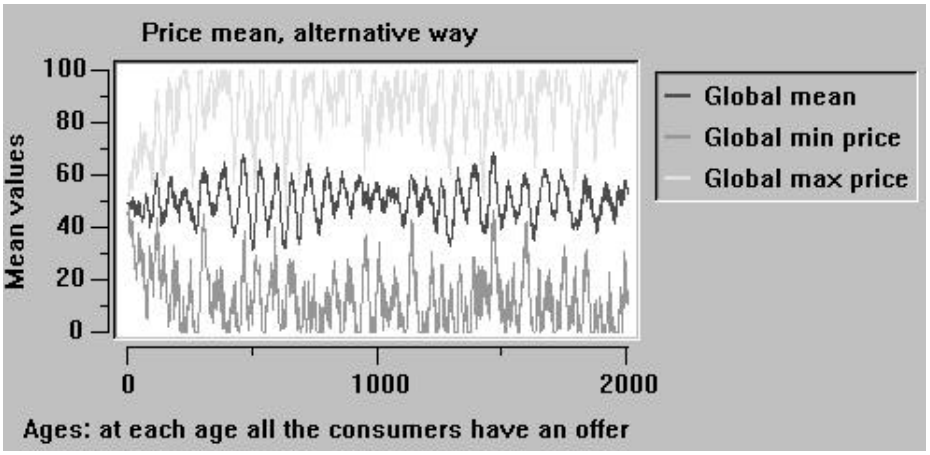

Fig. 2. Starting with a balanced situation, low reactivity and high sensitivity.

The experiments reported in Fig.1 and Fig.2 start form the balanced situation described at the beginning of the paragraph. With a low reactivity factor, the presence or the absence of the sensitivity parameter in the behavioral choices of the consumers only changes the amplitude of the fluctuations, but has no effect on the chaotic appearance of the global mean of the prices offered in each cycle.

[11] Technical note, to replicate the experiment use abcde-1_0_1-1_1 or abcde-1_0_2-1_2.

Technically: the FFT[12] of the series of data shows only one peak, related to the constant value; Lyapunov exponents are in the range 0.6-0.7 and both capacity and correlation dimensions are less than 5. In Fig. 2 data, in the range 501-1000, a strange attractor emerges in the singular value decomposition space. For a discussion concerning the meaning of chaos in these experiments, see Sec. 3.4.

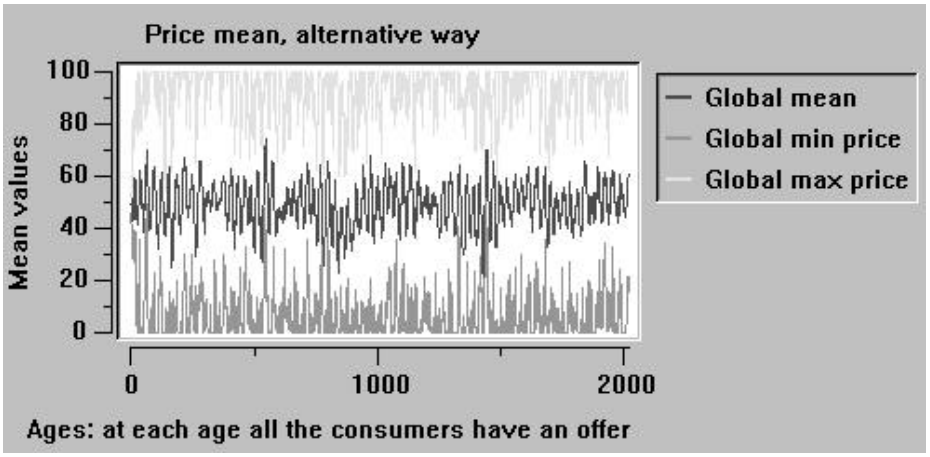

Fig. 3. Starting with a balanced situation, high reactivity and low sensitivity.

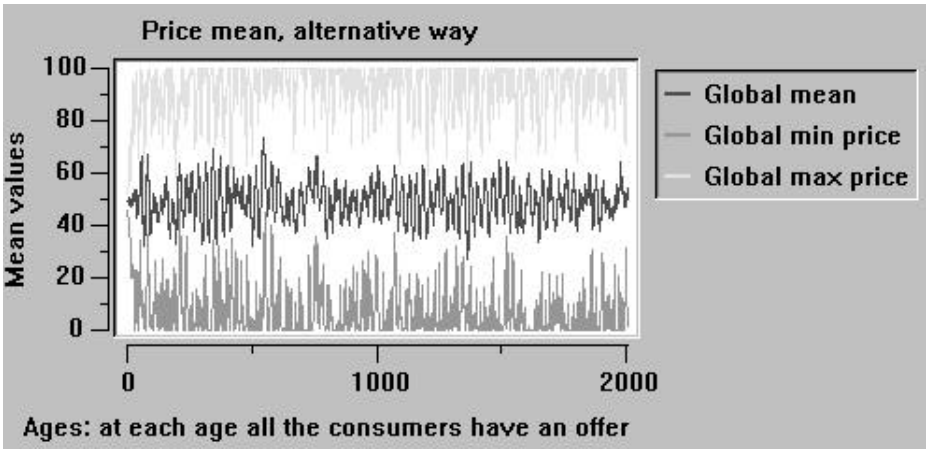

Fig. 4. Starting with a balanced situation, high reactivity and high sensitivity.

Also the experiments reported in Fig.3 and Fig.4 start form the balanced situation described at the beginning of the paragraph, but with a high reactivity factor, which improves the cyclical effect; the presence (Fig. 4) of the sensitivity parameter reduces the amplitude of the fluctuations; anyway, chaos appears.

[12] Fast Fourier Transformation.

The FFT shows the same results as above; Lyapunov exponents are in the range 0.5-0.8 and both capacity and correlation dimensions are less than 5. Data in Fig. 4 shows decidedly a strange attractor in the singular value decomposition space.

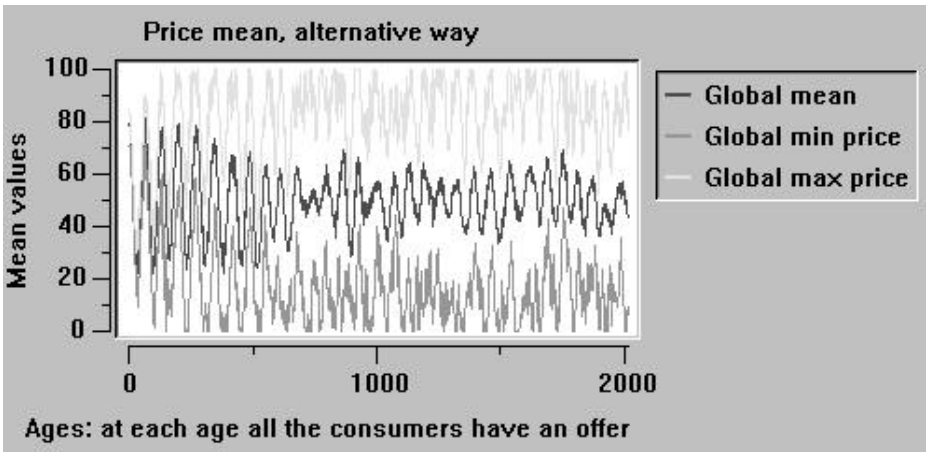

Fig. 5. Starting with an unbalanced situation, low reactivity and low sensitivity.

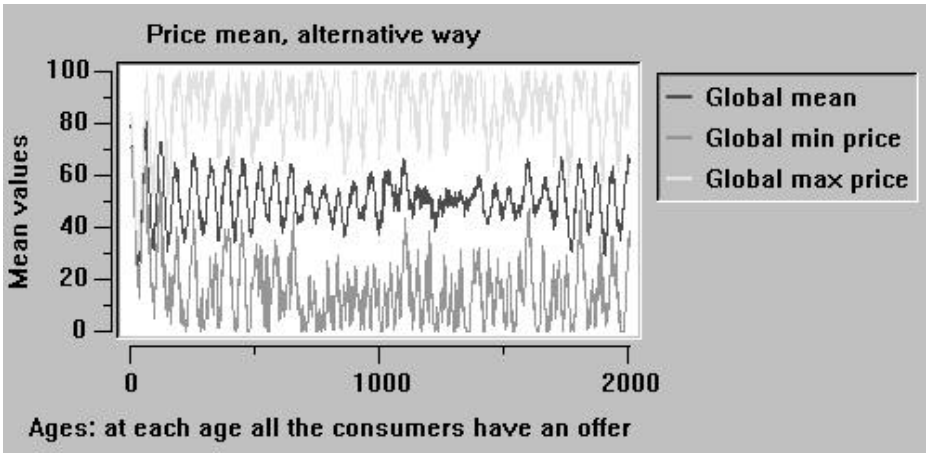

Fig. 6. Starting with an unbalanced situation, low reactivity and high sensitivity.

In the experiments reported in Fig.5 to Fig.8, the starting points are a buying price level of 25 (on a scale from 0 to 100) and a selling price range from 70 to 90 on the same scale; so we are in an unbalanced situation. With a low reactivity factor, in Fig.5 and Fig.6, chaos emergence is evident and the effect of the sensitivity parameter reinforces it.

The FFT shows the same results as above; Lyapunov exponents are in the range 0.4-0.5 and both capacity and correlation dimensions are less than 5.

With a high reactivity factor, in Fig.7 and Fig.8, cyclical effects appear to be more strong than the chaos emergence; anyway, introducing the sensitivity parameter, in Fig.8, chaos appears again, showing also a strange attractor (Fig. 8) in the singular value decomposition space.

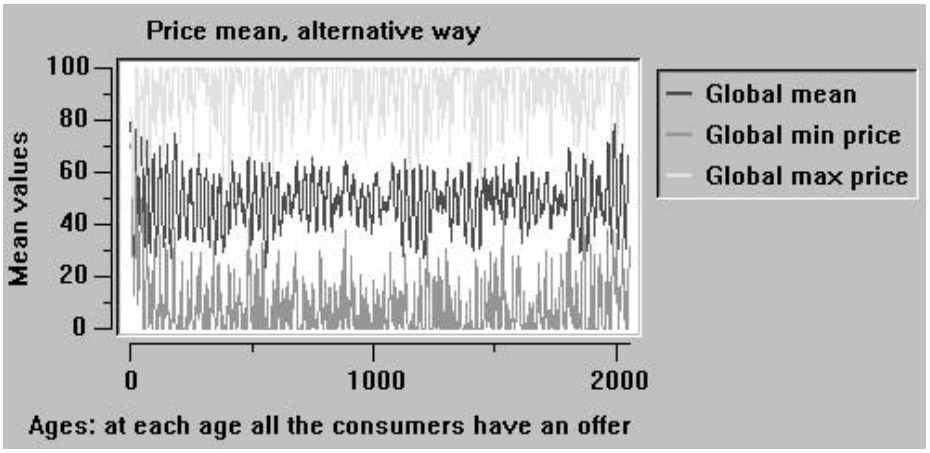

Fig. 7. Starting with an unbalanced situation, high reactivity and low sensitivity.

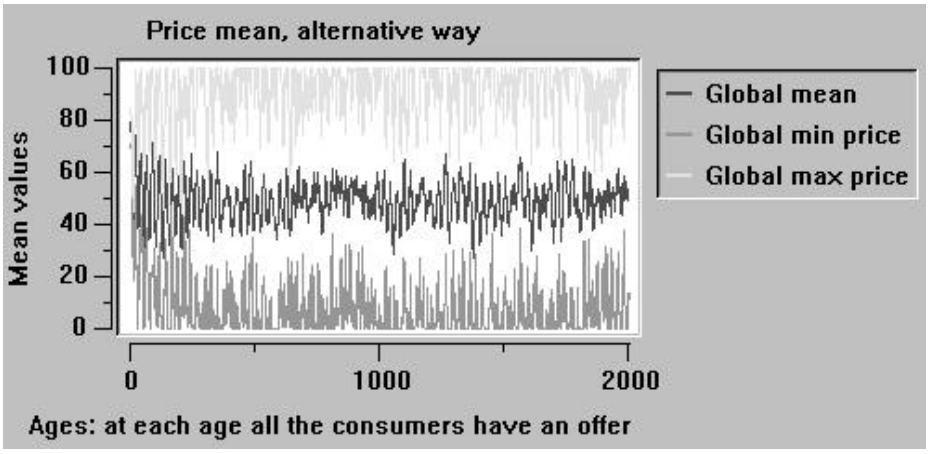

Fig. 8. Starting with an unbalanced situation, high reactivity and high sensitivity.

The FFT shows the same results as above; Lyapunov exponents are in the range 0.4-0.5 and both capacity and correlation dimensions are less than 5.

3.4 A general comment about chaos

A general comment about chaos in these experiments: we are here in presence of endogenous chaos, emerging from agent interaction. The price memory is diffused in

the agents, having a collective effect of synchronization or heterogeneous behavior in the agents' reaction to the exchange prices. Our model is not considering spatial coordinates, so an agent can bargain with another; with that the introduction of spatial distances and, as a consequence, of behavioral habitudes in agents' matches, the complexity of our structure can increase.

Chaos is here considered as random appearance of deterministic data; the measures reported above proof that our series are also technically chaotic; we know, by construction, that here chaos - or randomness - is not related to a well specified structure, but to collective behavior, which we are not able to describe ex ante in an analytical form. This is a form of chaos interesting for social sciences: the agent based chaotic dynamic emergence, or ABCDE.

4 The Self-development of Consistency in Agents' Behavior

With the purpose of generalizing the result obtained above, reducing the role of the user defined rules and, more generally, as a future perspective of work both in the agent based and in the Swarm contexts, in this Section we introduce the use of intelligent agents founded upon algorithms that can be modified by a trial and error process without *a priori* behavioral rules.

We remind that the project is that of developing, in a parallel way, models built upon simple agents interacting in the Swarm context with light rules (without the CT technique, as in Section 3) and with soft rules (mainly, the quest for consistency in the agent behavior) always in the Swarm context with CT technique.

In a soft rule context it is highly interesting also to replicate classical experiments built upon human agents, such as Chamberlin [7] and Smith [8], [9].

4.1 The Cross-Target (CT) Idea

We introduce the following general hypothesis (GH): an agent, acting in an economic environment, must develop and adapt her capability of evaluating, in a coherent way, (1) what she has to do in order to obtain a specific result and (2) how to foresee the consequences of her actions. The same is true if the agent is interacting with other agents. Beyond this kind of internal consistency (IC), agents can develop other characteristics, for example the capability of adopting actions (following external proposals, EPs) or evaluations of effects (following external objectives, EOs) suggested from the environment (for examples, following rules) or from other agents (for examples, imitating them). Those additional characteristics are useful for a better tuning of the agents in making experiments.

To apply the GH, we are at present using a tool employing artificial neural networks; the original program is developed in C language and will be transferred in the Swarm context. We observe, anyway, that the GH can be applied using other algorithms and tools, reproducing the experience-learning-consistency-behavior cycle with or without neural networks.

A general remark: in all the cases to which we have applied our GH, the preliminary choice of classifying agents' output in actions and effects has been useful (1) to clarify the role of the agents, (2) to develop model plausibility and results, (3) to avoid the necessity of *a priori* statements about economic rational optimizing behavior (see, for some examples, Beltratti et al. [2] and Terna [10]).

Economic behavior, simple or complex, can appear directly as a by-product of IC, EPs and EOs. To an external observer, our CT agents are apparently operating with goals and plans. Obviously, they have no such symbolic entities, which are inventions of the observer. The similarity that we recall here is that the observations and analyses about real world agents' behavior can suffer from the same bias. Moreover, always to an external observer, CT agents can appear to apply the rationality paradigm, with maximizing behavior.

Following the GH, the main characteristic of these CT agents is that of developing internal consistency between what to do and the related consequences. Always according to the GH, in many (economic) situations, the behavior of agents produces evaluations that can be split in two parts: data quantifying actions (what to do) and forecasts of the outcomes of the actions.

4.2 The Cross-Target Method

Choosing the artificial neural network tool to develop CT technique, we specify two types of outputs of the artificial neural network and, identically, of the CT agent: (1) actions to be performed and (2) guesses about the effects of those actions.

Both the targets necessary to train the network from the point of view of the actions and those connected with the effects are built in a crossed way, originating the name Cross Targets. The former are built in a consistent way with the outputs of the network concerning the guesses on the effects, in order to develop the capability to decide actions close to the expected results. The latter are similarly built with the outputs of the network concerning the guesses of the actions, in order to improve the agent's capability of estimating the effects emerging from the actions that the agent herself is deciding[13].

CTs, as a fulfillment of the GH, can reproduce economic subjects' behavior, often in internal “ingenuous” ways, but externally with apparently complex results.

The method of CTs, introduced to develop economic subjects' autonomous behavior, can also be interpreted as a general algorithm useful for building behavioral models without using constrained or unconstrained optimization techniques. The kernel of the method, conveniently based upon artificial neural networks, is learning by guessing and doing: control capabilities of the subject can be developed without defining either goals or maximizing objectives.

Fig.9 describes a CT agent learning and behaving in a CT scheme. The agent has to

[13] Guesses about actions are so always internally generated; at the beginning of the acting and learning phase, they are near random; then they are determined by the learning process, in a consistent way with effects guesses or estimations. External points of view can be superimposed to internal guesses (see EPs in Sec. 4.1).

produce guesses about its own actions and related effects, on the basis of an information set (the input elements are I_1, ..., I_k). Remembering the requirement of IC, targets in learning process are: (1) on one side, the actual effects - measured through accounting rules - of the actions made by the simulated subject; (2) on the other side, the actions needed to match guessed effects. In the last case we have to use inverse rules, even though some problems arise when the inverse function is undetermined.

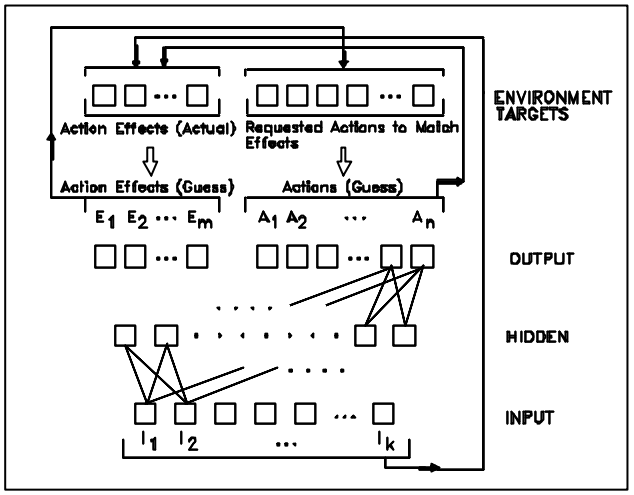

Fig. 9. The cross-target construction.

We think that it would be useful to develop the GH also in other ways, in case employing neither neural networks nor CT, to verify the reproducibility of our results in other contexts.

The flexibility in software structure necessary to perform this kind of methodological parallel experiment is the final reason for migrating, in the perspective of our future work, from a self developed tool as the present CT program, to a more standardized platform, such as Swarm.

4.3 CT: An example without Swarm

We introduce here an example of our CT experiments, related to our previous work, without the Swarm introduction. The presentation is aimed to facilitate the reader in considering the possible interest of our technique.

This first experiment is about motion of agents foraging for food (see Terna [10]). We apply the following scheme. On a plain with (x,y) coordinates, the subject is initially in (10,10) while the food is fixed in (0,0). The ANN (Artificial Neural Network) simulating the subject has the following inputs: X(t-1), position in the x direction at the time t-1; Y(t-1), position in the y direction at the time t-1; dX(t-1), step in the directions x, at time t-1 (bounded in the range ±1); dY(t-1), step in the directions y, at time t-1

(bounded in the range ±1).

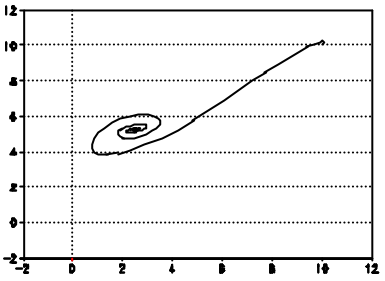

Fig.10. Moving toward food, without EO.

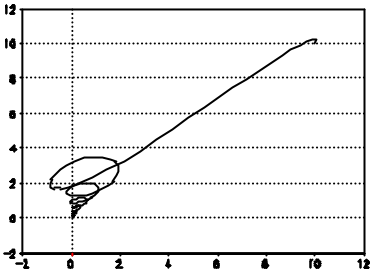

Fig.11. Moving toward food, with EO.

Using CT terminology, the ANN produces as outputs two guesses about effects and two guesses about actions. Guesses about effects are X(t), Y(t). Guesses about actions are dX(t) and dY(t), all with the same meaning of the input values. Positions X(t) and Y(t) have also the meaning of distance of the artificial subject from the food (distance evaluated employing rectangular coordinates). Summarising, the ANN representing the AAA has the following structure: 4 input, 6 hidden and 4 output nodes.

In Figure 10 we report the movement of the agent in 200 cycles of acting and learning. The agent goes toward the food on the basis of a simple implicit mechanism, which explains also the situation of locking in the middle of the path. The mechanism works in the following way: at the beginning of the experiment, the ANN produces random outputs, in a small interval around the central value between minimum and maximum. This effect is always present and is easily explained by considering the consequence of the initial random choice of the weights, that gives on average a null sum of the inputs of the sigmoidal transformation. In the case of the logistic functions, that input gives an output of about 0.5, corresponding to the mean between minimum and maximum values. As a consequence, the initial guesses about the effects of the movement give estimated positions around the central point where food is placed, with some variability.

In other term, the initial guess is that of being near the food. CTs immediately correct this wrong estimate, but they also correct the guesses about actions (the movements), to develop their consistency with the (wrong, but positively optimistic) guesses of effects. So, the artificial agent moves in the correct direction, but the process rapidly goes in a locking situation, with mutual consistency between effects and actions.

Now, imposing an EO on the side of the effects, that is the target of reducing in each cycle the distance from food to the 75% of the distance of the previous cycle, the food is easily gained, as reported in Figure 11. We underline that no suggestion is introduced about the direction of the movement.

Appendix: The Swarm Environment

In this Appendix we are turning more properly our attention to the Swarm world. The term "world" is appropriate, because Swarm is a system of software libraries, but also a vigorous interactive community working with them, as we can observe by taking part to the Swarm mailing lists[14].

In the Swarm context, we use the Object-Oriented Programming language Objective-C. According to the Swarm documentation, computation in a Swarm application takes place by instructing objects to send messages to each other. The basic message syntax is:

```
[targetObject message Arg1: var1 Arg2: var2]
```

where `targetObject` is the recipient of the message, `messageArg1:Arg2:` is the message to send to that object, and `var1` and `var2` are arguments to pass along with the message.

According to the Swarm documents: "The idea of Swarm is to provide an execution context within which a large number of objects can *live their lives* and interact with one another in a distributed, concurrent manner."

In the context of the Swarm simulation system, the generic outline of an experimental procedure takes the following form.

- Create an artificial universe replete with space, time, and objects that can be located, within reason, to certain "points" in the overall structure of space and time within the universe and allow these objects to determine their own behavior according to their own rules and internal state in concert with sampling the state of the world, usually only sparsely.
- Create a number of objects which will serve to observe, record, and analyze data produced by the behavior of the objects in the artificial universe implemented in the previous steps.
- Run the universe, moving both the simulation and observation objects forward in time under some explicit model of concurrency.
- Interact with the experiment via the data produced by the instrumentation objects to perform a series of controlled experimental runs of the system.

A remark about the consequences of publishing the result of a simulation: Only if we are using a high level structured programming tool, it is possible to publish simulation results in a useful way. Quoting again from Swarm documentation:

"The important part (. . .) is that the published paper includes enough detail about the experimental setup and how it was run so that other labs with access to the same equipment can recreate the experiment and test the repeatability of the results. This is hardly ever done (or even possible) in the context of experiments run in computers, and the crucial process of independent verification via replication of results is almost unheard of in computer simulation. One goal of Swarm is to bring simulation writing up to a higher level of expression, writing applications with reference to a standard set of simulation tools."

---

[14] http://www.santafe.edu/projects/swarm/mailing-lists.html#support

For this, the fact that the Swarm structure has two different levels is very useful. There is the model level (and we can have nested models of models, or swarms of swarms) and the observer level which considers the model (or the nested models) as a unique object to interact with, in order to obtain the results and to send them to various display tools and widgets.

Finally, the diffusion effect is a cumulative one, both for the production of reusable pieces of programs and for the standardization of techniques to allow experiments to be replicated easily.

Obtaining and Using Swarm: Swarm is developed at the Santa Fe Institute[15] and it is freely available[16], under the terms of the GNU[17] license. Swarm runs under Unix / Linux operating system and, more recently, under Windows 95 and Windows NT, using the "GNU-Win32: Unix for Win32[18]."

References

1. Conte, R., Hegselmann, R., Terna, P. (eds.): Simulating Social Phenomena. Springer, Berlin (1997).
2. Beltratti, A., Margarita S., Terna P.: Neural Networks for Economic and Financial Modelling. ITCP, London (1996).
3. Epstein, M.E. and Axtell, R.: Growing Artificial Societies - Social Science from the Bottom Up. Brookings Institution Press, Washington. MIT Press, Cambridge, MA(1996).
4. Gessler, N.: Growing Artificial Societies - Social Science from the Bottom Up. Artificial Life **3** (1997) 237-242.
5. Kirman, A.: Whom or What Does the Representative Agent Represent. Journal of Economic Perspectives **6** (1992) 126-39.
6. Terna, P: Simulation Tools for Social Scientists: Building Agent Based Models with SWARM. Journal of Artificial Societies and Social Simulation **2** (1998) <http://www.soc.surrey.ac.uk/JASSS/1/2/4.html>.
7. Chamberlin, E.: An Experimental Imperfect Market. Journal of Political Economy, April (1948) 95-108.
8. Smith, V.: An Experimental Study of Competitive Market Behavior. Journal of Political Economy, April (1962) 111-137.
9. Smith, V.: Microeconomic Systems as Experimental Science. American Economic Review **5** (1982) 923-955.
10. Terna, P.: A Laboratory for Agent Based Computational Economics: The Self-development of Consistency in Agents' Behaviour. In Conte, R., Hegselmann, R., Terna, P. (eds.): Simulating Social Phenomena. Springer, Berlin (1997).

[15] http://www.santafe.edu

[16] http://www.santafe.edu/projects/swarm/

[17] http://www.fsf.org/

[18] http://www.cygnus.com/misc/gnu-win32/