

Chapter 1

COGNITIVE AGENTS BEHAVING IN A SIMPLE STOCK MARKET STRUCTURE *

Pietro Terna

Università di Torino,

Dipartimento di Scienze economiche e finanziarie G.Prato,

corso Unione Sovietica 218bis, 10134 Torino, Italy

pietro.terna@unito.it

*Apri la mente a quel ch'io ti paleso
e fermalvi entro; ché non fa scienza,
sanza lo ritenere, avere inteso.*

—Dante, *Paradiso*, V

Abstract We introduce here the SUM model—the Surprising (Un)realistic Market model—an agent based framework that allows us to deal with the micro-foundations of a stock market. We avoid any artificially simplified solution about price formation, such as to employ an auctioneer to clear the market; on the contrary, our model produces time series of prices continuously evolving, transaction by transaction.

The core of the model is represented by a computational structure that reproduces closely the behavior of the computerized book of a real

*This research has been supported by a grant from the Italian Consiglio Nazionale delle Ricerche, inside the project “Neural Networks for Economic Analysis: Adaptive Agent Models”.

stock market. The agents send to the book their buy and sell orders, with the related limit prices.

The more interesting characteristic of this model is represented by the introduction of cognitive learning agents able to develop internally their behavioral rules. We have here the possibility of observing the effects of the presence of this kind of “minded” agents both on their own wealth and on the dynamics of the stock market as a whole.

Swarm represents the correct developing framework for our task: It provides us with a multilayer structure (the model and the observer looking at the model) and offers the computational power needed to run experiments with many agents of several types.

1. Introduction

The introductory quotation from Dante (*Paradiso*, V) can be translated as “An open mind is essential to understand novelties” and it is introduced here as a premise to discuss the role of the presence of an “artificial mind” wired into the agents populating our simulation model, but also the role of the mind (or brain) in the actual behavior of economic agents. There is a little bit of arrogance in this purpose, but I hope that minds are open . . . as Dante advocates.

The general idea is that in many cases the complexity of the economic situation is explained by the interaction of simple agents, mainly if behaving in a structured environment (see the random agents and bubbles and crashes in Section 4). So the question is: Are human agents so far from the complexity of the economic system, as ants are from their anthill? We also see here—and this is quite obvious—that only with the presence of “mind” a sort of intelligent behavior, useful to the agents, can emerge. The presence of “minded” agents affects the market, modifying the complexity of its structure: From the external point of view of the social scientist observing actual phenomena this is a puzzle difficult to solve! Have we always to search for the effect of agents’ rationality (“using” their mind) or can we open our minds and accept the Axelrod’s (1997) KISS principle¹?

To discuss these ideas we present, in Subsection 1.1, an overview of the SUM model; in Section 2 the mind/no mind argument; in Section 3 the environment of our artificial experiment; in Section 4 the basic runs with random agents; in Section 5 the runs with other types of agent (market imitating agents in Subsection 5.1, locally imitating agents in Subsection 5.2, “stop loss” agents in Subsection 5.3); in Section 6 we introduce agents that apply estimates of future prices, with Subsection 6.1 describing the forecasting agent that produces those estimates via artificial neural network and Subsection 6.2 reporting the effects of the

forecast application into the market; in Section 7 we have a mixed basic case, with stop loss agents and agents applying Section 6 estimates, as a basis for the successive steps; in Section 8 the structure of the cognitive agents is described, (the Cross Target method in Subsection 8.1 and the operating capabilities of this kind of agents in Subsection 8.2); the behavior of cognitive agents is presented in Section 9 with four cases (Subsections 9.1, 9.2, 9.3 and 9.4); finally, in Section 10, the conclusions.

1.1. An overview of the SUM model

The core of the Surprising (Un)realistic Market model (SUM) is represented by a computational structure that reproduces closely the behavior of the computerized book of a real stock market. The agents send to the book their buy and sell orders, with the related limit prices. The book executes immediately the orders if a counterpart is found in its log; otherwise, it records separately the buy and sell orders, to match them with future orders. The book is cleared at the beginning of each day.

A complex market emerges from the behavior of very simple random agents that: (i) know only the last executed price, (ii) choose randomly the buy or sell side and (iii) fix their limit price by multiplying the previously executed price times a random coefficient. This structure generates increasing and decreasing price sequences with relevant volatility. Also bubbles and crashes appear in this market, generated within the market structure, without the need of exogenous explanations.

In this framework, we then relax the hypothesis (i) for several groups of agents, in order to investigate the consequences of: (1) the presence of subjects using technical trading rules or continuously trained Artificial Neural Networks (ANNs) to forecast the future market prices; (2) the presence of cognitive agents able to learn from their experience and applying external goals in a CT (Cross Target) scheme, according to the methodology proposed in Terna (2000a). In some way, the groups of agents using ANNs as forecasting tools, or within the CT framework, can correspond to the artificially intelligent agents behaving as econometricians proposed by Sargent (1993).

More generally, within this model we can investigate empirical puzzles that are hard to understand using the traditional representative agent structure. Among these puzzles, the time series predictability and the volatility persistence.

The agents that forecast prices with ANNs, employ those generalized functions as econometric tools, fitting them to price series. The CT cognitive agents hold in input their past behavior and series of prices; they must develop the capability of matching their guesses about actions

that have to be done and about the effects of those actions. They have also to develop the strategic ability of buying, selling or doing nothing, in order to cope with the External Objective (in CT jargon) of increasing their wealth.

So we have here both an *econometrics with neural networks* and an *economics with neural networks*.

Swarm represents here the correct developing framework: It provides a multilayer structure and offers the computational power needed to run the experiments with a large number of agents ². Here the multilayer structure contains: (i) the observer layer, that shows the results, and (ii) the model layer, that runs either the time schedule and the environment, with the stock market book and the multiple groups of agents. The use of ANNs and CT scheme in Swarm is based here upon the bp-ct package introduced by Terna (2000a).

2. “Minded” or “no minded” agents in our models

According to Gilbert and Terna (2000):

Ostrom (1988) proposed that there are three different “symbol systems” available to social scientists: The familiar verbal argumentation and mathematics, but also a third way, computer simulation. Computer simulation, or computational modelling, involves representing a model as a computer program. Computer programs can be used to model either quantitative theories or qualitative ones. They are particularly good at modelling processes and although non-linear relationships can generate some methodological problems, there is no difficulty in representing them within a computer program.

The logic of developing models using computer simulation is not very different from the logic used for the more familiar statistical models. In either case, there is some phenomenon that we as researchers want to understand better. This is the “target”. We build a model of the target through a theoretically motivated process of abstraction (this model may be a set of mathematical equations, a statistical equation, such as a regression equation, or a computer program). We then examine the behaviour of the model and compare it with observations of the social world. If the output from the model and the data collected from the social world are sufficiently similar, we use this as evidence in favour of the validity of the model (or use a lack of similarity as evidence for disconfirmation).

The question now is: If our computer simulation model is based upon agents (e.g. built with Swarm, as the models presented here), to what extent must our agents be sophisticated? Should we provide them with a “mind”? The answer ranges from the simplicity principle (Axelrod, 1997) to the use of full BDI (Beliefs, Intentions, Desires) cognitive agents.

A possible classification is:

- A. “no-minded” agents, that behave in an unstructured environment;
- B. learning or “minded” agents, that behave in an unstructured environment;
- C. “no minded” agents, operating in a structured environment;
- D. learning or “minded” agents, operating in a structured environment.

In Terna (2000b) we discuss different models with rigid “no-minded” agents that behave in an unstructured market generating cycles and chaos, or with learning “minded” agents, that assure some stability to an emerging unstructured market.

Here, in Section 4, 5 and 6.1, we present “no minded” agents operating in a structured stock market (case C), with the sophisticated outcome of bubbles and crashes. In Section 8 we have “minded” agents, operating in a structured environment (case D) with effects both on the individual behavior and the related results on the market as a whole.

No generalized results come from this presentation, but many useful suggestions . . .

For a general discussion of agent based models, see Axtell (2000) and the introductory paper (Tsfatsion, 2001) on the Journal of Economic Dynamic and Control, vol.24, special issue 3-4.

3. The artificial experiment environment

The artificial experiments that we introduce here are based on SUM v,0.65; you can download³ the package⁴ and run it with Swarm 2.1.

The parameters are the following.

- In the Observer (the Swarm side of the program related to the observation of the results) we have two basic parameters (and many technical switches about graphics and files, not reported here).
 - *displayFrequency*, the frequency at which the graphic widgets are updated; e.g. if its value is 1000, only one price every 1000 will be reported in the graph; a good choice is to employ the total number of the agents to obtain the last price of each simulated day (each agent does one action per day).
 - *stopAtEpochNumber*, the number of simulated days at which the run will stop; a day is the time required to allow all the agents

to make an action; any action is a tick of a clock that makes `agentNumber` (see below) ticks per day.

In the following experiments we have always the following values: `displayFrequency` = 300 and `stopAtEpochNumber` = 2000.

- In the Model (the Swarm side of the program related to the execution of the agent based simulation) we have several crucial parameters.
 - *agentNumber*: The number of agents present in the model in each day; this value is internally calculated as a sum of the following parameters:
 - *randomAgentNumber*: The number of agents acting in a random way, as described in Section 4;
 - *marketImitatingAgentNumber*: Agents choosing the buy or sell side by imitation, i.e. buying if the market mean price is increasing from day -2 to day -1 and selling if the price is decreasing; see Subsection 5.1;
 - *locallyImitatingAgentNumber*: Agents choosing the buy or sell side on the basis of the majority of the decisions of the last N agents (here $N = 20$, see *localHistoryLength*) as in Subsection 5.2;
 - *stopLossAgentNumber*: Agents that sell or buy to stop loss (we can account for the real agent situation, i.e. if it is “long” or “short” on the market following the parameter *checking-IfShortOrLong* reported below) if the current price is decreasing or increasing, at a rate greater or equal to the *maxLossRate* parameter, when compared to the mean price of the day $t - \text{stopLossInterval}$; see below and Subsection 5.3;
 - *aNNForecastAppAgentNumber*: Agents that follow the forecasts made by a forecasting agent (see Subsection 6.1), buying or selling in a consistent way with the estimates of futures prices; all the uncovered operations are allowed (selling without shares and buying without cash), as described in Subsection 6.2;
 - *bPCTAgentAEO_EP_0_Number* or *EP_1*, *EP_2*, *EP_3*: these are CT agents⁵ of type A^6 , not used here;
 - *bPCTAgentBEO_EP_0_Number* or *EP_1*, *EP_2*, *EP_3*: these are CT agents of type B , introduced in Subsection 8.2. The

cognitive agents used to build our artificial experiment, classified as case D in Section 2 (using a structured environment with “minded” agents).

■ Global parameters.

- *asymmetricBuySellProb*: If one of the two imitative strategies described above is adopted, this is the probability p (here 0.9) of buying or selling as the imitative behavior suggests or of doing the opposite ($1 - p$).

In absence of an imitative behavior, i.e. in the case of random agents, the probability of choosing the buy or the sell side of the market is 0.5.

- *minCorrectingCoefficient*: The min value (here 0.9) of the random multiplying coefficient k used to fix the price of an agent’s buy or sell proposal (‘last price’ times k). This random correction of the buying or selling price to be sent to the book is used by all the types of agents to introduce a little bit of inconsistency or randomness in agents’ behavior; it is also the micro source of the price movement in our simulation experiments⁷. Only the stop loss agents, when adopting the stop loss strategy, use directly the last executed price for their proposal.
- *maxCorrectingCoefficient*: The max value (here 1.1) of the previous k coefficient.
- *asymmetricRange*: The correction added to the previous min and max limits to adopt an asymmetric behavior, if any (this parameter is not used in this paper).
- *agentProbToActBeforeOpening*: The probability of placing an order in the opening phase (here 0.05); so a day starts without an empty book, with a realistic effect (anyway, not crucial for the results); all the types of agents reported above use this parameter.
- *floorP*: The floor price (here 0.3, with 1.0 as first price of the simulation) said below; all the types of agents reported above, but the CT agents of type A or B , use this parameter and the next one.
- *agentProbToActBelowFloorPrice*: The probability (here 0.5) that an agent would buy if the price falls below floorP.
- *maxOrderQuantity*: The max buying or selling quantity in each order placed by an agent (once a day); the actual quantity n is

an integer number chosen randomly in a range from 1 to *maxOrderQuantity*⁸; all the types of agents reported above use this parameter.

- Parameters used by imitating agents (see Subsections 5.1 and 5.2).
 - *meanPriceHistoryLength*: The length (here 200) of the vector of mean prices; this parameter is used also by the forecasting agent of Subsection 6.1.
 - *localHistoryLength*: The length (here 20) of the vector recording agent actions.
- Parameters used by “stop loss” agents (see Subsection 5.3).
 - *maxLossRate*: here 0.10; see above, sub *stopLossAgentNumber*.
 - *stopLossInterval*: here 2; see above, sub *stopLossAgentNumber*.
 - *checkingIfShortOrLong*: here 1 or *yes*; see above, sub *stopLossAgentNumber*.
- Parameters used by the forecasting agent (see Subsection 6.1).
 - *dataWindowLength*: The number of data (here 30), used as inputs of the Artificial Neural Network (ANN); data are return indexes of each daily mean price upon the same value of $-nAheadForecasting$ days before⁹.
 - *nAheadForecasting*: The output of the ANN is the estimate $\frac{p_n}{p_0}$ (here $n = 10$; see the previous note).
 - *forecastingTrainingSetLength*: The number of sequences (here 100) of input data and expected output used to train the ANN; this is a moving training set, which is modified with the internal time of our experiment; note that it must be $meanPriceHistoryLength \geq forecastingTrainingSetLength + nAheadForecasting$ and the artificial neural network training starts only at day $forecastingTrainingSetLength + nAheadForecasting + 1$.
 - *epochNumberInEachForecastingTrainingCycle*: The number of the learning epochs (here 100) in each training phase of the ANN; an epoch contains *forecastingTrainingSetLength* training cycles.
 - *learningProcessEveryNDays*: Every *NDays* (here 10) we repeat the training of the ANN (here with 10,000 training cycles).

- *cleanForecastingANNEveryMgtemN*Days: Every M days (50 in our case; $MgtemN$ stays for M greater than or equal and multiple of N) the learning process restarts from scratch, with initial random parameters (the ANN weights¹⁰) to account for major changes in the structure of our input data.
- Parameters used by agents applying ANN forecasts (see Subsection 6.2).
 - *aNNInactivityRange*: The range (here 0.02 or 2%) of the relative forecasted modification of price in which this kind of agent is inactive (doing nothing) even if the following parameter allows it to act.
 - *aNNForecastAppAgentActDailyProb*: The probability (here 0.1) of acting within each day; this parameter is introduced as a realistic characteristic of the agents that operate on the basis of forecasting and so do not modify continuously their behavior.
- Parameters used by CT agents of type A or B (type A is not used here; for type B , see Subsection 8.2).
 - *epochNumberInEachBPCTTrainingCycle*: The epoch number (100 in our case) in long term learning, applied every day on a moving data set of the last 10 days¹¹.
 - *agentAEO_EPDelta*: A parameter used to measure the effect of External Objectives (EO) for CT agents of type A (here 0.1, but never used, because we have no type A agents in this set of experiments).
 - *agentBEO_EPDelta*: A parameter used to measure the effect of EO for CT agents of type B (here 10 as default value, but modified in the experiments reported Section 9).

4. Basic runs: random agents only

In this first application of our experimental environment, we are in the case of “no minded” agents, operating in a structured environment (case C of Section 2).

In the runs reported in Figures 1.1 and 1.2, we adopt the hypotheses (i), (ii) and (iii) introduced above (Subsection 1.1). Our random agents¹² know only the last executed price, choose randomly the buy or sell side and fix their limit price by multiplying the previously executed price times a random coefficient (see Section 3 about the parameters used).

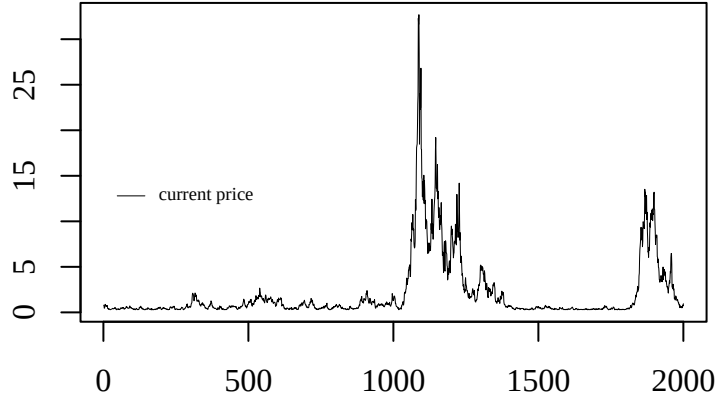


Figure 1.1. Random agents: The price sequence of the first basic simulation, with $randomAgentNumber = 100$ and $maxOrderQuantity = 6$.

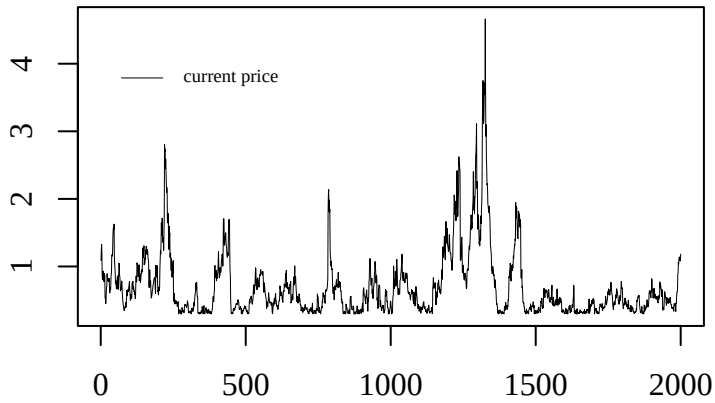


Figure 1.2. Random agents: The price sequence of the second basic simulation, with $randomAgentNumber = 300$ and $maxOrderQuantity = 3$.

We introduce also the rule of buying with a fixed probability (here $p = 0.5$) if the price falls below a specific floor.

Bubbles and crashes appearing in this framework are a direct consequence of the structure (the electronic book) of the market.

The emergence of this kind of anomalies in a model of type C is particularly interesting, because it shows the importance of rules (here the technical structure of the market) in influencing the behavior and, mainly, the interaction among agents. So simple agents produce complex results.

Prices are generated transaction by transaction, one per tick (if the agent required to act in a tick does not act, the price is kept unchanged).

As we have seen introducing the *displayFrequency* parameter above in Section 3, we display on the graph only one price every day (the last one).

We are looking here for a basic situation, choosing a number of random agents, and a max quantity of orders for each agent in each day, in order to generate a market with limited bubbles and crashes.

Few agents, with more than one order as max order quantity for each agent and day, can generate a world with huge bubbles and crashes, interesting *per se*, but not easy to be used to investigate the role of different types of agents in the models.

This is the case of Figure 1.1 where we have a run with 100 random agents that can place up to 6 orders per day.

We can also investigate the predictability of the market. Considering the log of the “book” operating in the experiment reported in Figure 1.1, i.e. the number of unmatched buy orders (b) and the number of unmatched sell orders (s) at the end of each day (as a proxy of the same measures at the exact tick where a bubble or a crash begins), we can simply estimate the R^2 coefficient in $pi = c_0 + c_1b + c_2s$, where pi is a daily price index calculated upon last prices as $\frac{p_0}{p_{-1}}$, and we obtain $R^2 = 0.35$. The presence of (rare) sequences of actions on the same side of the market is the explanation of huge movements in prices. If we have a lot of random agents with a limited quantity of orders in each tick, the white noise effect prevails and we have a regular market.

In this first basic experiment we consider also the estimates of the prices 10 days ahead (this is a parametric choice) made by a forecasting agent that is looking at the market, but not acting in it. The forecasting agent, based upon an artificial neural network, is introduced in Subsection 6.1.

A simple test about the estimate success is the count of the quota of correct forecasting signs (here 52%).

Besides the interest on bubbles and crashes, our goal here is to obtain a relatively stable market. So we increase the number of random agents, lowering the max quantity of orders in each tick to 3, as in the run reported in Figure 1.2. (The quota of correct forecasting signs is here 48%).

In Figure 1.3 we can also verify the effect of the agents’ behavior on their wealth, evaluated as liquidity plus shares at the daily mean price. Initial endowments are zero liquidity and shares; agents can anyway operate doing uncovered operations without limits. We show here our random agents min, max and mean wealth. Being the random agents (r. in the Figure) the unique category of agents in the market (the totality of the agents, in this case) the mean wealth is zero by definition (the

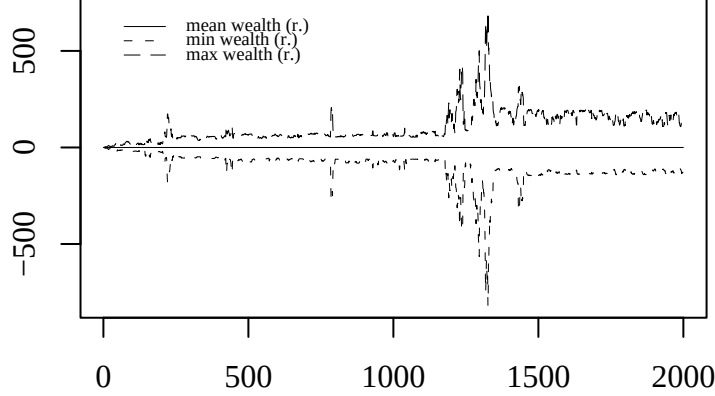


Figure 1.3. Random agents: The wealth series of the second basic simulation, with $randomAgentNumber = 300$ (r.) and $maxOrderQuantity = 3$.

exchanges, via the book, are always among agents); min and max shows the possibility of huge gains or losses adopting a random (obviously, on the border lines we have different agents as the time evolves).

5. Introducing other types of agents

To evaluate the effects of the presence of phenomena such as imitation and “stop loss” strategies, we introduce now new categories of agents. Our goal is both to explore the effects of the action of these types of artificial subjects and to build a sound framework for the introduction of agents able to apply market forecasts (Section 6) and for the experiments with cognitive or “minded” agents (Section 9).

5.1. Market imitating agents

We have here, within the basic structure of Section 4, the presence of market imitating agents¹³.

Referring to Subsection 1.1 we relax now the hypotheses (i)—to know only the last executed price—and (ii)—to choose randomly, in a balanced way, the buy or sell side—for a small quota of the agents, in order to investigate the consequences of the presence of subjects imitating the market (general imitation). Their choice—to buy or to sell—will be unbalanced, according to the probability $p = asymmetricBuySellProb$ introduced in Section 3.

The parameter p measures the probability that an agent would choose the buy or sell side as an imitative act of the market as a whole, buying with probability p if the mean price is increasing from day $t - 2$ to day

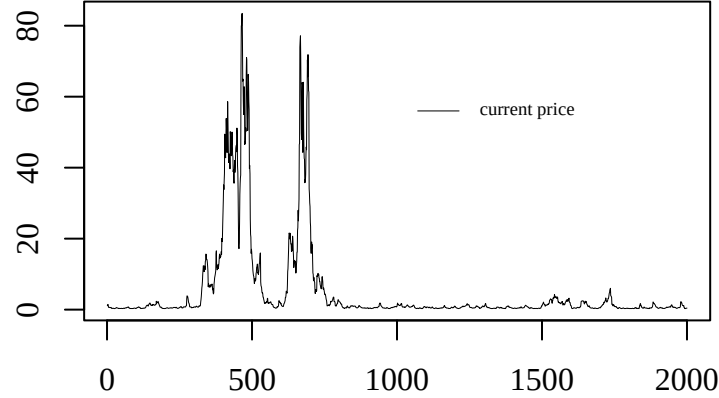


Figure 1.4. Market imitating agents: The price sequence of the simulation with *randomAgentNumber* = 285, *marketImitatingAgentNumber* = 15 and *maxOrderQuantity* = 3.

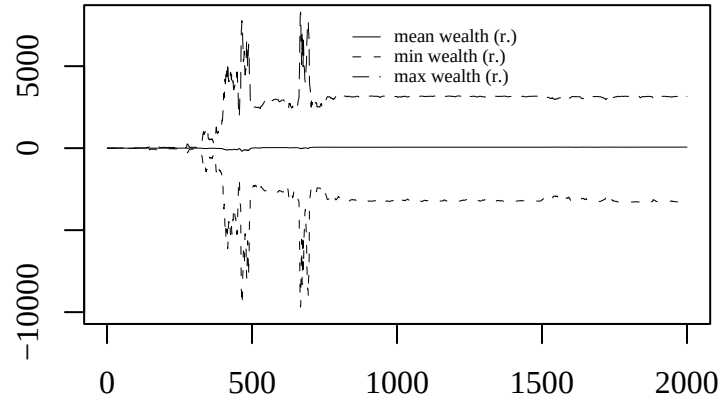


Figure 1.5. Market imitating agents: The wealth series simulation with *randomAgentNumber* = 285 (r.), *marketImitatingAgentNumber* = 15 and *maxOrderQuantity* = 3.

$t - 1$ and selling with the same probability if the price is decreasing; this is an imitation effect, but also a proxy of the behavior of “chartists” (Lo *et al.*, 2000). The presence of this kind of agents—also in a small quota—deeply increases the appearance of bubbles and crashes.

In Figure 1.4 we report the price sequence of a run of the simulation with market imitating agents, with two enormous bubbles. The quota of correct forecasting signs is here 53%.

In Figures 1.5 (random agents) and 1.6 (market imitating agents) we have the effect of the agents’ behavior on their wealth¹⁴. Remembering

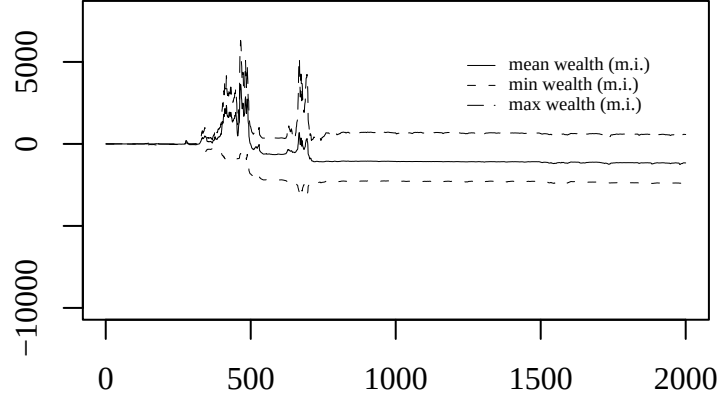


Figure 1.6. Market imitating agents: The wealth series simulation with *randomAgentNumber* = 285, *marketImitatingAgentNumber* = 15 (m.i.) and *maxOrderQuantity* = 3.

that on the border lines we have different agents as the time evolves, we can see the enormous consequences of these types of behavior in agents endowments and the special effect of imitation, in limiting losses.

5.2. Locally imitating agents

We have here, always within the basic structure of Section 4, the presence of locally imitating agents¹⁵.

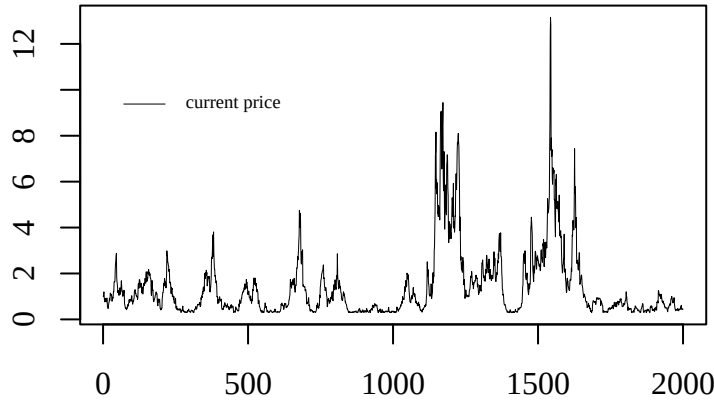


Figure 1.7. Locally imitating agents: The price sequence of the simulation with *randomAgentNumber* = 285, *locallyImitatingAgentNumber* = 15 and *maxOrderQuantity* = 3.

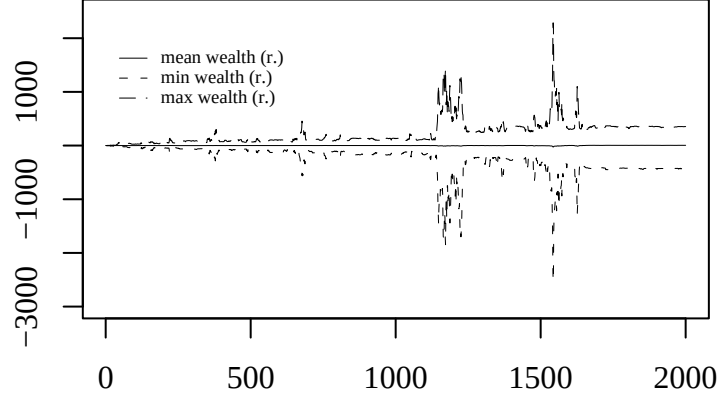


Figure 1.8. Locally imitating agents: The wealth series simulation with *randomAgentNumber* = 285 (r.), *locallyImitatingAgentNumber* = 15 and *maxOrderQuantity* = 3.

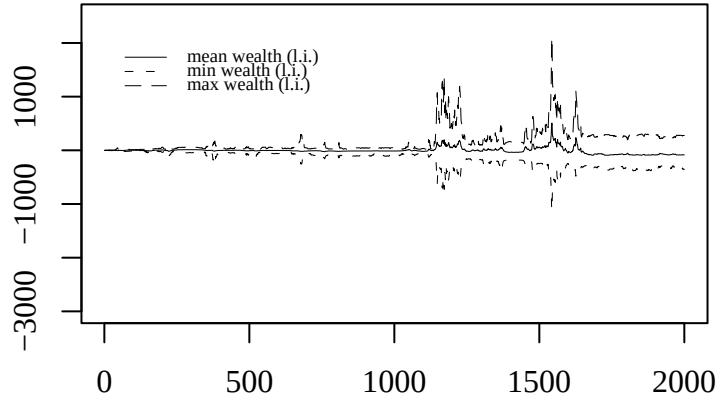


Figure 1.9. Locally imitating agents: The wealth series simulation with *randomAgentNumber* = 285, *locallyImitatingAgentNumber* = 15 (l.i.) and *maxOrderQuantity* = 3.

Referring to Subsection 1.1 we again relax the hypotheses (i)—to know only the last executed price—and (ii)—to choose randomly, in a balanced way, the buy or sell side—for a small quota of the agents, in order to investigate the consequences of the presence of subjects locally imitating other agents' behavior (local imitation). Their choice to buy or sell will be unbalanced, according to the probability $p = asymmetricBuySellProb$ introduced in Section 3.

p is the probability of buying or selling following the majority of the last $n = localHistoryLength$ (here $n = 20$) agents' decisions. The “local

history” sequence—which is a moving window continuously updated—does not consider idle agents (this is especially important in the opening phase, where only a little quota of the agents is active).

In Figure 1.7 we report the price sequence of a run of the simulation with locally imitating agents, with two bubbles, but not of the dimension of those of the market imitation case. This kind of market structure has a low predictability: The quota of correct forecasting signs is here 47%.

In Figures 1.8 (random agents) and 1.9 (locally imitating agents) we have the effect of the agents’ behavior on their wealth¹⁶. Remembering that on the border lines we have different agents as the time evolves, we can see the relevant effect of these types of behavior on agents endowments and again the special effect of imitation, in limiting losses.

5.3. Stop loss behavior

We have here, again within the basic structure of Section 4, the presence of “stop loss” agents¹⁷.

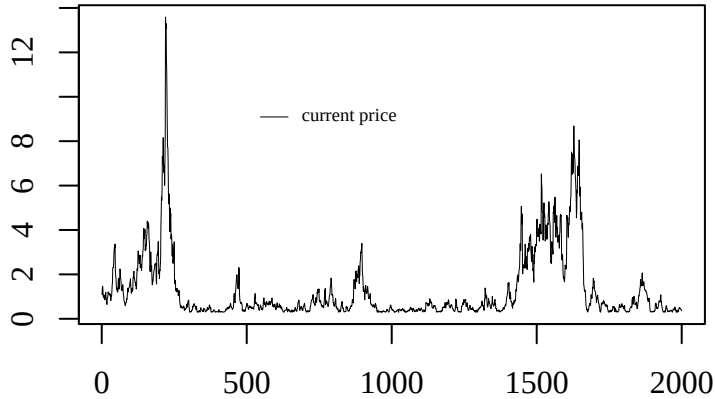


Figure 1.10. “Stop loss” agents: The price sequence of the simulation with *randomAgentNumber* = 285, *stopLossAgentNumber* = 15 and *maxOrderQuantity* = 3.

We have here agents that sell or buy to stop a loss¹⁸ if the current price, compared to the mean price of the day $t - \text{stopLossInterval}$ (see Section 3), is decreasing or increasing at a rate greater or equal to the *maxLossRate* parameter.

In Figure 1.10 we report a run with stop loss agents (if loss > 10% in two days, agents apply stop loss). The effect of stop loss is to produce reasonable bubbles. The quota of correct forecasting signs is here 54%, so this kind of market appears to have an interesting degree of predictability.

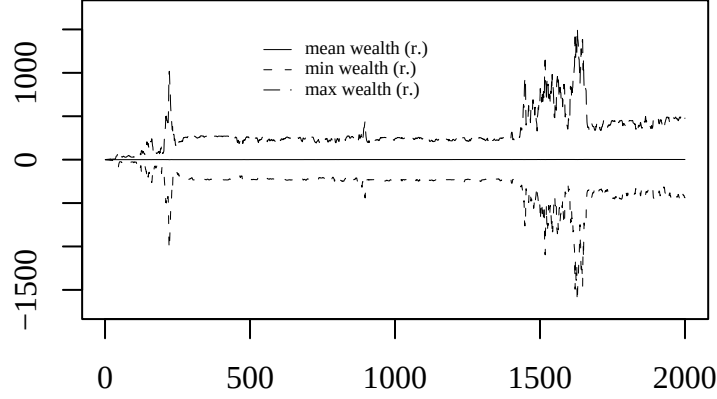


Figure 1.11. “Stop loss” agents: The wealth series simulation with *randomAgentNumber* = 285 (r.), *stopLossAgentNumber* = 15 and *maxOrderQuantity* = 3.

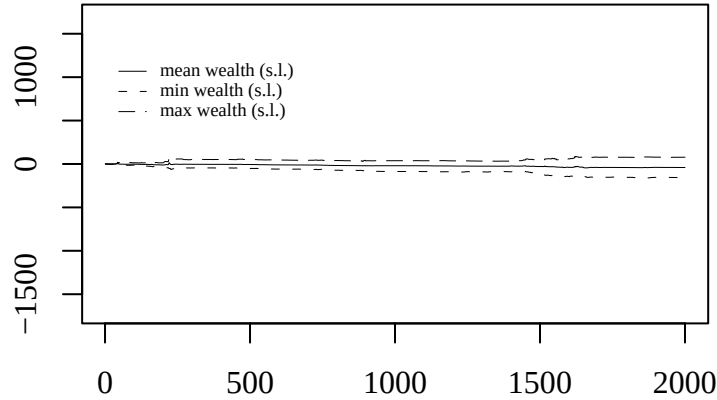


Figure 1.12. “Stop loss” agents: The wealth series simulation with *randomAgentNumber* = 285, *stopLossAgentNumber* = 15 (s.l.) and *maxOrderQuantity* = 3.

In Figures 1.11 and 1.12 we have the consequences of the two types of behavior on the wealth of the agents, with—as always—huge gains or losses for the agents (r.) operating by chance, but also with a defensive situation of stop loss agents (s.l.).

6. Agents applying ANN estimates of future prices

We introduce now the “applying ANN forecast agents”, or more simply “applying forecast agents” (a.f.a.), that follow the estimates of the forecasting agent described below.

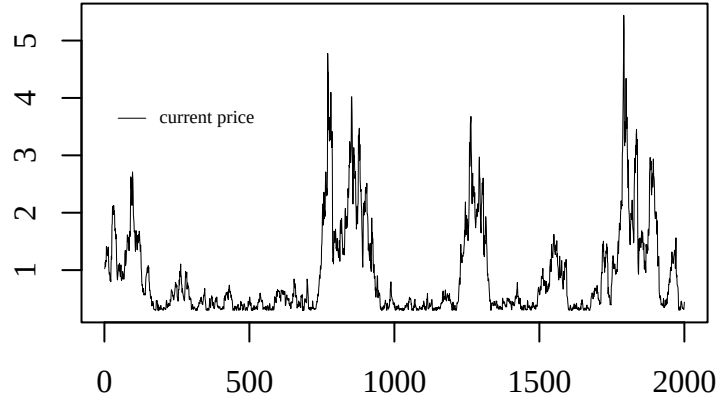


Figure 1.13. Applying forecast agents: The price sequence of the simulation with $randomAgentNumber = 285$, $aNNForecastAppAgentNumber = 15$ and $maxOrderQuantity = 3$.

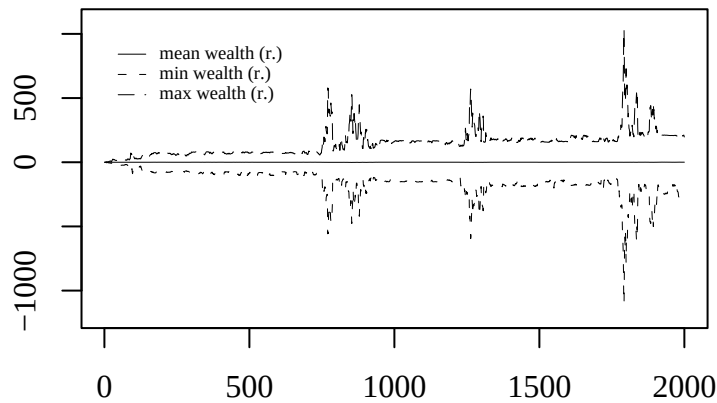


Figure 1.14. Applying forecast agents: The wealth series simulation with $randomAgentNumber = 285$ (r.), $aNNForecastAppAgentNumber = 15$ and $maxOrderQuantity = 3$.

We are here near to the field of self-fulfilling prophecies. Our forecaster uses N ANN to produce its estimates and the quality of the estimates is related to the degree of predictability of the market; but as we can see below, if a lot of agents apply those estimates to behave, we have a strong reinforcing feedback to the predictability of the market¹⁹.

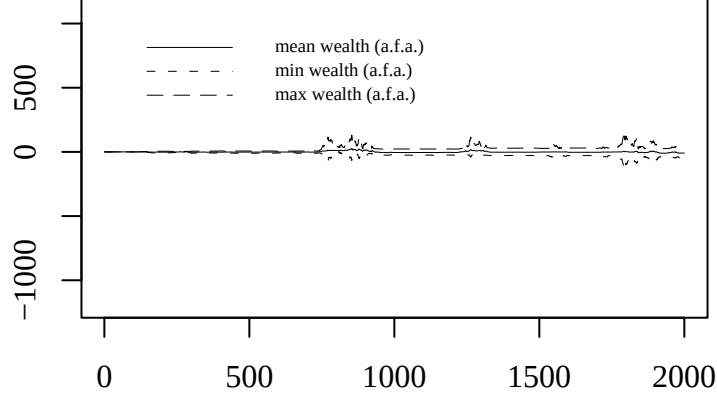


Figure 1.15. Applying forecast agents: The wealth series simulation with $randomAgentNumber = 285$, $aNNForecastAppAgentNumber = 15$ (a.f.a.) and $maxOrderQuantity = 3$.

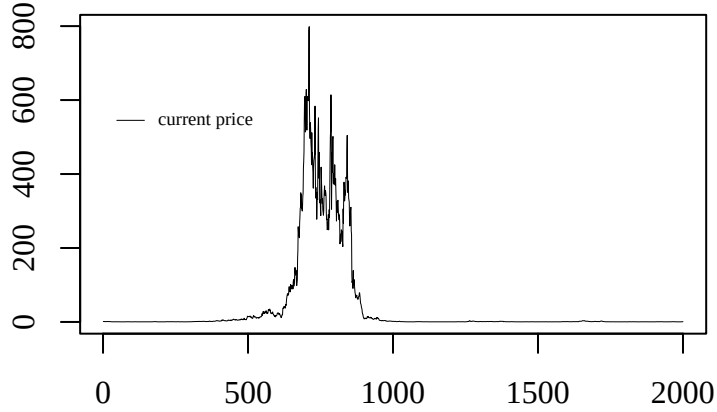


Figure 1.16. Applying forecast agents: The price sequence of the simulation with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 30$ and $maxOrderQuantity = 3$.

6.1. The forecasting agent

We build a new agent, not behaving in the market, that produces in each day a forecast of the stock market price $nAheadForecasting$ (here 10) days ahead.

Referring to parameters of Section 3, the forecaster uses an Artificial Neural Network (ANN) with $dataWindowLength$ (here 30) data as inputs. Data are return indexes of each daily mean price upon the same value of $-nAheadForecasting$ days before. The output of the ANN is the estimate $\frac{p_n}{p_0}$ (here $n = 10$). $forecastingTrainingSetLength$ is the

number of sequences (here 100) of input data and expected output used to train the ANN; this is a moving training set, which is modified with the internal time of our experiment. *epochNumberInEachForecasting-TrainingCycle* is the number of learning epochs (here 100) in each training phase of the ANN; an epoch contains *forecastingTrainingSetLength* training cycles.

Every *learningProcessEveryNDays* days (here 10) we repeat the training of the ANN (here with 10,000 training cycles). Finally, every *clean-ForecastingANNEveryMgtemNDays* days (here 50) the learning process restarts from scratch, with initial random parameters (the ANN weights) to account for major changes in the structure of our input data.

The number of hidden nodes is internally determined as one half the input nodes number; this value can be externally established adding a line with 'hiddenNodeNumber x' into the file `forecasting.setup`.

For a concise introduction to ANN technique, considering multilayer feedforward ANNs trained with the backpropagation technique, see Section 4 in Terna (2000a).

6.2. The effects of the forecast application

We have here, always within the basic structure of Section 4, the presence of agents²⁰ behaving consistently with the estimates of the forecasting agent, acting with uncovered operations in a mechanical way (buying or selling whenever the prediction is respectively greater or less than the current price).

More specifically, agents do not operate if the forecasted price index is $> 1 + aNNInactivityRange$ or $< 1 - aNNInactivityRange$. Agents also avoid nonsense situations, not acting if the current price is already over the forecasted one.

In Figure 1.13 we report the price sequence of a run of the simulation with a limited number (15) of applying forecast agents, with a reasonable movement in prices and small bubbles. The quota of correct forecasting signs is here 52%.

In Figures 1.14 (random agents) and 1.15 (applying forecast agents) we have the effect of the agents' behavior on their wealth. Remembering that on the border lines we have different agents as the time evolves, we can see the consequences of these types of behavior in agents endowments and the special effect of forecast application, with a limited advantage of gains against losses, while in random behavior losses seem to prevail.

In Figure 1.16 we report the price sequence of a run of the simulation obtained doubling to 30 the number of applying forecast agents, with

an “astronomical” bubble. The quota of correct forecasting signs is here 58%, as we are close to a situation of self-fulfilling prophecies.

7. A mixed basic case, with stop loss agents and agents applying ANN estimates

This is the key case used in Section 8 to introduce the simulations with the cognitive CT agents of type B.

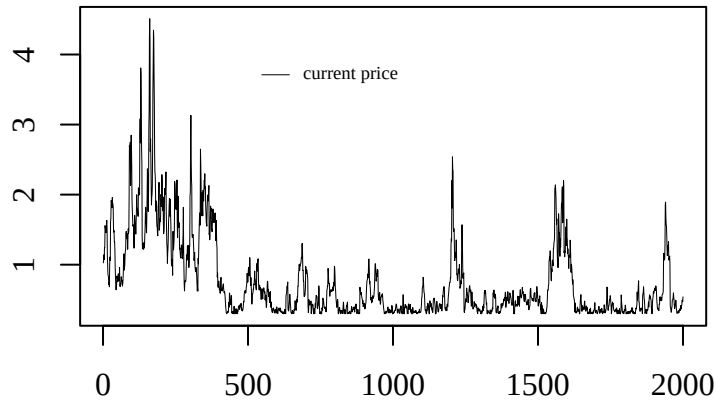


Figure 1.17. Mixed basic case: The price sequence of the simulation with *randomAgentNumber* = 275, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3.

Our goal here is to produce a basic structure with a good level of predictability, without unstable or explosive situations. So we join a limited number of applying forecast agents (Subsection 6.2) and of “stop loss” agents (Subsection 5.3) and obtain a market such as that reported in Figure 1.17, with a quota of correct forecasting signs of 55%. Also the effects on the agents’ wealth are bounded in reasonable limits (± 300 for the random agents).

8. The structure of the CT cognitive agents

We finally introduce “minded” agents in a structured market (case D of the classification reported in Section 2).

We use here the CT scheme (Terna, 2000a), which will be briefly explained in Subsection 8.1. This kind of agent uses forecastings as proxies of the true model of the economy; in some way, these are agents applying rational expectations in the Sargent (1993) sense, reported in Subsection 1.1.

8.1. The CT method

To develop our 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.

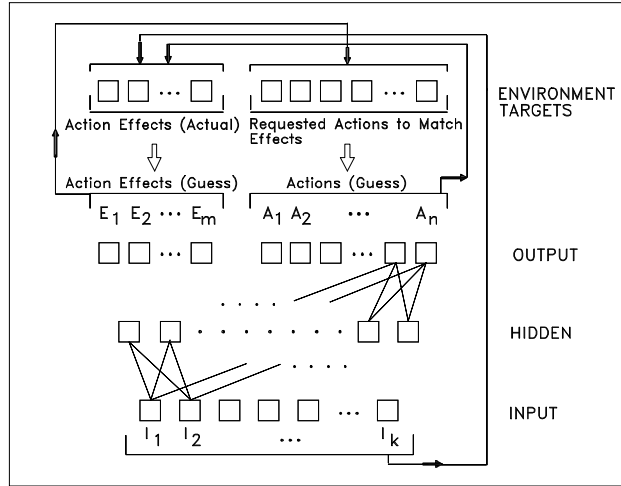


Figure 1.18. The Cross-Target (CT) scheme.

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

Following the GH, the main characteristic of these 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. So we specify two types of outputs of the ANN

and, identically, of the agent: (i) actions to be performed and (ii) guesses about the effects of those actions.

We choose the neural networks approach to develop CT method, mostly as a consequence of the intrinsic adaptive capabilities of neural functions.

Figure 1.18 describes an agent learning and behaving in a CT scheme. The agent has to produce guesses about its own actions and related effects, on the basis of an information set (the input elements are $I_1, \dots, I_k$). 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. Technical explanations of CT method (Beltratti *et al.*, 1996) are reported also in Section 5 in Terna (2000a).

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.

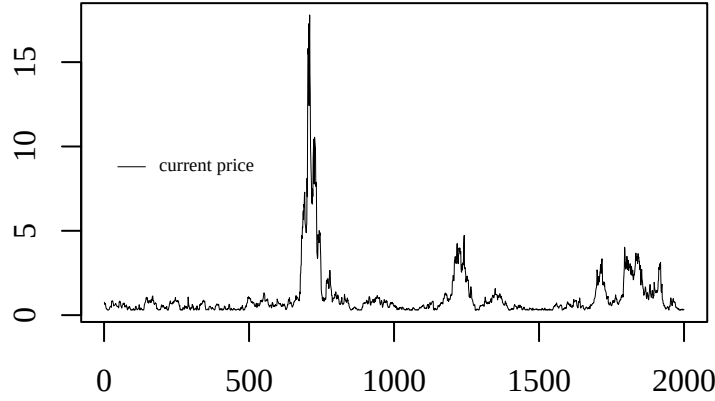


Figure 1.19. CT agents: The price sequence of the simulation with *randomAgentNumber* = 270, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 5.

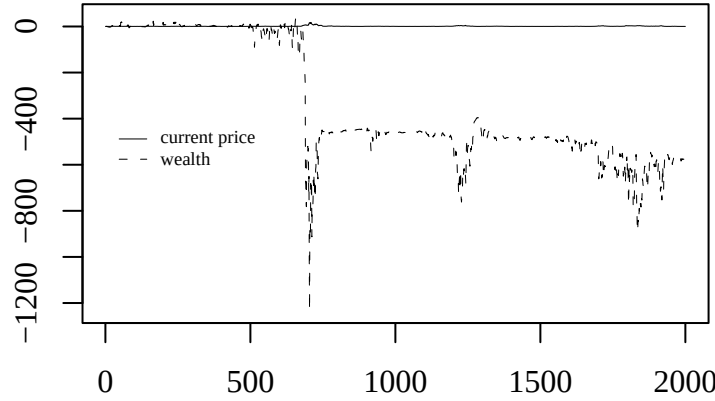


Figure 1.20. CT agents: Price and agent wealth series with *randomAgentNumber* = 270, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 5.

Now we may introduce some technical explanations about CTs, with the aid of the general scheme of Figure 1.18, observing that (i) the inputs of the model are mainly data coming from the environment or

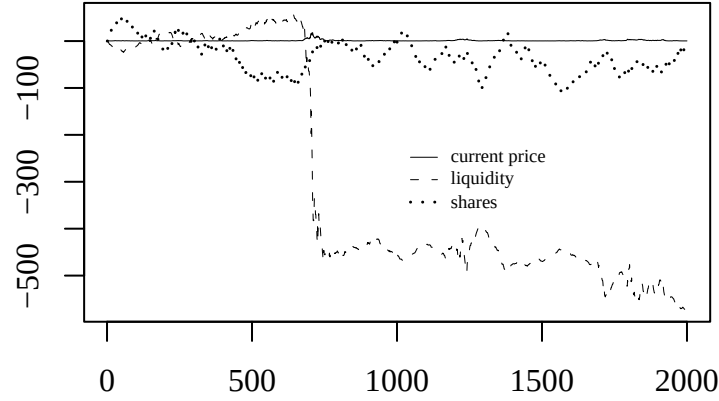


Figure 1.21. CT agents: Price, liquidity and shares series with *randomAgentNumber* = 270, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 5.

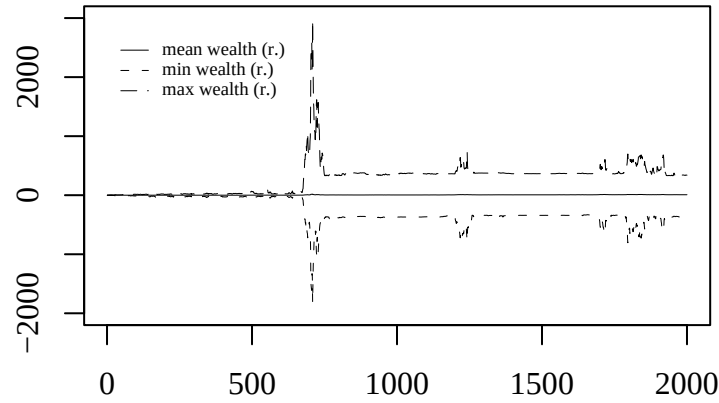


Figure 1.22. CT agents: The wealth series with *randomAgentNumber* = 270 (r.), *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 5.

from other agents' behavior, (ii) they can be dependent or independent from the previous actions of the simulated artificial subject, (iii) targets are known only when actions take place.

The CT algorithm is a learning and acting one: action is necessary to produce the information by which we can construct targets to train the ANN that simulates the subject.

Learning and acting take place in four steps each "day"; a day is the sum of the four steps required to perform a full cycle of estimation of

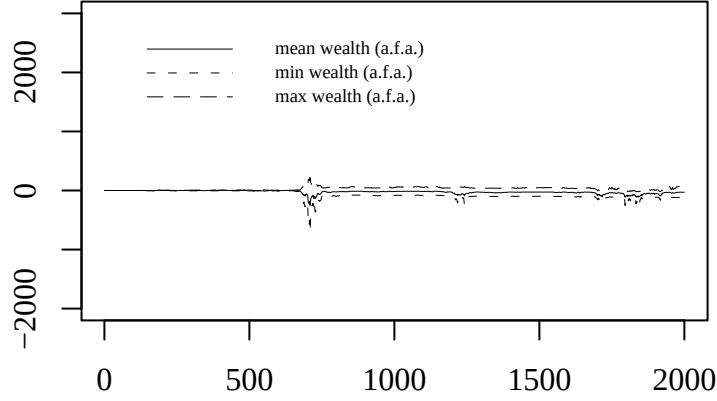


Figure 1.23. CT agents: The wealth series with $randomAgentNumber = 270$, $aNN-ForecastAppAgentNumber = 15$ (a.f.a), $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ and $EO = 2$, $d = 5$.

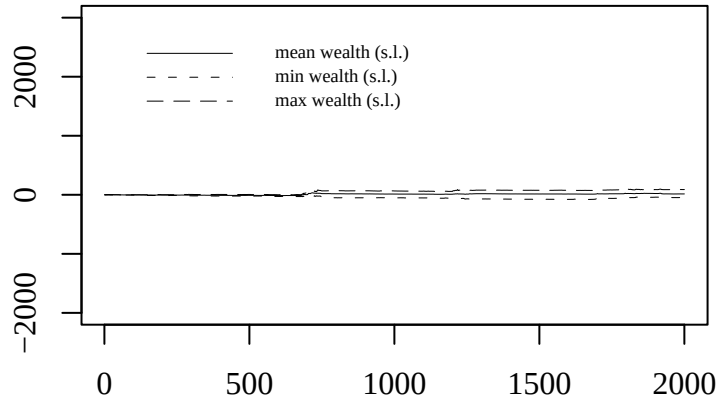


Figure 1.24. CT agents: The wealth series with $randomAgentNumber = 270$, $aNN-ForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ (s.l.) and $maxOrderQuantity = 3$, $BPCTB = 5$ and $EO = 2$, $d = 5$.

outputs and of backpropagation of errors, correcting the neural network weights. Initial weights are randomized in a given range.

Looking at Figure 1.18, the four steps can be introduced in the following sequence.

- 1 Outputs of the ANN: the actions to be accomplished, reported in the right side of Figure 1.18, and the effects of these actions, reported in the left side of the same figure, are guessed following inputs and network weights.

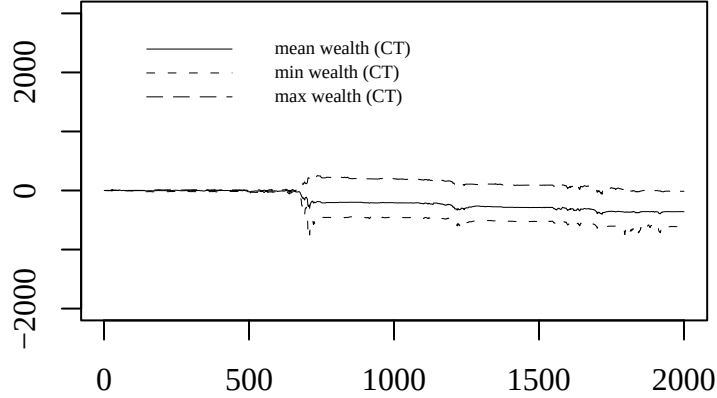


Figure 1.25. CT agents: The wealth series with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ (CT) and $EO = 2$, $d = 5$.

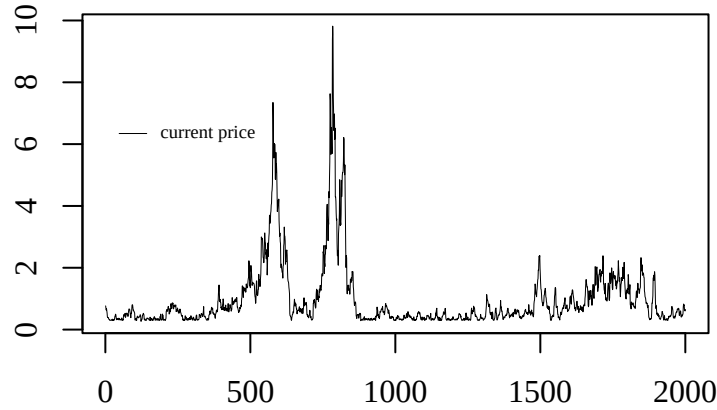


Figure 1.26. CT agents: The price sequence of the simulation with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ and $EO = 2$, $d = 10$.

- 2 Targets for the left side of the network: the targets for the effects supposed to arise from actions, as guessed in the left side of the output layer in Figure 1.18, are figured out by the independently guessed actions. In this way, guesses about effects become more close to the true consequences of actual actions.
- 3 Targets for the right side of the network: the differences measured in step (2) among targets and ANN outputs on the effect side can be inversely interpreted as starting points for action modifications, to match the guessed effects. So they are used to build the targets

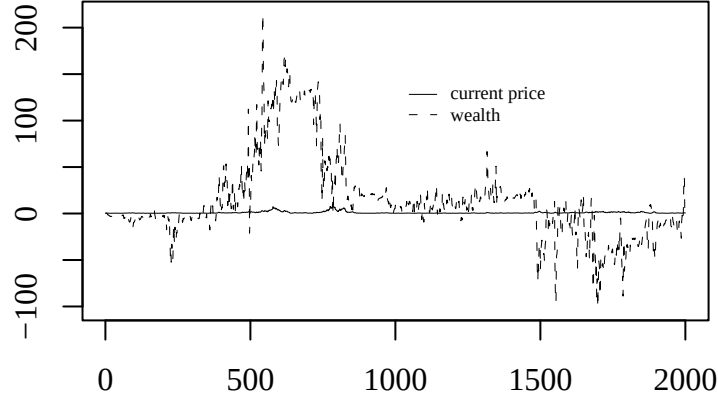


Figure 1.27. CT agents: Price and agent wealth series with *randomAgentNumber* = 270, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 10.

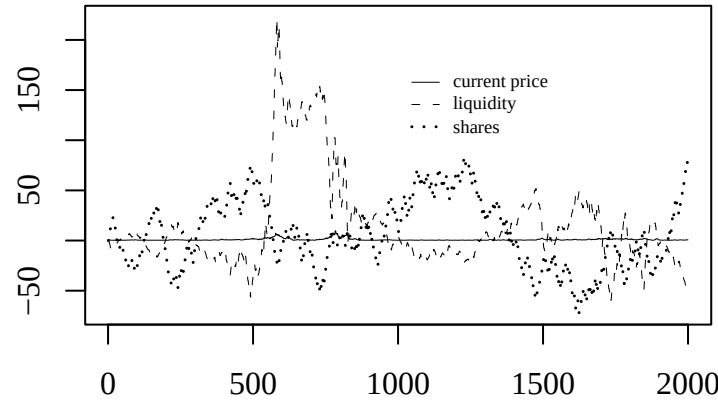


Figure 1.28. CT agents: Price, liquidity and shares series with *randomAgentNumber* = 270, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 10.

for the mechanism that guesses the actions. Being the inverses of the formulas shown below often undefined, corrections are shared randomly among all the targets to be constructed; besides, when several corrections concern a target, only the one with the largest module is chosen. In this way, we would like to imitate the actual behavior of a subject requested of obeying to several independent and inconsistent commands: probably the most imperative, here the largest value, will be followed.

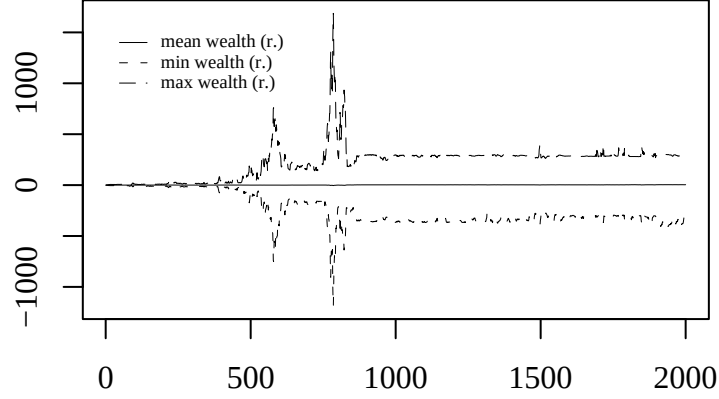


Figure 1.29. CT agents: The wealth series with $randomAgentNumber = 270$ (r.), $aNNForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ and $EO = 2$, $d = 10$.

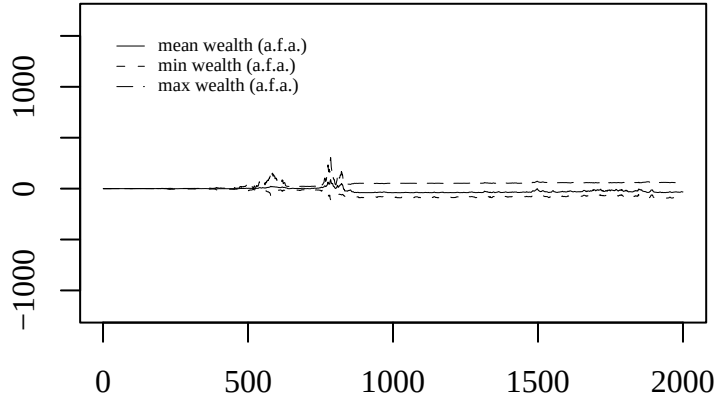


Figure 1.30. CT agents: The wealth series with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 15$ (a.f.a.), $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ and $EO = 2$, $d = 10$.

- 4 Backpropagation: learning takes place, correcting weights in order to obtain guessed effects closer to the consequences of guessed actions, and guessed actions more consistent with guessed effects. Thus, we have two learning processes, both based upon the guesses of the elements of the opposite side of the network.

This double sided process of adaptation, with interaction among the agents and long term learning introduced above, ensures the emergence of non trivial self-developed behavior, from the point of view of time paths of the values generated by the outcomes of the agents.

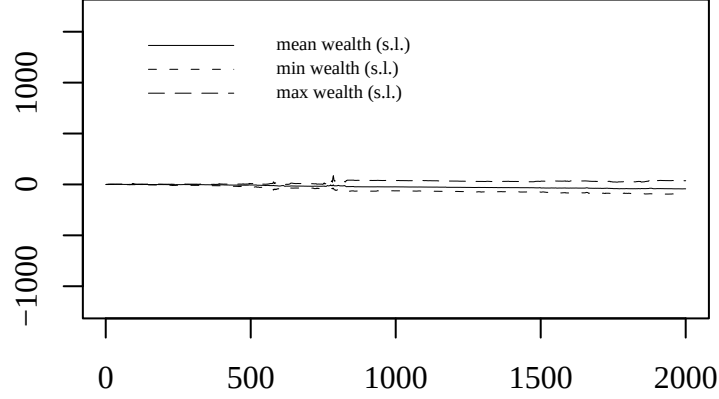


Figure 1.31. CT agents: The wealth series with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ (s.l.) and $maxOrderQuantity = 3$, $BPCTB = 5$ and $EO = 2$, $d = 10$.

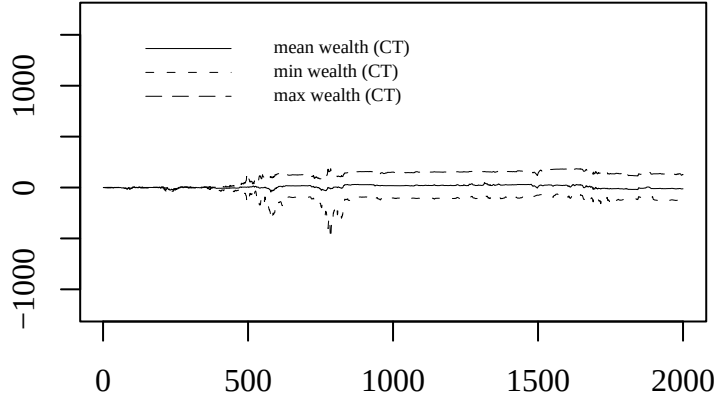


Figure 1.32. CT agents: The wealth series with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ (CT) and $EO = 2$, $d = 10$.

8.2. The [BP]CT B agent

We introduce now the structure and the behavior of the cognitive agents constituting the core of our agent based simulation. First of all, the name: CT agents or Cross-Target agents of type B, being the SUM package (see Section 3 for the download) able to deal both with simple CT type A agents not used here and with sophisticated type B ones; we use also, mainly internally to the program, the BPCT name (backpropagation CT) referring to the technique used to train the ANN that wires the agents. “bp-ct” is also the name of the general package

(Terna, 2000a) about the CT technique; this package is now included in SUM²¹, but it is also useful as a stand alone tool²².

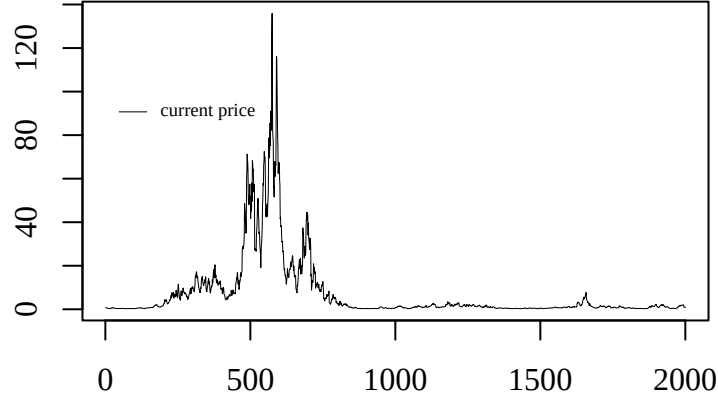


Figure 1.33. CT agents: The price sequence of the simulation with *randomAgentNumber* = 270, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 20.

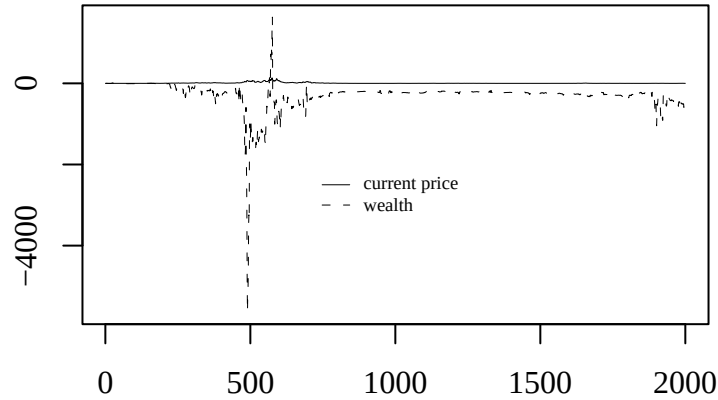


Figure 1.34. CT agents: Price and agent wealth series with *randomAgentNumber* = 270, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 20.

The type B agent has eight nodes in input: the mean price from day $t - 5$ to day $t - 1$ (we are in t); the agent's liquidity and share values at the end of day $t - 1$ and the estimated price index of day $t + n - 1$, with respect to $t - 1$ value, where $n = nAheadForecasting$ (see Section 3). The number of hidden nodes is six.

Following the CT scheme, we have four outputs on the *effect* side and one output on the *action* side. Effects (evaluations at the end of day

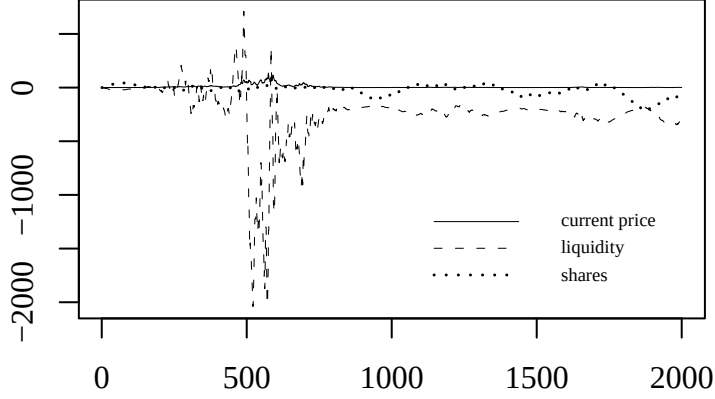


Figure 1.35. CT agents: Price, liquidity and shares series with *randomAgentNumber* = 270, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 20.

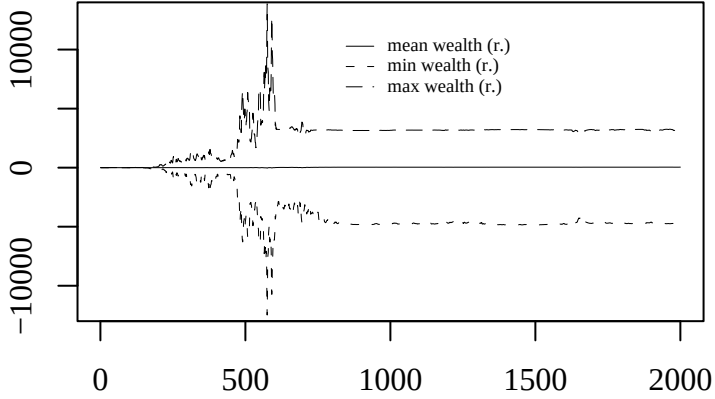


Figure 1.36. CT agents: The wealth series with *randomAgentNumber* = 270 (r.), *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 20.

t): liquidity; share quantity; wealth, using the last daily price; wealth, using the forecasted price. Actions: the buy or sell decision.

The buy or sell (*bs*) decision is represented (in external metrics) as a real number in the range $\pm \text{maxOrderQuantity}$; if *bs* > 0, the agent is on the buy side; if *bs* < 0, on the sell side; if *bs* = 0 (an absolutely rare case), the agent does nothing; the quantity to be bought or sold (or the number of unitary orders) is determined rounding $|bs|$ to the upper unit.

The EOs are here related to the two wealth effects and are determined²³ adding directly to the wealth internal guesses (those arising from

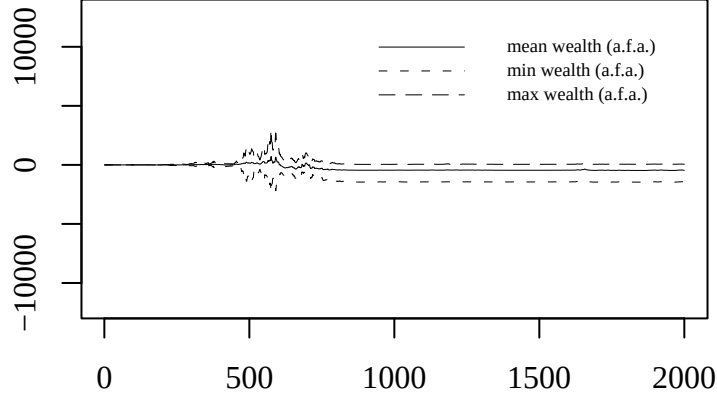


Figure 1.37. CT agents: The wealth series with *randomAgentNumber* = 270, *aNN-ForecastAppAgentNumber* = 15 (a.f.a.), *stopLossAgentNumber* = 10 and *maxOrder-Quantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 20.

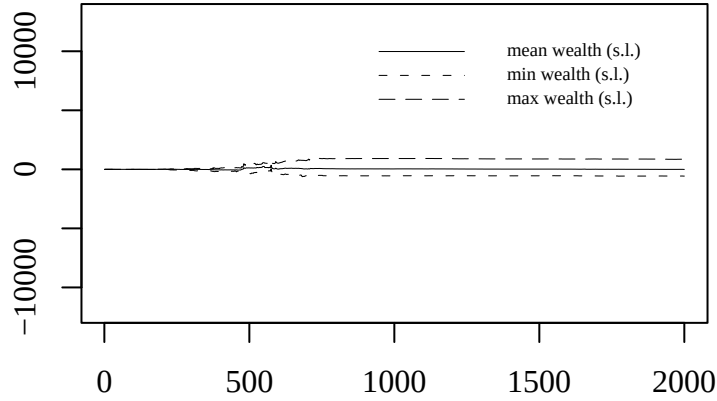


Figure 1.38. CT agents: The wealth series with *randomAgentNumber* = 270, *aNN-ForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 (s.l.) and *maxOrder-Quantity* = 3, *BPCTB* = 5 and *EO* = 2, *d* = 20.

the development of the IC of the agent) a fixed amount d (measured in wealth metrics). We can have: $EO = 0$ or no EOs; $EO = 1$, i.e. to increase the wealth evaluation based on the closing price; $EO = 2$, i.e. to increase the wealth evaluation based on the forecasted price; $EO = 3$ (not used here) as the attempt of increasing both the evaluations.

9. The behavior of the CT cognitive agents

In the following Subsections we introduce the various experiments with the cognitive CT agents²⁴, modifying the d value to have light

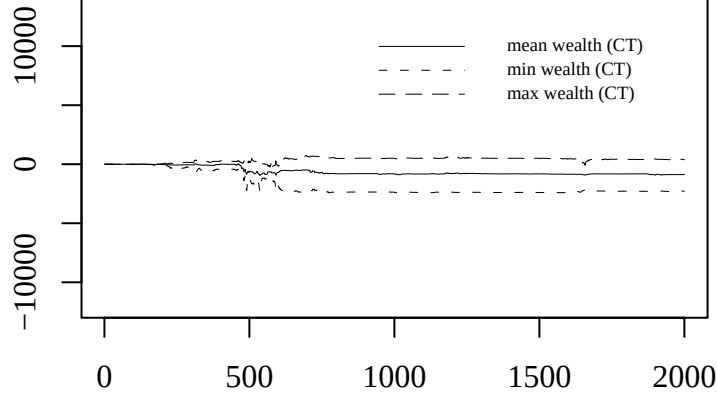


Figure 1.39. CT agents: The wealth series with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ (CT) and $EO = 2$, $d = 20$.

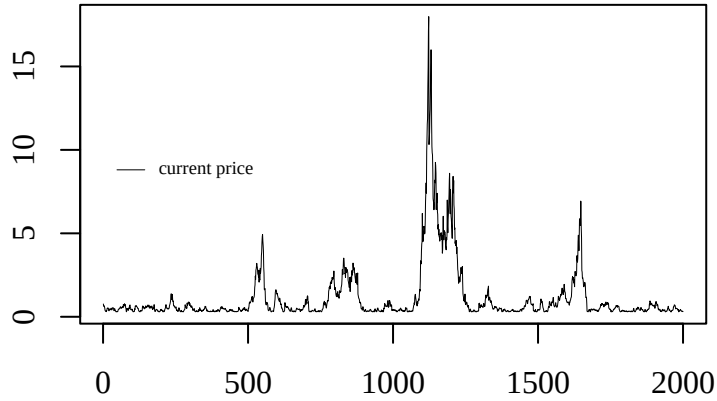


Figure 1.40. CT agents: The price sequence of the simulation with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ and $EO = 1$, $d = 10$.

or heavy $EO = 2$ corrections and using also, for comparative reasons, $EO = 1$.

9.1. Case 1: $EO = 2$; $d = 5$

The first case, with $EO = 2$ (to increase the wealth evaluation based on the forecasted price) and $d = 5$ shows (Figure 1.19) a reasonable price sequence, with one relevant bubble. The quota of correct forecasting signs is here 54%, confirming the predictability of the mixed basic case introduced in Section 7.

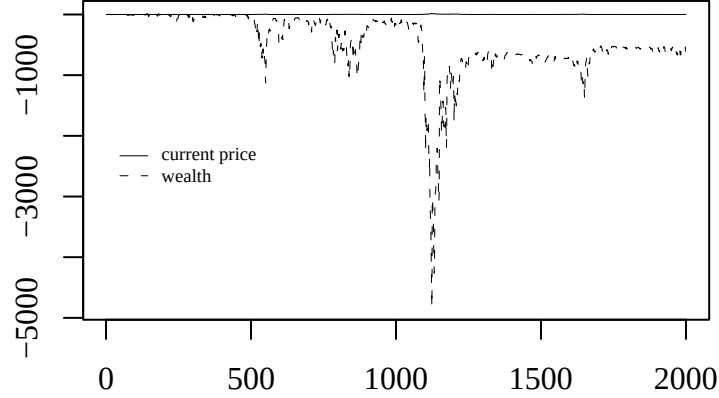


Figure 1.41. CT agents: Price and agent wealth series with *randomAgentNumber* = 270, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 1, *d* = 10.

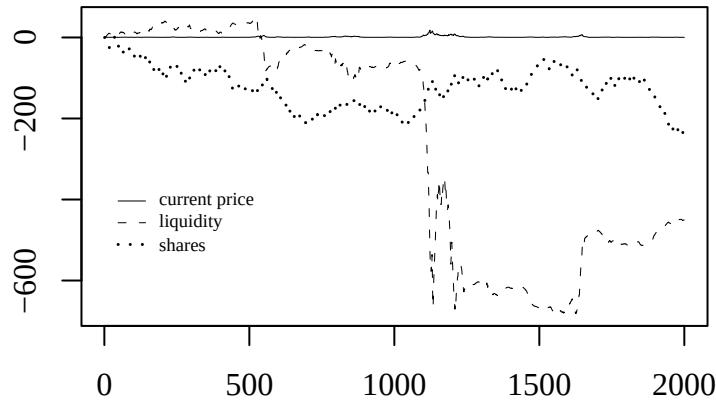


Figure 1.42. CT agents: Price, liquidity and shares series with *randomAgentNumber* = 270, *aNNForecastAppAgentNumber* = 15, *stopLossAgentNumber* = 10 and *maxOrderQuantity* = 3, *BPCTB* = 5 and *EO* = 1, *d* = 10.

Analyzing the results of the first of the five CT agents, we see in Figure 1.20 that with this degree of EO correction we have no effect on the agent wealth (evaluated on the forecasted price): With the bubble, wealth becomes largely negative and also the significant fluctuation of the share quantity of Figure 1.21 does not change the situation.

Comparing (Figures 1.22, 1.23, 1.24 and 1.25) the min and max wealth situations for the four types of agents involved in the simulation (remembering that on the border lines we have different agents as the time

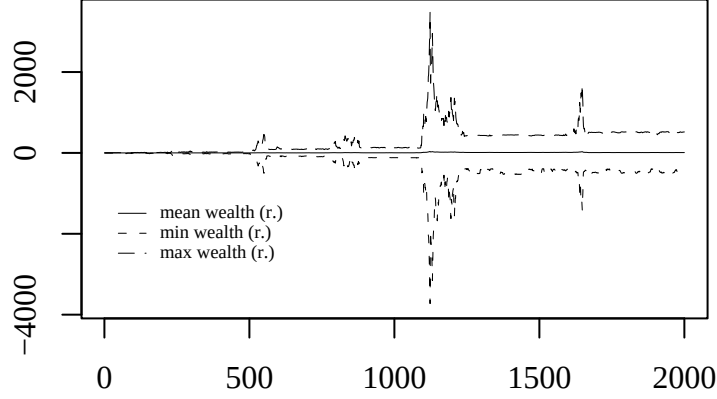


Figure 1.43. CT agents: The wealth series with $randomAgentNumber = 270$ (r.), $aNNForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ and $EO = 1$, $d = 10$.

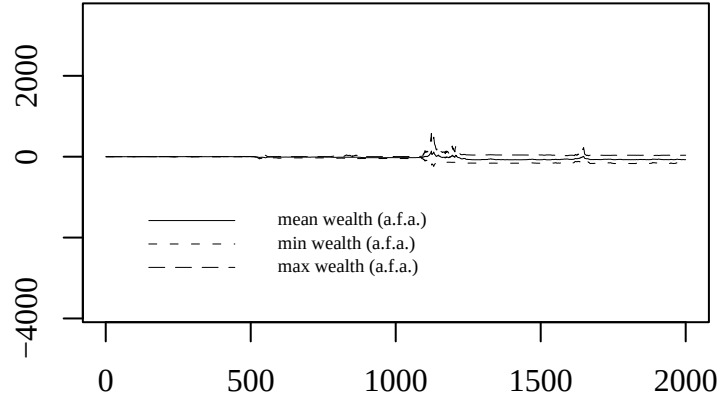


Figure 1.44. CT agents: The wealth series with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 15$ (a.f.a.), $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ and $EO = 1$, $d = 10$.

evolves), we can see that this light EO presence gives no advantage to our CT agents.

9.2. Case 2: $EO = 2$; $d = 10$

The second case, with $EO = 2$ and $d = 10$, shows (Figure 1.26) a realistic price sequence with two bubbles of an acceptable dimension in 2,000 days. The quota of correct forecasting signs is here 56%.

In Figure 1.27 we see that now the agent is able to use bubbles to increase in a significant way its wealth, tough in an unstable way. In

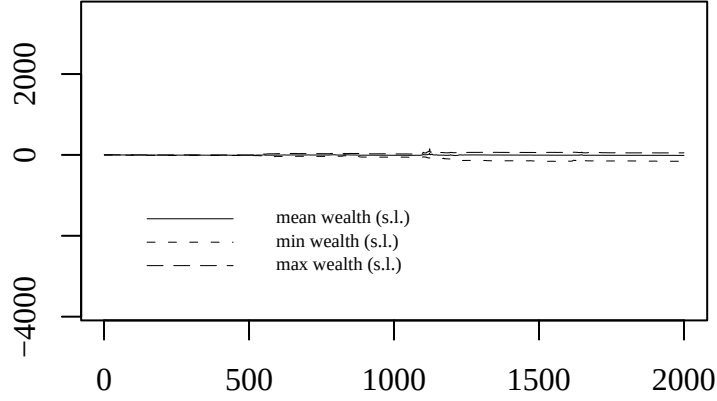


Figure 1.45. CT agents: The wealth series with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ (s.l.) and $maxOrderQuantity = 3$, $BPCTB = 5$ and $EO = 1$, $d = 10$.

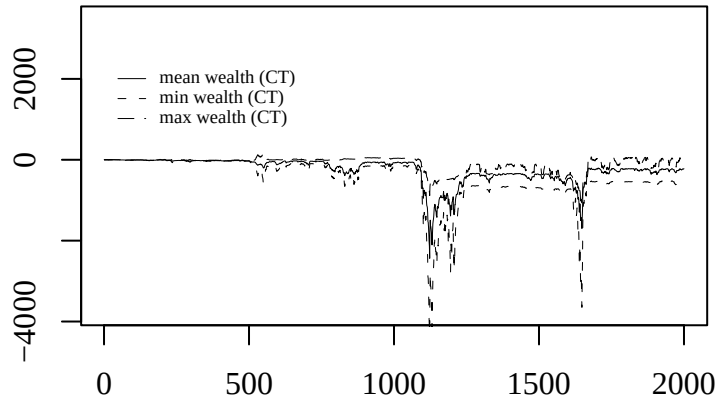


Figure 1.46. CT agents: The wealth series with $randomAgentNumber = 270$, $aNNForecastAppAgentNumber = 15$, $stopLossAgentNumber = 10$ and $maxOrderQuantity = 3$, $BPCTB = 5$ (CT) and $EO = 1$, $d = 10$.

Figure 1.28 we can see the “intelligent” use of the share quantity in presence of the two bubbles; remember that this kind of behavior is self-developed thanks to the CT method and long term learning (see above in Subsection 8.1).

Comparing (Figures 1.29, 1.30, 1.31 and 1.32) the min and max wealth situations for the four types of agents involved in the simulation, we can see that this medium EO gives a significant advantage to the CT agent group.

9.3. Case 3: $EO = 2$; $d = 20$

The 3rd case, with $EO = 2$ and $d = 20$, shows (Figure 1.33) an unrealistic price sequence with an enormous bubble, as a consequence of the forced behavior of the CT agents in the presence of this heavy EO. The quota of correct forecasting signs is anyway always 56%.

In Figure 1.34 we see that the agent reacts to the losses suffered when the bubble begins and in Figure 1.35 we can investigate the complex behavior adopted.

Comparing (Figures 1.36, 1.37, 1.38 and 1.39) the min and max wealth situations for the four types of agents involved in the simulation, we can see that this heavy EO does not give a significant advantage to the CT agent group.

9.4. Case 4: $EO = 1$; $d = 10$

The 4th case, with $EO = 1$ (i.e. to increase the wealth evaluation based on the closing price) and $d = 10$, shows (Figure 1.40) a realistic price sequence with a bubble, but the presence of this short term evaluating agents reduces the predictability of the market: The quota of correct forecasting signs is here only 49% (vs. the 55% of the basic mixed situation of Section 7).

In Figures 1.41 and 1.42 we see that the agent is not able to develop a positive strategy, also in the absence of bubbles.

The analysis of the four Figures related to min and max wealth (1.43, 1.44, 1.45 and 1.46) confirms the absolute weakness of this short term EO.

10. Conclusions

The following statements may constitute a tentative conclusion:

- first of all we can fully accept the Axelrod simplicity principle, since it is easily shown that also very simple agents can generate complex emerging structures;
- external constraints (the “book” of the market) are as strong as a mind proxy (the CT scheme) in determining the emergence of hard to explain patterns, like bubbles in a stock market;
- finally, intelligent behavior can emerge in artificial agents, on a self-developed basis.

So, the (repeated) question is: Are human agents so far from the complexity of the economic system, as ants are from their anthill? But

we underline also the result obtained: only with the presence of a (micro) “mind”, intelligent behavior—useful to the agents—can emerge.

Future directions of investigation are: the role of the environment and of the agent swarm (Bonabeau *et al.*, 1999) in developing intelligence; the micro foundation of the behavior, applying econometric techniques to the links existing between individual decision and the prices, wealth and forecasting.

This is a “universe” just waiting to be explored!

Notes

1. Keep It Simple, Stupid; Rosaria Conte in a discussion of ICCS & SS II in Paris last September proposed to reinterpret it as Keep It Simple as Suitable; my counterproposal is: Keep It Sufficiently Simple (to correspond to real life agents.)

2. Swarm is also promising as a common language in agent based simulation, having anyway a look to tools as UML (Hunt, 2000) and IMT, the Integrating Modelling Toolkit based upon XML (<http://flock.cbl.umces.edu/imt/>).

3. Or obtain it directly from the author.

4. From <http://eco83.econ.unito.it/~terna/sum/sum.html>.

5. Cross Targets (CT) are introduced in Subsection 8.1.

6. Agents considering only liquidity and share quantities.

7. This is a realistic microfoundation of price movement also in actual world.

8. A technical detail: The program emulates the different quantities in orders repeating n times—in the same tick—an order of one unit.

9. With $p_i = \text{mean price of day } i$, $d = \text{dataWindowLength}$ and $n = nAheadForecasting$ input data are: $\frac{p-d+1}{p-d+1-n}, \dots, \frac{p_0}{p-n}$.

10. For a concise introduction to ANN technique, considering multilayer feedforward ANN trained with the backpropagation technique, see Section 4 in Terna (2000a).

11. The length of the training set is defined by the parameter *bpCTPatternNumberInTrainingSet* contained in the *bp.setup* file of the package.

12. Background characteristics. Random agents: (a) can operate in the opening phase of the market, (b) can act if the price falls below a specific floor and (c) correct their order prices with a random coefficient.

13. Background characteristics. Market imitating agents: (a) can operate in the opening phase of the market, (b) can act if the price falls below a specific floor and (c) correct their order prices with a random coefficient.

14. The mean wealth of each category is no more constantly zero, since we have exchanges among agents of different categories.

15. Background characteristics. Locally imitating agents: (a) can operate in the opening phase of the market, (b) can act if the price falls below a specific floor and (c) correct their order prices with a random coefficient.

16. We do not recall, here and after, the content of the note about mean wealth, which is anyway always relevant.

17. Background characteristics. “Stop loss” agents: (a) can operate in the opening phase of the market, (b) can act if the price falls below a specific floor and (c) correct their order prices with a random coefficient, but when adopting the stop loss strategy, they use directly the last executed price for their buy or sell proposal.

18. In this experiment we account for the real agent situation (“long” or “short” on the market); see the *checkingIfShortOrLong* parameter in Section 3.

19. A student of mine is substituting the forecaster with an “astrologer” producing its estimates on the basis of random applied rules, with the purpose of simulating various degrees of self-fulfilling prophecies.

20. Background characteristics. Applying forecast agents: (a) can operate in the opening phase of the market, (b) can act if the price falls below a specific floor and (c) correct their order prices with a random coefficient.

21. SUM, as bp-ct, internally uses the ERA scheme (Terna, 2000a), with Rule Masters and Rule Makers.

22. It can be obtained directly from the author or downloaded following the link to the Luna and Stefansson book at <http://www.swarm.org>, or from the “anarchy” Section in the same site (the present version is 1.1).

23. This is an innovation to the methodology used in Terna (2000a).

24. Background characteristics: CT agents (a) can operate in the opening phase of the market, (b) do not care if the price falls below a specific floor and (c) correct their order prices with a random coefficient.

References

- Axtell, R. (2000). “Why agents? On the varied motivations for agent computing in the social sciences.” Center on Social and Economic Dynamics, Working Paper No. 17, <http://www.brookings.edu/es/-dynamics/papers/agents/agents.htm>.
- Axelrod, R. (1997). “Advancing the Art of Simulation in the Social Sciences.” Conte, R., Hegselmann, R. and Terna, P. (eds.), *Simulating Social Phenomena, Lecture Notes in Economics and Mathematical Systems 456*, pp.21-40, Berlin: Springer.
- Beltratti, A., Margarita, S., Terna, P. (1996). *Neural Networks for Economic and Financial Modelling*. London: International Thomson Computer Press.
- Bonabeau, E., Dorigo, M., Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Science of complexity. New York: Oxford University Press.
- Gilbert, N. and Terna P. (2000). “How to build and use agent-based models in social science.” *Mind & Society*, 1.
- Hunt, J. (2000). *The Unified Process for Practitioners: Object Oriented Design, UML and Java*, Berlin: Springer.
- Kirman, A. (1993). “Ants, Rationality, and Recruitment.” *Quarterly Journal of Economics*, 1.
- Lo, A., Mamaysky, H. and Jiang, W. (2000). “Foundations of technical analysis: computational algorithms, statistical inference, and empirical implementation”. *Journal of Finance*.
- Ostrom, T. (1988). “Computer simulation: the third symbol system.” *Journal of Experimental Social Psychology*, Vol. 24, pp.381-392.
- Sargent, T.J. (1993). *Bounded Rationality in Macroeconomics*. Oxford: Clarendon Press.

- Terna, P. (2000a). Economic Experiments with Swarm: a Neural Network Approach to the Self-Development of Consistency in Agents' Behavior, F. Luna and B. Stefansson (eds.), *Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming*. Dordrecht and London: Kluwer Academic.
- Terna, P. (2000b). "The «mind or no mind» dilemma in agents behaving in a market." Ballot, G. and Weisbuch, G. (eds.), *Applications of Simulation to Social Sciences*. Paris: Hermes Science Publications.
- Tesfatsion, L. (2001). "Introduction to the special issue on agent-based computational economics." *Journal of Economic Dynamic and Control*, Vol. 25, Issue 3-4, pp.281-293.