Artificial experiments in economics, with agent based models

Pietro Terna

Dipartimento di Scienze economiche e finanziarie G.Prato, Università di Torino

terna@econ.unito.it

Abstract

We introduce here a small set of artificial experiments in social science and a simplified tool useful to develop them in the perspective of the analysis of complex systems, with different degrees of cognitive definition of agents' behavior. The straightforward way to build this kind of structures requires the use of agent based simulation techniques, i.e. models in which small software routines behave representing artificial agents within artificial environments, and ecosystems are built that allow the emergence of institutions such as market, organizations, hierarchal structures. jESOF (java Enterprise Simulation Open Foundations, web.econ.unito.it/terna/jes) is a tool created with the aim of simplifying for social scientists the access to agent based simulation, that does not require to be high level specialized computer scientists.

This kind of research is in a middle point amid an applied work and a theoretical speculation because we can use this line of analysis both to model actual situations, analyzing also the effects of changes, and to build abstract structures of interacting units, firms or districts, to simulate emerging behavior. The application to the actual world is also useful to collect facts and situations to be used as stylized template into the theoretical framework.

The two examples presented in this analysis show that it is relatively easy to develop models with jESOF and that we are virtually unconstrained in the definition of the related contents. The first model is a test implementation of the well known Preys Predators Model (PPM) with the capability of introducing non only two, but three or more interacting levels (hyper-predators etc.). The second model is based upon two coevolving population: Workers, with their Skills, and Firms (WSF): the two populations have quasi-independent behaviors, based on a closely bounded rationality paradigm.

In perspective, the goal is that of discovering what happens if we introduce a more consistent behavior in a cognitive perspective, also evolving agents' rules with soft computing techniques such as classifier systems or neural networks, using in this case the Cross Targets technique.

General Hypothesis (GH) for agent based models

To develop cognitive agent based experiments, 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 example, 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.

The GH is in the background of the models introduced here, using the general tools provided by the jES and jESOF structures. GH and IC will be improved using the experimented Cross Target Technique (fig. 1 and below Appendix II).
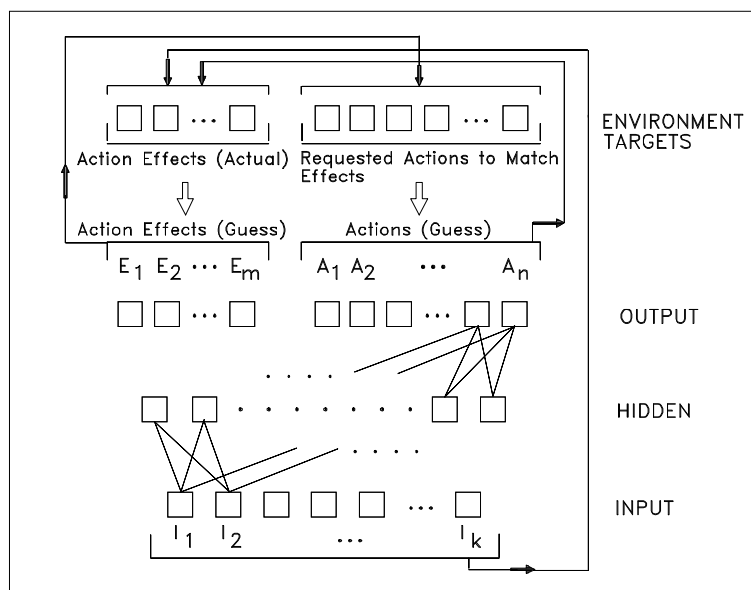

Fig. 1. The Cross Target scheme (see Appendix II).

Simulating organizations: from jES to jESOF

jESOF has been developed upon the Swarm (www.swarm.org) simulation library, from jES (java Enterprise Simulator, same source as above). With our tool we describe in a detailed way a two side world, considering both the actions to be done, in terms of orders to be accomplished (the "What to Do" side, WD), and the structures able to do them, in terms of units or agents (the "which is Doing What" side, DW). Our simulation framework is, first of all, a description of enterprises, organizations, entities, agents and their networks, as they are. Just like the various flight simulator programs put at our fingers the control of the simulated airplane and then execute our choices, jES and jESOF executes exactly what we suggest has to take place into the simulated enterprises or organizations or agents, on the two sides described above.

With the simulator we can reproduce in a detailed way the behavior of a firm, of an organization, of a system of firms and organizations into a computer, specially if we build the simulation model employing agent based techniques (Axtell, 2000; Terna and Gilbert, 2000; Tesfatsion, 2001; Gilbert and Troitzsch, 2005).

To run enterprise and organization simulations we introduced a few years ago a Swarm based framework, the Java Enterprise Simulator (*jES*, http://web.econ.unito.it/terna/jes). With our tool we describe in a detailed way a two side world, considering both the actions to be done, in terms of orders to be accomplished (the "What to Do" side, WD), and the structures able to do them, in terms of production units (the "which is Doing What" side, DW). Our simulation model is, first of all, a description of the enterprise or of the organization, as it is. Just like the various *flight simulator* programs put at our fingers the control of the simulated airplane and then execute our choices, *jES* executes exactly what we suggest has to take place into the simulated enterprise, on the two sides described above. The plane flights or lands; the enterprise produces or stays clogged if our WD and DW choices are inconsistent.
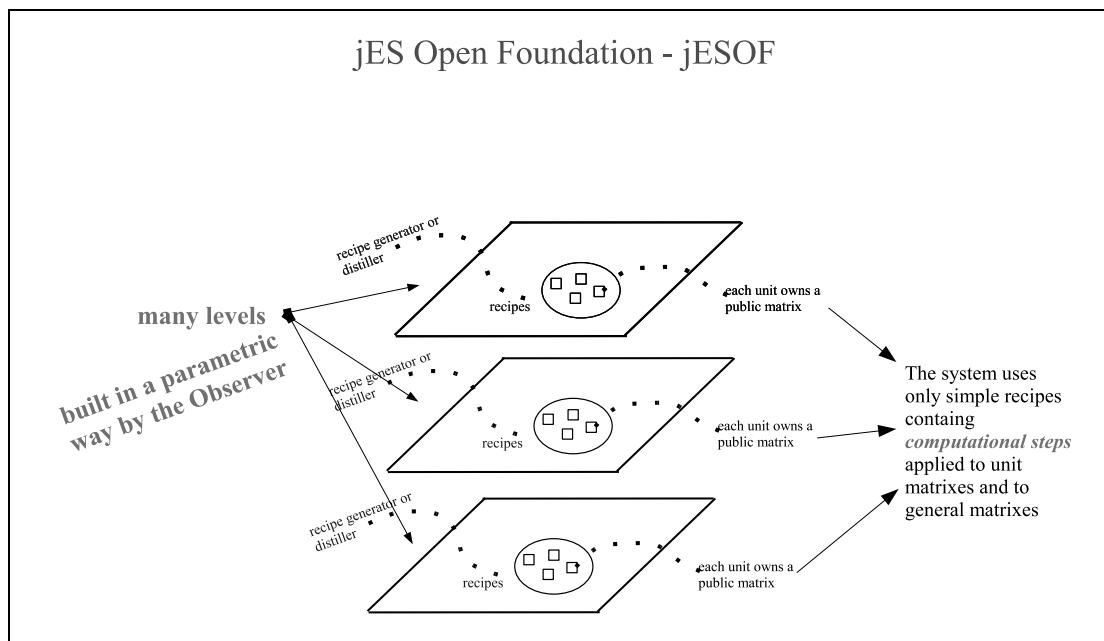

Fig. 1. The conceptual scheme of *jESOF*.

The basis of the method has to be found into agent based simulation techniques, i.e., the reconstruction of a phenomenon via the action and interaction of minded or no minded agents within a specific environment, with its rules and characteristics. In our *jES* case, the world is populated with the things to be done (orders) and the units able to work with them; we have also minded agent s, i.e. the agents expressing decisions within the model.

The step ahead is *jESOF* (jES Open Foundation), a *jES* generalization built to simulate multi-model frameworks of system of units or agents. Both packages are based on Swarm (www.swarm.org), the first multipurpose library of functions for building agent based simulation models showing: (i) a clear distinction between the model layer and the observer layer; (ii) a direct tool to deal with time and events, via an object oriented construction of the groups of actions and of the related schedules; (iii) a system of probes to be used to look inside agents while the model is running and the agent are behaving into the artificial environment. *jESOF* simulates systems of enterprises or production units (or systems of units without direct economic meanings, as in the case of the preys-predators model) in static or evolutionary contexts. In this second case new unit-agents arise continuously and some of the old are dropped out.

In Fig. 2 we present the conceptual scheme of *jESOF*, with several interacting models; in particular the scheme shows: (i) units: a unit is able to perform one of the steps required to accomplish an order, as a productive structure or, more generally, as an agent; (ii) orders, representing goods or services to be produced or actions to be done; an order contains technical information and their recipes, i.e. the sequence of steps to be accomplished to perform them; (iii) sequences of orders, coming from archives and managed via an "order distillers" or randomly generated via an "order generators"; may also concern the construction or elimination of units; (iv) matrixes to manage data, both at a unit level and at a general environment level; recipes, as sequences of steps, refer to matrixes running computational steps; (v) time schedule, managing "order distillers" and "order generators" actions, one for each layer or model. All this features are driven through to a specialized time oriented script language, incorporated in text files and spreadsheet files; you can find the formalism in a document contained in *jESOF* distributions (look at http://web.econ.unito.it/terna/jes_files/ for the latest *How to use jES_O_F* file). The key of the script language is constituted by the recipes, simple as sequences of single acts or complex ones, calling computational functions written in Java, which are useful: to make complicated steps such as prey-predators or firms-workers interactions; to account for the actions of the different units etc.

The Preys Predators Model (PPM)

About the models introduced in this paper, we underline first of all the presence of *soft* or *hard* links or connections in agent interaction.

As we can see below, workers and firms are obviously connected, but with a form of loose relationships, where firms are hiring daily the workers specified via a simulation supplied sort of throughput and workers can work daily also for more than one firm; firms are able to store the excess of work, if any, and to employ the stored quantities subsequently. In the future a new version of the model will consider a strictly accounted form of relationships, implying that only necessary workers are hired in each production period and that the maximum work capabilities are controlled (future version). The PPM works in a opposite way, because each predator agent is supposed to eat exactly one prey in each cycle, if it is possible to find one of them. It is interesting to underline that both models are producing results with interesting results, offering some emergence insight, and suggesting the possibility to explore, as a theme of research, the level of strength to be placed in stylized agent links. PPM uses fixed rules in a deterministic way and complexity comes from the characteristics of the problem and of its environment.

We introduce now the preys predators model, with three levels instead of the classical two. A first level contains the grass; the grass grows at a fixed rate applied to the existing units; the new units appear near the existing ones. A second level contains the preys, or rabbits, eating the grass; in case of shortage of grass, rabbits disappear. The third level contains foxes, eating rabbits; in case of shortage of rabbits, foxes disappears. New rabbits and foxes appear at a fixed rate also in these cases applied to the existing units; the new units appear near the existing ones. If a level becomes empty, it is impossible to fill it again, due the rate of growth mechanism. Rabbits eat the grass and foxes eat the rabbits only if the units are reciprocally visible, i.e. if their light areas in fig.3 have in common at least one point.

Look at the complex codetermination of grass, preys and predators in the figure: with this model it is easy to produce complex evolutionary scenarios, like that in which predators disappear and after few cycle grass goes to a minimum level, due the excess of prey quantity.
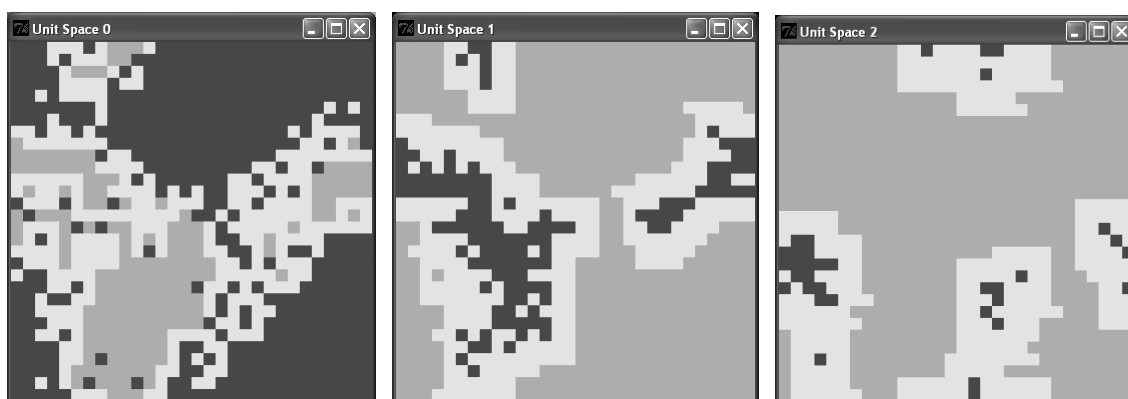
Fig. 3. The grass, on the left; the rabbits, in the middle; the foxes, on the right. Dark area are agents of the three types, light area report the visibility space of the agents.

We introduce here this well known model to compare the hard link used among agents in this case to the soft one described in the WSF model. But we introduce it also to demonstrate the easiness of jESOF programming, with the characteristic reported in the Appendix I.

Workers, Skills, Firms (WSF) model

The model has five strata: stratum 0 for the workers (all together), which are separately represented also in strata 1, 2 and 3 according to their different (three) skills; stratum 4 contains the enterprises. The starting point is the global worker stratum (stratum 0), with an initial presence of all the types of workers (types 1, 2 and 3; types and skills are coincident). The environment can be interpreted as a (social) space with both (i) metaphorical distances, representing trustiness and cooperation among production units (the social capital), or (ii) physical distances among the several units.

The activity, in an abstract sense, is represented by sequences of orders; each order contains a recipe, i.e. the description of the sequence of steps to be done by several units to complete a specific production or action. Two units can cooperate in the production process only if they are mutually visible in our (social) space, i.e. when their visibility areas (growing around them) are overlapping. Units that do not receive a sufficient quantity of orders, as well as the ones that cannot send the accomplished orders to successive units, disappear. New enterprises continuously arise, in the attempt of filling the holes of our social network. A complex structure emerges from our environment, with a difficult and instable equilibrium whenever the social capital is not sufficient.
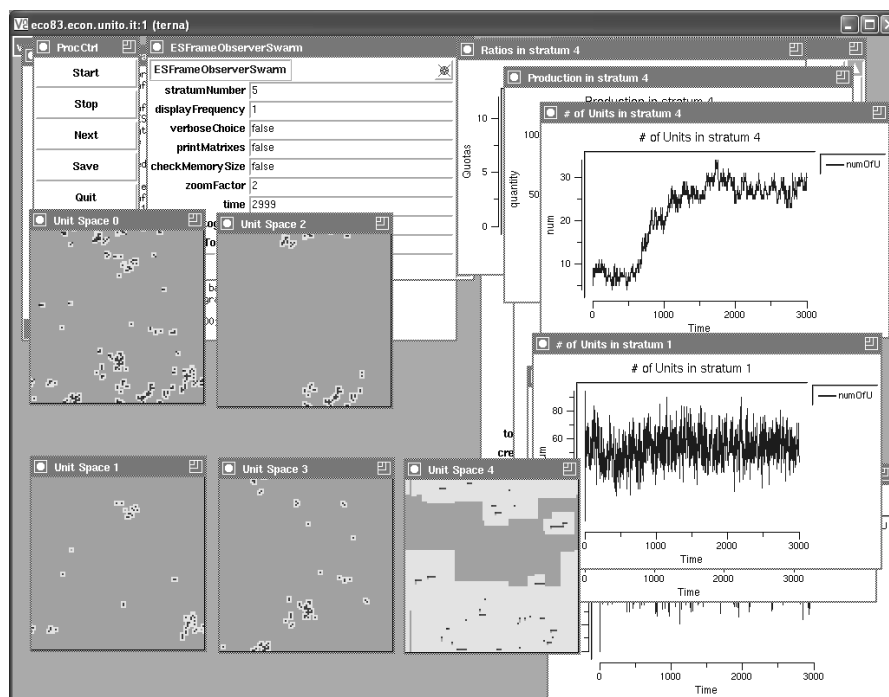
Fig. 4. Workers concentration by type of skill, with a relatively slow takeover of the economic system.

The scrip language uses two different sets of recipes included in orders. In the firm stratum we have recipes related to production, with sequences of steps describing the good to be produced. In the workers stratum, which is also the interaction place, recipes produce five kinds of effects: (i) new workers appear in the simulation context, either near to similar ones, or randomly distributed; (ii) firms hire workers and recipes modify workers and firms private matrixes; this is done accounting for both the availability of the labor production factor (firm side) and household income (workers side); (iii) firms make use of available labor production factors; (iv) firms either short of orders to be processed, or lacking adequate workers on the market, or being unable to deliver produced goods disappear from the economic scenario; (v) workers also disappears if unable to find a firm for prolonged time. Recipes are able to perform complex tasks, such as those described above, and are developed via computational steps. These steps can be interpreted as calls to code functions (methods of a Java class) invoked by the scrip language.

We can observe also the consequences of workers localizations in terms of system development (number of firms). When workers are clustered (by skill type), we observe a slow takeover of the economic system; by contrast when worker are randomly located we can observe a fast initial development of the economic system with an immediate stability of larger districts. This effect can be easily observed in the figs. 4 and 5, where the graphs of workers (stratum 1) and firms (stratum 4) are compared under the two situations.
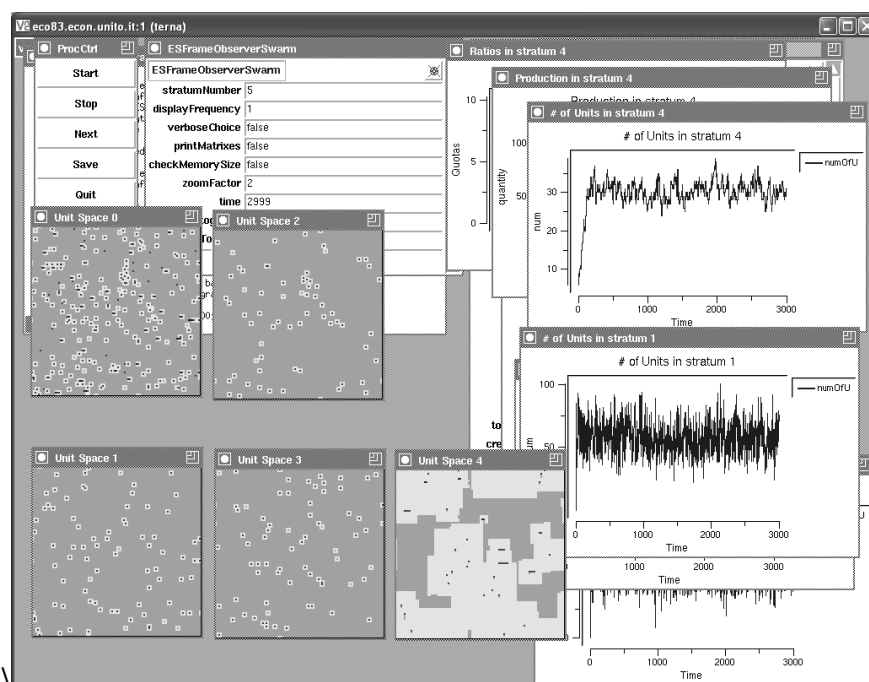
Fig. 5. Random diffusion of different skill workers, with a rather fast initial development of the economic system.

The orders coming from the market and their attribution to different firms determine cycles both in the workforce and the production. While in the relevant literature the frequency of cycles in the labor market is considered a challenging puzzle (see Hall, 2005; Shimer, 2005) in our model it is possible both to give a temporal dimension to cycles length and to give a theoretical interpretation of their occurrence. If three months of inactivity for real firms are assumed to be sufficient for default, the inactivity parameter we use in the model allows us to find the temporal dimension of our simulations. For instance, when setting the inactivity default parameter to 15 turns, a cycle of 1200 periods corresponds to about 20 years of actual time; in this span of we could observe (fig. 5) about 8 turns and this is not too far from the empirical evidence.

We underline that all the scripting mechanism necessary to obtain this complex result is very similar to those of the Appendix I, plus a few small Java codes, specialized to solve the computational step cited above. In the distribution package of jESOF, folder apps/workers_skills_firms, you can find all the necessary file to reproduce the experiment.

Future development

In perspective, the goal of the WSF model is that to discover what happens if we introduce a more consistent double behavior of workers and firms in a concrete cognitive perspective, also by evolving agents' rules through soft computing techniques such as classifier systems or neural networks, using in this case the Cross Targets technique (Terna, 2000), summarized at web.econ.unito.it/terna/ct-era/ct-era.html.

The WSF model can be improved introducing a population of banks, again with both a realization based on quasi-independent behavior and one on a structured cognitive solution, using CT (see above, Appendix II).

Appendix I - jESOF programming

Always at http://web.econ.unito.it/terna/jes/ look for the last version of jESOF, named in the form jesopenfoundation-x.y.zz.tar.gz; inside the distribution file you will find a folder apps/tutorial/ step3b_the_predators with the file running the example reported before. The programming code of the application is here reported in the spreadsheet of figs. 6, 7 and 8, respectively for grass, rabbits and foxes.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	#	Recipes	;										
2		grassGeneration	1	c	1201	2	0	0	1	s	0	;	
3		utility	100	c	1100	3	0	1	2	1	s	0	;

Fig. 6. Coding system for grass.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	#	Recipes	;												
2		preysEating	1	1001	s	0	c	1202	2	1	1	1	s	0	;
3		timeStep	23	c	1299	2	1	1	1001	s	0	;			
4		preysReproducing	4	c	1206	2	1	1	1001	s	0	;			

Fig. 7. Coding system for rabbits.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	#	Recipes	;												
2		predatorsEating	1	2001	s	0	c	1202	2	2	2	1001	s	0	;
3		timeStep	23	c	1299	2	2	2	2001	s	0	;			
4		predatorsReproducing	4	c	1206	2	2	2	2001	s	0	;			

Fig 8. Coding system for foxes.

To study the rules of this script language please refer to the "How" to document cited above.

Langton ants (Steward, 1994) application is another significant example of the flexibility of jESOF; you can find the code in the folder apps/LangtonAnts.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	#	Recipes	;										
2		antMoving	1	c	1301	2	0	0	1	s	1	;	
3		utility	100	c	1100	3	0	1	2	1	s	0	;
4		addingAnts	2	c	1104	2	1	1	1	s	0	;	
5		addingMarks	3	c	1104	2	1	1	1001	s	0	;	

Fig. 9. Coding system for Langton's ants.

With a few lines of scripting code in jESOF (fig. 9) and the use of the related Java computational steps, we can easily obtain the classical result of fig.10, here with more than one ant simultaneously acting.
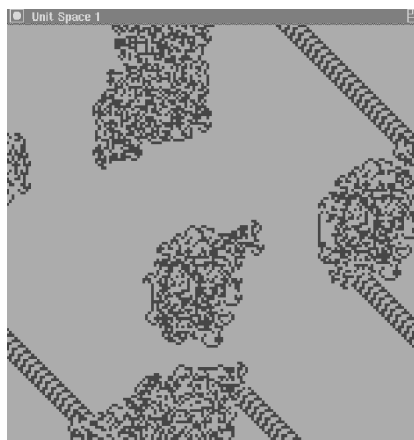

Fig. 10. The typical Langton's ants behavior reproduced with jESOF.

Appendix II - The Cross Targets (CT) technique

Following Beltratti et al. (1996), to apply the GH to the models shown above it will be highly useful to use artificial neural networks; 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.

We have here internal consistency (IC), capability of adopting actions (following external proposals, EPs) or evaluations of effects (following external objectives, EOs) suggested from the environment (for example, following rules) or from other agents (for examples, imitating them). Economic behavior, simple or complex, can appear directly as a by-product of IC, EPs and EOs. To an external observer, our Artificial Adaptive Agents (AAAs) 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, AAAs can appear to apply the rationality paradigm, with maximizing behavior.

The main problem is: obviously agents, with their action, have the goal of increasing or decreasing something, but it is not correct to deduce from this statement any formal apparatus encouraging the search for complexity within agents, not even in the *as if* perspective. With our GH, and hereafter with the Cross Target (CT) method, we work at the edge of Alife techniques to develop Artificial Worlds of simple bounded rationality AAAs: from their interaction, complexity, optimizing behavior and Olympic rationality can emerge, but externally to the agents.

In order to implement this ideal target without falling in the trap of creating models that are too complicated to be managed, we consider artificially intelligent agents founded upon

algorithms which can be modified by a trial and error process. In one sense our agents are even simpler than those considered in neoclassical models, as their targets and instruments are not as powerful as those assumed in those models. From another point of view, however, our agents are much more complex, due to their continuous effort to learn the main features of the environment with the available instruments.

The name Cross-Targets (CTs) comes from the technique used to figure out the targets necessary to train the ANNs representing the artificial adaptive agents (AAAs) that populate our experiments (fig. 1, above).

Following the GH, the main characteristic of these AAAs 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. So we specify two types of outputs of the ANN and, identically, of the AAA: (i) actions to be performed and (ii) 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 of the effects, in order to develop the capability to decide actions close to the expected results. The latter, similarly, are built in a constant way 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.

CTs, as a fulfillment of the GH, can reproduce economic subjects' behavior, often in internally ingenuous ways, but externally with 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 ANNs (but it could also be conceivable with the aid of other mathematical tools), is learning by guessing and doing: the subject control capabilities can be developed without defining either goals or maximizing objectives.

We choose the neural networks approach to develop CTs, mostly as a consequence of the intrinsic adaptive capabilities of neural functions. Here we will use feed forward multilayer networks.

Fig. 1 describes an AAA learning and behaving in a CT scheme. The AAA has to produce guesses about its own actions and related effects, on the basis of an information set (the input elements are I1,...,Ik). Remembering the requirement of IC, targets in learning process are: (i) on one side, the actual effects - measured through accounting rules - of the actions made by the simulated subject; (ii) 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 is indeterminate.

A first remark, about learning and CT: analyzing the changes of the weights during the process we can show that the matrix of weights linking input elements to hidden ones has little or no changes, while the matrix of weights from hidden to output layer changes in a

relevant way. Only hidden-output weight changes determine the continuous adaptation of ANN responses to the environment modifications, as the output values of hidden layer elements stay almost constant. This situation is the consequence both of very small changes in targets (generated by CT method) and of a reduced number of learning cycles.

The resulting network is certainly under trained: consequently, the simulated economic agent develops a local ability to make decisions, but only by adaptations of outputs to the last targets, regardless to input values. This is short term learning as opposed to long term learning.

Some definitions: we have (i) short term learning, in the acting phase, when agents continuously modify their weights (mainly from the hidden layer to the output one), to adapt to the targets self-generated via CT; (ii) long term learning, ex post, when we effectively map inputs to targets (the same generated in the acting phase) with a large number of learning cycles, producing ANNs able to definitively apply the rules implicitly developed in the acting and learning phase.

A second remark, about both external objectives (EOs) and external proposals (EPs): if used, these values substitute the cross targets in the acting and adapting phase and are consistently included in the data set for ex post learning. Despite the target coming from actions, the guess of an effect can be trained to approximate a value suggested by a simple rule, for example increasing wealth. This is an EO in CT terminology. Its indirect effect, via CT, will modify actions, making them more consistent with the (modified) guesses of effects. Vice versa, the guess about an action to be accomplished can be modified via an EP, affecting indirectly also the corresponding guesses of effects. If EO, EP and IC conflict in determining behavior, complexity may emerge also within agents, but in a bounded rationality perspective, always without the optimization and full rationality apparatus.

References

Axtell, R. (2000). Why agents? On the varied motivations for agent computing in the social sciences. *Forthcoming. Center on Social and Economic Dynamics*, Working Paper No. 17, www.brook.edu/es/dynamics.

Beltratti A., Margarita S., Terna P. (1996). *Neural Networks for Economic and Financial Modelling*, ITCP: London.

Gilbert N. and Terna P. (2000). How To Build and Use Agent-Based Models in Social Science. *Mind & Society*, *1*, 57-72.

Gilbert N., Troitzsch K.G. (2005). *Simulation for the Social Scientist*. Buckingham: Open University Press.

Hall R.E. (2005), Employment Fluctuations with Equilibrium Wage Stickiness, *American Economic Review*, *95*, 50-65.

Shimer R. (2005), The Cyclical Behavior of Equilibrium Unemployment and Vacancies. *American Economic Review*, *95*, 25-49.

Stewart I. (1994), The Ultimate in Anty-Particles. *Scientific American*, July 1994, independently reported at www.imsc.res.in/~sitabhra/teaching/cmp03/ian_stewart.html.

Terna P. (2000), Economic Experiments with Swarm: a Neural Network Approach to the Self-Development of Consistency in Agents' Behavior. In F. Luna and B. Stefansson (eds*.), Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming* (pp.73-103). Dordrecht and London: Kluwer Academic.

Tesfatsion L. (2001), Agent-Based Computational Economics: Growing Economies from the Bottom Up. *Artificial Life*, *8*, 55-82.