Econophysics: The emergence of complexity patterns in a stock market using agent based models

Umberto Forte

Abstract

In this work, adopting an agent based approach, we studied the connection between microscopic behavior of the investors and complex dynamics in a stock market .

To this extent we modeled social interactions of agents, focusing on their behaviors considered to be the cause for the so called "herding effect", which is behind phenomena such as market's bubbles and crashes.

In building this model we took inspiration by the B.LeBaron [1] important scientific contribution to agent based stock market modeling, while we considered big data studies by Pentland [3] to model social interactions between agents.

In this paper we critically analyze both works, highlighting some critical issues coming from the classical financial description of equilibrium theory and "rational markets hypothesis".

Then we reviewed the main steps in model implementation, providing a description of the main problems encountered and the solutions adopted. The program was written in Netlogo 5.2, which is the most widely used framework for agent based simulating in the social sciences.

We finally compare time series obtained from simulated market returns, stored in a .csv file, with a real market one (Nasdaq index from 1985 to 2016), and we show enough significant statistical similarities in terms of complexity that makes us able to assert that the social interactions considered in this model may be responsible for real market complexity.

Contents

1 Introduction

This model fits into the line of studies on stock market through agent-based models. Many similar studies already exist in literature such as for example the well-known Santa Fe Artificial Stock Market B.LeBaron [1] or T. Lux [5].

In this section we briefly discuss about agent based modeling of stock market modeling by listing some previous works that have inspired us in designing our model.

We will first look at the first pioneering ABM on the stock market that is the so-called "Santa Fe artificial stock market"[2], from the Santa Fe Institute of Technology research group that carried out this study.

1.1 The Santa Fe artificial stock market

The most popular agent based model for stock market is arguably the *Santa Fe artificial stock market*, which was designed as described in paper LeBaron [2] using the Swarm simulation package, a simulating environment written in Objective C and Java.

The artificial stock market was used by authors to test various hypothesis and settings about stock market. In particular the authors gave more emphasis to the emergence of complexity, trying to find sufficient evidence to support an alternative approach to classical theory of rational investors expectations.

In this model agents are able to optimize their portfolio strategy by diversifying their investment in a risk-less asset (as BTP, BUND, T-Bills, or also making a deposit in a bank) which ensure a fixed rate, and a risky one. Choosing a finite number N of agents (about 100 agents) which interact just by mean of market dynamics and not in other ways as seen by other simulation works. Dividends for assets at time $t+1$ are generated by an artificial Ornstein-Uhlenbeck process $d(t+1) = \rho d(t) + \alpha\eta(t)$, where $\eta(t)$ is a zero norm gaussian noise.

Each investor diversify his investment by choosing a *market share* $h_i(t)$, which represent the relative percentage of financial wealth invested in the risky asset, so that the one invested in the risk-less one will be $1 - h_i(t)$. This way the agents will have as total financial wealth $w_i(t) = M_i(t) + h_i(t)p(t)$, that is the amount of householders wealth invested in financial activities.

The agent have the overall market information consisting in: value of price, values of dividends, total number of asks and bids for a price at each time by the investors; and they can use different statistical predictive models to optimize their strategies.

All these informations are summarized in a seven binary digits string, as required by the usage of a genetic algorithm. Where each digit takes values 0 or 1, corresponding to the seven possible predictors that can be rather adopted or replaced by agents.

The fist three predictors are related to fundamentalist agent strategies. The *fundamentalists* are agents that compare the market price of a security with the "true value" of the corresponding asset.

Three other predictors take into account of technical investors strategies. Where *technical investors* are those who analyze market trend using moving averages for different estimation periods, and other price forecasting statistical techniques.

They assume that all the available information on the asset is completely discounted only in its price and that by analyzing the time series of its past realizations it is possible to predict its future outcomes.

Each agent is able to optimize his portfolio choice according to the *efficient markets hypothesis* (by Fama and French studies), which provides an analytical solution for optimal market shares in case of constant absolute risk aversion (CARA) of the investor

$$h_i = \frac{\mathbb{E}_i[p(t+1)+d(t+1)] - p(t)(1+r)}{\lambda\sigma_i^2} \tag{1.1}$$

The letter λ indicates a constant value for relative risk aversion of all investors. The expectation value of future asset returns is $\mathbb{E}_i[p(t+1)+d(t+1)]$, and it variates from one agent to another due to the usage of different forecasting strategies.

This study also aims to understand when the transition between rational and complex regimes occurs, and this is achieved by studying how model's complex behavior may change by varying a particular parameter, representing the genetic algorithm update frequency.

According to classical financial theory, asset information is all discounted in its price, which means that each agent is aware of the other agents expectations in every time. This implies that price goes randomly towards an equilibrium state, leaving no chance to technical investors to find considerable profit opportunities by predicting a specific trend.

But this is not what happen in a real markets,where each agent will have its own independent expectation on the basis of what he thinks other agents expectations could be at a certain time. In every moment agent-i will estimate $\mathbb{E}_i[d_{t+1} \mid I_t]$ and $\mathbb{E}_i[p_{t+1} \mid I_t]$ with volatility $\sigma^2_{i,t}$, he will determine a price

$$p_t = \beta \sum_j w_{j,t}(\mathbb{E}_j[d_{t+1} \mid I_t] + \mathbb{E}_j[p_{t+1} \mid I_t]) \tag{1.2}$$

where $\beta{=}\frac{1}{1+r}$, with r indicates the risk-free rate, and $w_{j,t} = \frac{\frac{1}{\sigma^2_{j,t}}}{\sum_k \frac{1}{\sigma^2_{k,t}}}$ the relative confidence for the the agent-j expectation.

In case of homogeneous agents' expectations due to the fact that the agents use the same predictive methods, the efficient markets theory states that every agent is aware of any other agent expectation and future prices can be predicted with high accuracy, so that in this way price dynamics is already determined from the start.

Tanks to rational agents' expectations, by mean of CARA analytic solutions, different expectation values may converge to the same equilibrium price.

While case of heterogeneous agents' expectations instead, the agents have got different forecasting models. They cannot provide the other agents' expectations with great accuracy as in the previous case, and therefore price indeterminacy will increase as

$$\mathbb{E}_i[p_{t+1} \mid I_t] = \beta\mathbb{E}_i\left[\sum_j w_{j,t+1}\left(\mathbb{E}_j\left[d_{t+2} \mid I_t\right] + \mathbb{E}_j\left[p_{t+2} \mid I_t\right]\right)\middle| I\right] \tag{1.3}$$

, and price in from (1.2) will inevitably in turn depend on other future expectations.

We finally obtain this way after k-steps recursively a complete uncertainty on price dynamics, and that's the reason why a *deductive reasoning* like this is impracticable in case of agents with heterogenous expectations. Adopting an *inductive reasoning* agents may choose among different forecasting hypothesis and select at each time that one which gives the best accuracy.

The agents can do this selection adopting genetic algorithms, which allow to optimize agents' strategies. This selection techniques adopt a *fitness* criterium to choose “best strategy”. This criterium allows to assign weights to predictive hypothesis on the basis of their predictions accuracy, in order to optimally combine them in new crossed strategies assigning higher weights to those which best performed.

This approach make us able to take into account of model complexity, generated by individual heterogeneous expectations combined in a non-linear way causing speculative bubbles and crashes. That's the case wherein technical investors are able to predict future prices outcomes with high accuracy discovering profit opportunities.

So in a first step authors conduced experiments looking for results already provided by the classical theory, finding an equilibrium price . Then in a second step, they introduced agents' heterogeneity assigning them different linear predictors, and they still observed convergence to an equilibrium state.

Therefore in a last step, making agents choose their strategies inductively by using genetic algorithms, they observed values for volatility and trading volumes comparable with those observed in real markets.

But in [1] as in other works of agent based simulation of stock market, time series of endogenous quantities such as future asset dividends are generated though a standard stochastic model. This mathematical description aims to capture uncertainty surrounding market dynamics and it succeeds in describing some realistic elements,but it doesn't explain microscopical interactions responsible for such a dynamic.

But this description ignores the complexity and nonlinearity present in the interactions of agents and typical of many social phenomena, such as those held responsible for the so-called "herding effect", that is the cause of bubbles and market crashes.

From this point of view the classical techniques of portfolio optimization become meaningless, because they are based on the efficient market hypothesis. In which the investor decisions are made according to an utility function which leads to analytical solutions, as showed in equation (1).

In real stock markets price is continually updated by single exchanges occurring between buyers and sellers, and not just at the equilibrium when demand meets supply, as assumed in most of the models. In fact price mainly depend on desires and expectations of the investors, which are the real source of its complexity, and that allow it to evolve out of equilibrium.

This model [1] also uses fundamentalist investors, who estimate a "true price" for the risky asset by comparing multiple macroeconomic data from a real market. But in the model adopted have been considered dividend time series generated by a standard stochastic process, so taken by the model itself and not the result of the data analysis of a real market.

The exchanges between agents in a real market not occur simply at random but, on the contrary, the investors show the tendency to come back from their counterpart with which they made good business. In fact they represent the small group of agents in which they trust, as discussed by Pentland [3]. But on the contrary in a perfect market every agent should always trade with a different agent, according to price fluctuations .

So we can highlight two main types of market exchanges. One corresponding to real observations, in which individuals trade following an underlying *social network* based on trust relationships; and a theoretical one, in which exchanges occur completely at not allowing that an agent reaches an higher centrality against the other agents.

In Pentland [3] the author describes first type of dynamics as that which would lead to creation of a network structure, which it proves to be very robust in periods of high market uncertainty characterized by high volatility in prices and returns. This dynamics also leads to lack of contrast in the new market strategies, destabilizing it by encouraging the emergence of aggregation phenomena such as the herding effect.

The investor interactions in today's financial markets are better described by dynamics based on trust relationships than one in which the exchanges just simply occur at random. Today market information can spread more quickly thank to social media, ensuring that a greater number of agents follow a few number of specialists trading strategies, resulting in larger bubbles and crashes. This happens because the investors still choose to trade with a few trusted individuals, who now get a great centrality within a larger network.

2 The model

2.1 Building the model

Using the analogy between a spin system and the binary choice of buying or selling, we decided to fit the problem for agent based simulating. Therefore we introduced zero-intelligence agents also called **random traders**, i.e. agents who do not follow a specific strategy but that randomly "flip" their spin according to Bernoulli extractions.

We also introduced a kind of agents, called *followers*, able to follow the most successful agent strategies, as addressed by Pentland [3] in his work on the world's largest web day trading platform called *eToro*. On this trading platform the investors can see other users' strategies, and may choose to follow the someone else strategy. In this way the aforementioned dynamic is triggered, i.e. the investors will tend to follow the most successful users.

Within these *followers* we select a few agents with highest financial wealth, the so called *hubs*, among which the *followers* can choose the one from which to copy a strategy.

This way the hubs will be able, differently from other agents, to significantly affect the macroscopical price dynamics , introducing the heterogeneity features necessary for a real complex description.

2.2 The code

Price has been updated just proportionally to difference between number of buyers and sellers, where the proportional coefficient has a role similar to investors *relative risk aversion,* present in classical in financial theory.

Here we show the setup procedure, where the main features of the agents are set. We select number of agents to be assigned to each class based on relative percentages (howManySavers, howManyFollowers, howManyRandomTraders, howManyHubs), in order to perform simulations by varying the size (total number of agents) without changing the relative percentages for the various types of agents.

```
to setup
  clear-all
  reset-ticks
  set trend 0
  set price initialPrice   set prices []
  create-savers howManySavers * (#Agents / 100) [setxy random-xcor random-ycor
```

```
                                                  set action ""
                                                  set shape "arrow"
                                                  set color gray
                                                  set size 2
                                                  set openingPrice price]
  let i 0
  ask savers [ifelse i < positiveSavers * count savers / 100[ set action "buy"
                                                               set order 1
                                                               set shape "arrow"
                                                               set color green
                                                               set i i + 1
                                                             ][ set action "seller"
                                                               set order 10
                                                               set shape "downArrow"
                                                               set color red]]
  create-followers howManyFollowers * (#Agents / 100)[ setxy random-xcor random-ycor
                                                        assignFollowersQualities
                                                        set order (random maxQuantities + 1)
                                                        set detrust 0
                                                        set wealth random-normal 1000 500]
  ifelse randomClosure?[  ask followers [set closingPercentage (random 10) + 7]
                     ][  ask followers [set closingPercentage setClosingPercentage]]
  create-randomTraders howManyRandomTraders * (#Agents / 100)[
                                    setxy random-xcor random-ycor
                                    set shape "computer workstation"
                                    set size 1
                                    set color grey
                                    set order 1]
  loop[ ifelse count hubs < howManyHubs * (#Agents / 100)[
                   ask one-of followers with [wealth = max [wealth] of followers][
                                                 set breed hubs
                                                 assignHubsQualities
                                                 let j random 2
                                                 set openingPrice price
                                                 ifelse j = 0 [
                                                    set action "buy"
                                                    set color green
                                                 ][ set action "sell" set color red]
                                                          ]][stop]]
end
```

We also experimented what if the hubs could change their "strategy" by reversing their trading position in case they materialize too many consecutive losses. This mechanism is controlled by a threshold HLT (hubs loss tolerance) which measures the number of consecutive losses that a hub can tolerate.

The *netResult* variable measure price variation from the position opening price to its current value. For example, if the agent decides to sell and price falls down from the opening value the netResult will be positive because the investment position is gaining value; on the other hand if the agent sells and price goes up, netResult will result negative.

We introduced a new agents "state", different from buy or sell, which is *passing traders*, to which they come in case of bankruptcy (i.e. when they completely lose their wealth) or if they can no longer purchase any stocks (cash <price).

```
to changeStrategy
  let loserHubs []
  ask hubs with [action != "pass"][
                      ifelse netResult-1 > netResult[
                             if waiting >= hubsLossTolerance[
                                    set loserHubs fput who loserHubs
                                    set waiting 0]
                             if waiting < hubsLossTolerance [
                                    set waiting waiting + 1
                                 ]][set waiting 0]]
  foreach loserHubs [ask hub ? [ifelse action = "buy" [
                                   set action "sell"
                                   if PassBeforeChange? [
                                   set order (random maxQuantities + 1)
                                    set netResult 0
                                   set netResult-1 0]
                               ask link-neighbors [
                                   set action  "sell"
                                   if PassBeforeChange? [
                                            set order (random maxQuantities + 1)
                                            set netResult 0
                                            set netResult-1 0]]
                             ][set action "buy"
                               if PassBeforeChange?[
                                      set order (random maxQuantities + 1)
                                      set netResult 0
                                      set netResult-1 0]
                               ask link-neighbors [
                                    set action "buy"
                                    if PassBeforeChange? [
                                            set order (random maxQuantities + 1)
                                            set netResult 0
                                            set netResult-1 0]]]]]
end
```

Making hubs can change their strategy, we introduced a control parameter that is the *hubsLossTolerance* (HLT). By varying this parameter we discovered that for high values (setting HLT about 100) hubs maintain heterogeneous strategies and trend is determined by random traders, and price follow a pure random walk. While setting low values for parameter (HLT < 20) also small price fluctuations can lead hubs and their followers to be polarized in a global behavior, corresponding to the so-called herding effect, that brings price to rise or fall very quickly.

It follows that by varying this parameter we are able to control strength of the herding effect on our simulation, causing markets bubbles and crashes. We can also program some simulated stock market crisis at specific time steps, in order to study their effect on volatility by analyzing the price time series.

We also introduced the possibility for *followers* and *hubs* to close their position in case they reach a satisfactory value of gain, by assigning a *closingPercentage*, which represents the percentage change in price, starting from the opening value, to which the agents decide to close their position. This new feature brings stability in price fluctuations, because as soon as the price would be higher or lower than a certain value there will be buyers or sellers, that being satisfied, will decide to close their buying or selling position, bringing price to a balance. So by this parameter we are implicitly able to control average price fluctuations, because on average the price will not continue to increase or decrease for more than a certain period, with no agents closing their position.

We can choose by the *passingPercentage* the relative number of agents which can pass so that the *passingTraders* are kept constant. It follows that if number of passing agents exceed the specified threshold, they will return in the game tanks to *recconnectFollowers* procedure.

```
to reconnectFollowers
 if count hubs < howManyHubs * (#Agents / 100) [reassignHubs]
 let passingTraders followers with [action = "pass"]
 let activetraders turtles with [ breed = followers and action != "pass"]
 if count passingTraders > passingPercentage * (count followers)/100[
    if count hubs != 0[
       ask n-of (count passingTraders - passingPercentage * (count followers)/100) followers wit
       action = "pass"][
                      ifelse count link-neighbors = 0 [
                                   create-link-with one-of hubs
                                   set openingPrice price
                                   set order (random maxQuantities + 1)
                                   set action [action] of one-of link-neighbors
                                 ][set action [action] of one-of link-neighbors]]]]
  ask activeTraders with [count link-neighbors = 0][
                                   create-link-with one-of hubs
                                   set openingPrice price
                                   set order (random maxQuantities + 1)
                                   set action [action] of one-of link-neighbors]
end
```

We also introduced a different type of investor, the *savers*, which can only keep a constant position for all the simulation time. So if they start buying they will keep buying, or selling otherwise, till the end of the simulation. We chose to control relative numbers of savers by varying the *howManySavers* percentage, and how many of them choose to buy or sell by the *positiveSavers* percentage. In this way we can select distance between number of savers who buy and those who sell, and it will determine the average price trend.

The true core of this simulating program is the price updating procedure. We call all the agents (followers, hubs, and savers) which don't pass, and counting how many of them choose to buy or sell we update the price proportionally to this difference, so the price will evolve with positive or negative trend depending on difference of how many agents buy or sell representing the analogue of *excess markets demand* . And we update price multiplying this difference times a very small proportionality coefficient ($\gamma = 0.01$), which is the equivalent of relative risk aversion coefficient in the analytical solution.

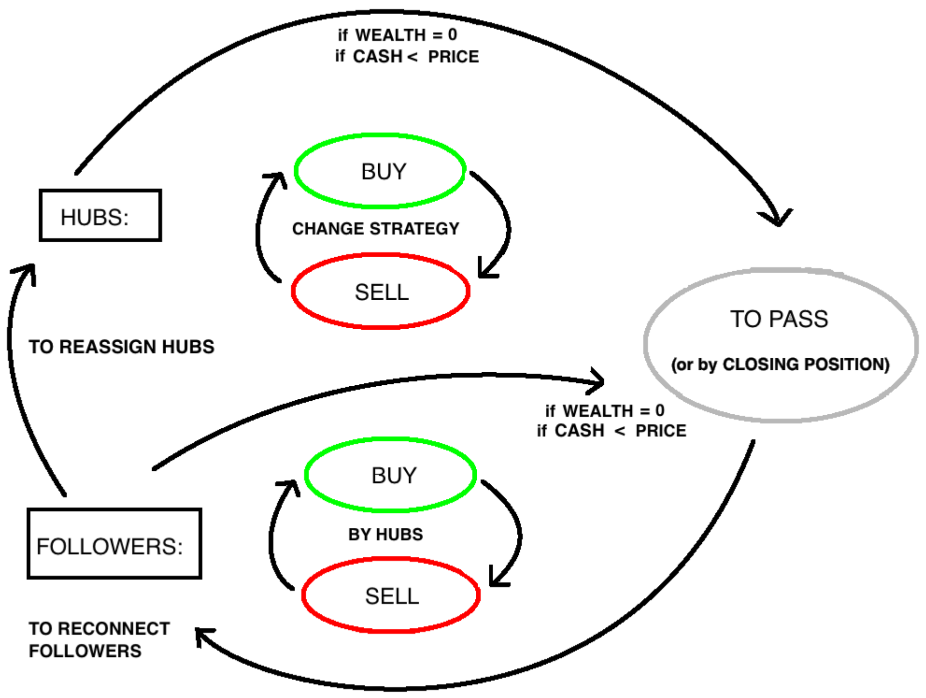
So we also update value of the agents trading position, which is represented by the *netResult* variable. But we also update a *netResult-1* variable we use to record the last position value that we need while running *changeStrategy* procedure. Where comparing netResult with netResult-1 agents are able to determine whether their position is gaining or losing value.

```
to changePrices
  ask turtles with [breed = hubs or breed = followers] [
                                    set netResult-1 netResult]
  let traders turtles with [
      (breed = hubs or breed = followers
       or breed = randomTraders or breed = savers)
                  and action != "pass"]
  let sellers count traders with [action = "sell"]
  let buyers count traders with [action = "buy"]
 ifelse quantities? [
      let buyersOrders []
      let sellersOrders []
      ask traders with [action = "buy"][
          set buyersOrders fput order buyersOrders]
      ask traders with [action = "sell"][
          set sellersOrders fput order sellersOrders]
      let delta  0.005 * abs (sum buyersOrders - sum sellersOrders)
      if sum buyersOrders > sum sellersOrders[
                     set price price + delta
                     ask traders with [action = "buy"][
                       set netResult netResult + (delta * order)]
                     ask traders with [action = "sell"][
                       set netResult netResult - (delta * order)]]
      if sum buyersOrders < sum sellersOrders[
                        set price price - delta
                        ask traders with [action = "buy"][
                            set netResult netResult - (delta * order)]
                        ask traders with [action = "sell"][
                            set netResult netResult + (delta * order)]]
][let delta 0.005 * abs(buyers - sellers)
  if sellers > buyers[ set price price - delta
                       ask traders with [action = "sell"][
                           set netResult netResult + delta ]
                       ask traders with [action = "buy"][
                           set netResult netResult - delta ]]
  if sellers < buyers [set price price + delta
                       ask traders with [action = "sell"][
                           set netResult netResult - delta ]
                       ask traders with [action = "buy"][
                           set netResult netResult + delta ]]
  ifelse buyers - sellers > 0 [set trend 1][set trend -1]]
  set prices lput (list ticks price) prices
end
```

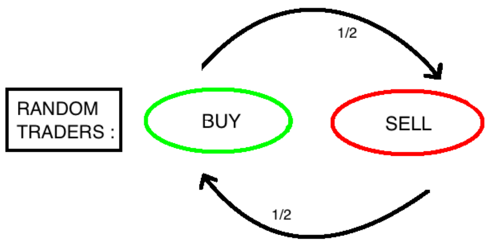
And this is the *closePosition* procedure whereby followers and hubs can close their positions if they reach a satisfactory profit. If an *hub* decides to close its position his links will be removed, while followers which close their position only need to remove one link. To pass agents will be reset to the state of followers, also if they are hubs, and all their qualities are reset including wealth.

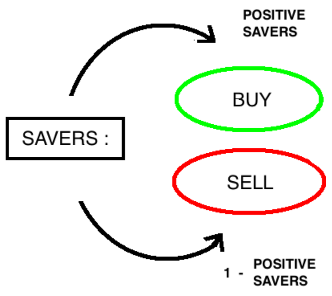
```
to closePosition
  let traders turtles with [ breed = hubs or breed = followers and action != "pass"]
  ask traders [if netResult >= (closingPercentage * (openingPrice / 100))[
                    set wealth wealth + netResult
                    if breed = followers and count link-neighbors != 0[
                           ask link-with one-of link-neighbors[die]]
  if breed = hubs [ask link-neighbors [ask link-with one-of link-neighbors[die]]
                   set breed followers]
  assignFollowersQualities  set action "pass"]]
ask traders with [action = "pass"][
      set wealth random-normal 1000 500
      set netResult 0  set netResult-1 0]
end
```

We can summarize the main model features as showed in the following diagram. The hubs choose their strategy initially at random and then change it according to the *changeStrategy* procedure, while the followers only copy their strategy. To pass all agent, including hubs, have to be reset to followers. Both hubs and followers pass if they go bankrupt (reaching a zero wealth), or if they can't pay for a stock, or even if they choose to close their position. Then *passingTraders* can be back to trade by *reconnectFollowers*, and hubs are substitute by followers by the *reassigningHubs* procedure



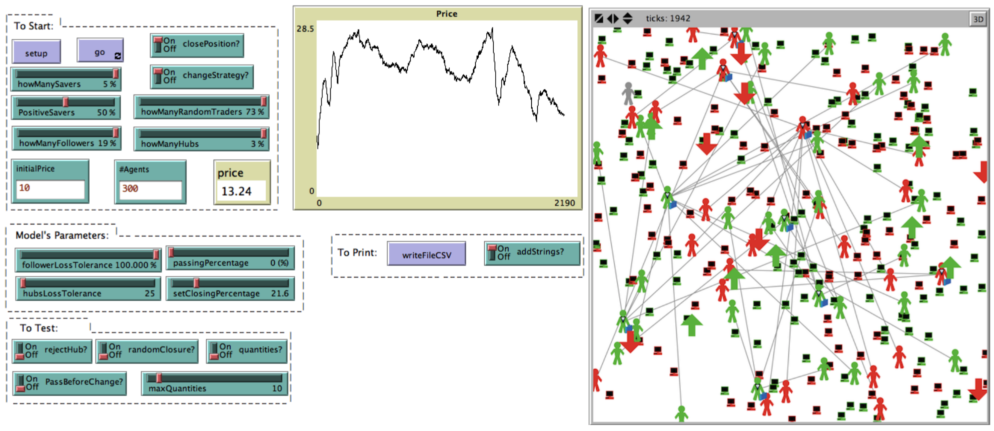
Random traders can only randomly choose their strategy every time by a binomial process, while savers strategies are chose by mean of a model parameter (positiveSavers) to select relative distance between number of buyers and sellers. By definition the savers will maintain the same strategy (buying or selling) for all simulation time.





2.3 Experiments

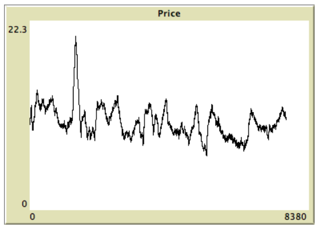
The program interface looks as reported in the following image:



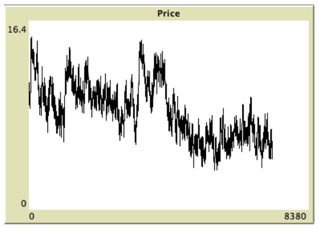
closePosition? On
changeStrategy? On
howManySavers 5 %
PositiveSavers 50 %
howManyRandomTraders 73 %
howManyFollowers 19 %
howManyHubs 3 %
initialPrice 10
#Agents 300
price 13.24
followerLossTolerance 100.000 %
passingPercentage 0 (%)
hubsLossTolerance 25
setClosingPercentage 21.6
rejectHub? Off
randomClosure? Off
quantities? Off
PassBeforeChange? Off
maxQuantities 10
addStrings? On

In NetLogo graphics savers are represented with arrows, the *hubs* and *followers* by person shapes, and *randomTrader* by the computer shape. In the interface there is a first command block by which we can choose the *initialPrice* of the stock, the simulation size (total number of agents), and relative percentages for all agent types. E.g. selecting 100 agents, with 5% of savers 19% of followers, I will have 5 savers 19 followers, and so on.

We decided to test whether the size of the simulation affects the price dynamics, keeping constant model parameters and the agents' type percentages, we conducted two simulation sessions first with 100 and then with 1000 agents. We noted that, as shown by the following plots, a larger number of agents leads to an increase in noise, as a result of the greater number of *randomTraders*.



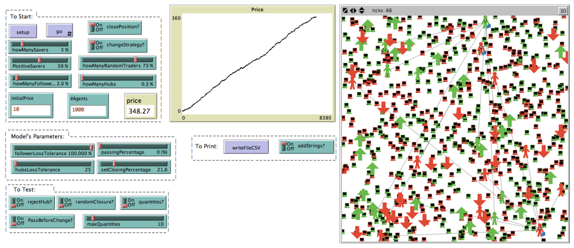
I) 100 agents' market



II) 1000 agents' market

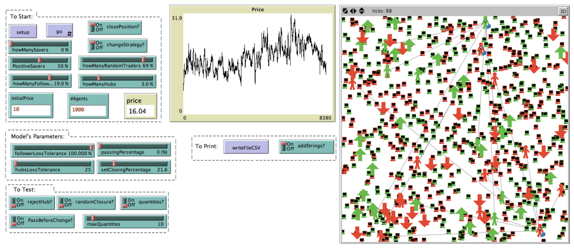
We also tried what happen if we increase the model size leaving unchanged number of hubs and number of followers (not keeping constant only percentages as before), as in the following example are 19 and 3 respectively. We saw that higher price fluctuations with hubsLossTolerance = 25, caused by a relative higher percentage of randomTraders, didn't allow hubs change their strategy, because price random behavior doesn't allow too much consecutive losses to occur, leading to a long bull (increasing) trend carried by savers effect.

When we maintain the same relative percentages of agent types in the model we realize lower price changes but with much higher volatility, because of the stronger herding effect. We observe that there are more followers than randomTraders, because the first cause this complex dynamic.



howManySavers 5 %
PositiveSavers 50 %
howManyRandomTraders 73 %
howManyFollowe... 2.0 %
howManyHubs 0.3 %
initialPrice 10
#Agents 1000
price 348.27
followerLossTolerance 100.000 %
passingPercentage 0 (%)
hubsLossTolerance 25
setClosingPercentage 21.6
maxQuantities 10

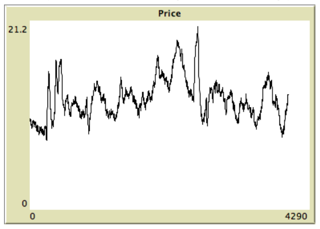
I) 1000 agents and same previous number of followers



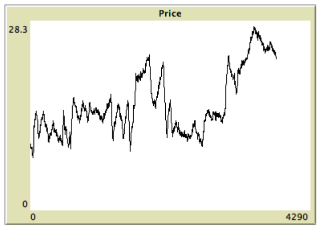
howManySavers 0 %
PositiveSavers 50 %
howManyRandomTraders 69 %
howManyFollow... 19.0 %
howManyHubs 3.0 %
initialPrice 10
#Agents 1000
price 16.04
followerLossTolerance 100.000 %
passingPercentage 0 (%)
hubsLossTolerance 25
setClosingPercentage 21.6
maxQuantities 10

II) 1000 agents and same previous percentages of followers

Then we carried out experiments on what happen allowing agents to pass before they change strategy (buy -> sell, or vice versa). In this case no substantial differences have been detected, as also showed in the following price plots. This demonstrate that both in cases in which followers and hubs continuously change strategies, that in cases in which they have to pass from their position before they can change strategy, the effect is the same .



I) without passing before change strategy

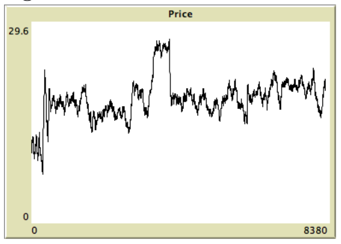


II) passing before change strategy

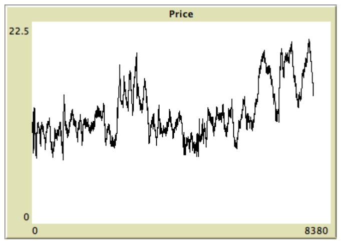
Then we studied what happen if, instead of assigning a unique *closingPercentage* for all the agents, we assign it randomly to each agent. As described by the following code block, in which we chose a specific range of values which resulted interesting from our tests, and that is $closingPercentage = 7 \div 20$.

```
ifelse randomClosure? [ask followers [set closingPercentage (random 14) + 7
                     ]][ask followers [set closingPercentage setClosingPercentage]]
```

This change has not had any effect, but that could suggest how to reduce external commands by assigning different behaviors to the agent population.



I) with random closure

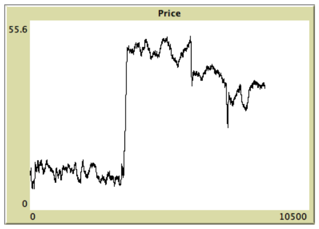


II) without random closure

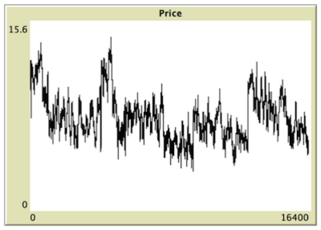
The *lossTolerance* of hubs is a model parameter very important for the model, while some burst in price trend are due to passing hubs. This parameter seems to be very similar to refresh rate present in genetic algorithms, as in B.LeBaron [1], because it accelerate the herding effect when HLT is low or to prevent it's high.

So this can be considered as the main cause for herding effect, that is the lack of system robustness under external shocks, due to a strong polarization. Then we can think to reproduce bubbles and crashes in vitro to study complex price dynamics .

In order to highlight the importance of *hubsLossTolerance*, we compare two situations, one in which hubs much hardly change their strategies with one in which hubs more frequently switch between states in which the system result fully polarized in herd behaviors. So we chose two settings $HLT = 23$ and $HLT = 12$ to test those cases, and as reported in figures,we observed in the first case strong evidence for herding effect, while in the second we note a more noisy and stochastic behavior typical of high uncertainty periods on markets.

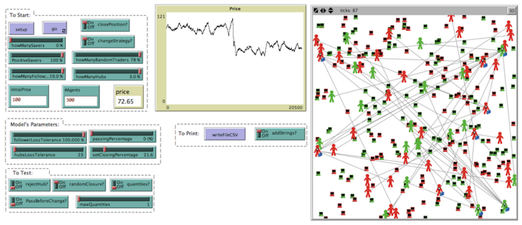


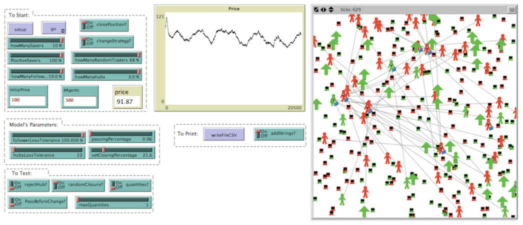
I) low hubs tolerance



II) high hubs tolerance

We compared cases in presence of savers or not in the model, and we note that their introduction lead to a lightly more stable trend. So we proved that savers introduction make more stable the market, but in case of much more savers than followers trend price will always increase (or decrease, depending on savers net strategy). We found a specific limit ratio of savers per followers, that is 5 to 8, and we also observed how exceeding this limit ratio there will be a constant trend.


I) market without savers


II) market with savers

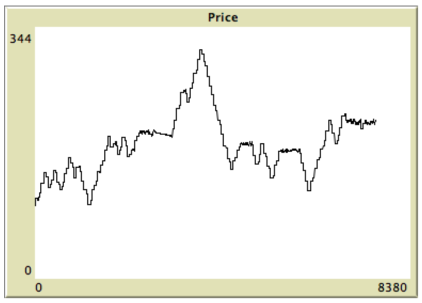
2.3.1 Quantities of stocks

Then we tried to observe what happen letting agents trade quantity of stocks different from unit. To address this issue we adopt a simulating trick, which we call *parallel order executing*, that consists in in multiplying single agents orders counting them as much times as the quantity of stocks the agents wants to trade. I.e. if a buyer decides to buy 5 stocks we count 5 buyers in the price updating procedure, in this way previous price dynamics will result more simple to manage for cases like this.

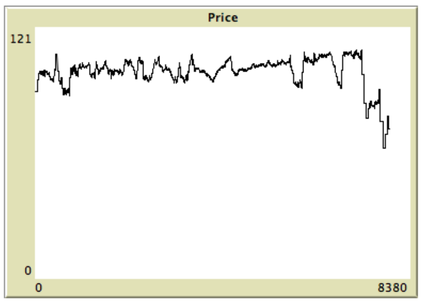
```
ifelse quantities? [
    let buyersOrders []
    let sellersOrders []
    ask traders with [action = "buy"][
      set buyersOrders fput order buyersOrders]
    ask traders with [action = "sell"][
        set sellersOrders fput order sellersOrders]
    let delta  0.005 * abs (sum buyersOrders - sum sellersOrders)
    if sum buyersOrders > sum sellersOrders[
                        set price price + delta
```

```
                    ask traders with [action = "buy"][
                        set netResult netResult + (delta * order)]
                    ask traders with [action = "sell"][
                        set netResult netResult - (delta * order)]]
    if sum buyersOrders < sum sellersOrders[
                    set price price - delta
                    ask traders with [action = "buy"][
                        set netResult netResult - (delta * order)]
                    ask traders with [action = "sell"][
                        set netResult netResult + (delta * order)]]
  ][let delta 0.005 * abs(buyers - sellers)
    if sellers > buyers [ set price price - delta
                    ask traders with [action = "sell"][
                        set netResult netResult + delta ]
                    ask traders with [action = "buy"][
                        set netResult netResult - delta ]]
    if sellers < buyers [ set price price + delta
                    ask traders with [action = "sell"][
                        set netResult netResult - delta ]
                    ask traders with [action = "buy"][
                        set netResult netResult + delta ]]
    ifelse buyers - sellers > 0 [set trend 1][set trend -1]]
    set prices lput (list ticks price) prices
```

Introducing quantities in trading orders and keeping same before *hubsLossTolerance* ($HLT = 25$), we note that price fluctuate more and it needs more time to invert trends. We have also kept constant to zero the *passingTraders* percentage. But for enough high HLT price results more static with maximum variation of 10%



I) random quantities with HLT=25



II) random quantities with HLT=150

2.3.2 To reject hubs

We wanted to experiment what if followers were able to change their hub if they suffer too much losses, rearranging change of strategies by hubs in case of followers dynamics. So programming a to *rejectHubs* procedure. We decided to test this hypothesis to verify if also this social behavior could generate herding dynamics, and if it is more determinant in emerging of this effect.

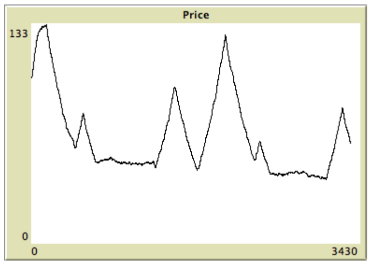
As in the *changeStrategy* procedure we increment a follower attribute (called *detrust*) at each time followers realize consecutive losses in their *netResult*. Then comparing this variable with a threshold controlled by a parameter, called *followerLossTolerance* (FLT), followers can look for a different hub to connect to.

```
to rejectHub
  let loserFollowers []
  ask followers with [action != "pass"][
                ifelse netResult-1 > netResult [
                       if detrust >= followerLossTolerance  [
                                   set loserFollowers fput who loserFollowers
                                    set detrust 0 ]
                       if detrust < followerLossTolerance [
                                  set detrust detrust + 1
                               ]][set detrust 0]]
  foreach loserFollowers [
          ask follower ?  [
              let myhub one-of link-neighbors with [breed = hubs]
              let refusedHub 0
               if myhub != nobody [ask myhub[ set refusedHub who ]
                          ask link-with myhub [die]
                          if count hubs with [ who != refusedHub] != 0[
                                         create-link-with one-of turtles with [
                                               breed = hubs and who != refusedHub]]]]]
end
```

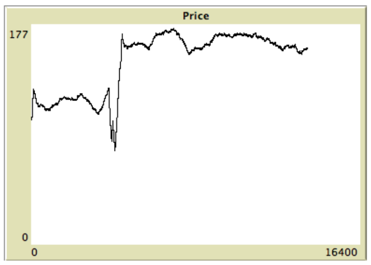
So we fixed $HLT = 600$, an enough high value to avoid any possible interference between *rejectHub* and *changeStrategy* effects. Then varying the *followerLossTolerance* parameter we discovered several different behaviors. By choosing $FLT = 7$ we had a strong intermittent herd behavior, highlighted by the consecutive very steep picks in a relatively "short" time.

Then passing to $FLT = 100$ we note how a so high threshold could hardly be reached by the followers *detrust* random variable, which strictly depends on volatility in price generated by *randomTraders* . So decreasing volatility with a strong trended behavior we will see again bubbles and crashes.

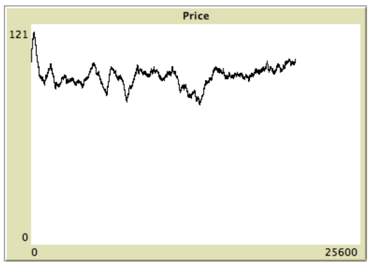
Finally choosing an intermediate value, not too far from the common variating range of the *detrust* variable, we obtain a much more realistic behavior with stochastic dynamic interrupted by unexpected bubbles or crashes.



I) followersLossTolerance = 7



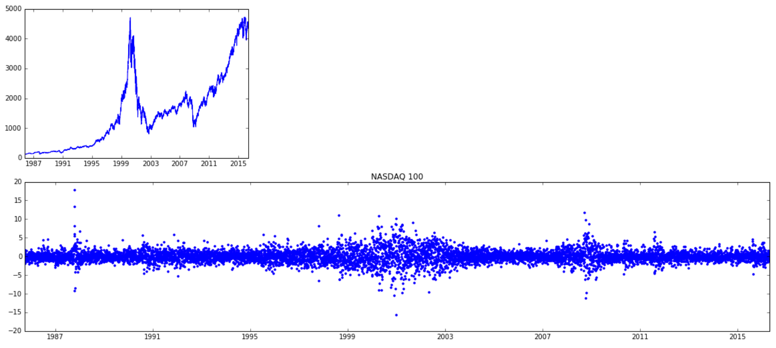
II) followersLossTolerance = 10



III) followersLossTolerance = 100

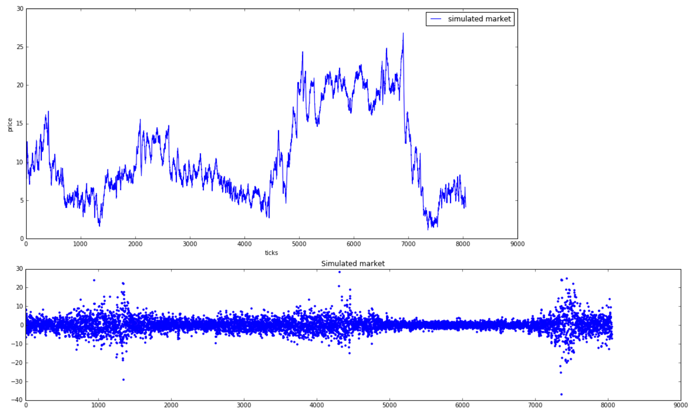
2.4 Comparing our results with those of a real market

In order to make a comparison between our simulated market and a real market behavior, we done that by choosing real data of the NASDAQ100 index time series from October 1985 to 2016, obtained from `finance.yahoo.com`. By these date we plotted first for price time series , and then by a scatterplot the time series of returns. We derived these plots using the *matplotlib* python library, in IPython which is a command shell for interactive computing in python language.



As can be seen from these two plots real markets returns follows a power law distribution, in fact the majority (about the 93%) of these 7710 data points lies in $[+5\%, -5\%]$ range of changes in returns, while there are few points (about the 1%) which reach 10% of changes in returns, with piques of 18%. This is a significant complexity feature present in lots of stock markets, and we want to show in future that many other measures can be performed to support of the markets complexity hypothesis.

Then we did the same procedure for our simulated market, storing the time series obtained for prices in a .CSV file, and in this way we could make the same kind of analysis also on our simulated data.



From these the two plots obtained by choosing a proportionality coefficient for price updating procedure $\gamma = 0.01$, without enabling quantities of stocks but just setting $FLT = 15$ and $HLT = 18$. We allowed followers agents to change their hub and hubs to change their strategy, and both these two effect combined give rise to markets complexity observed in the previous plots. This is just a first example of how much agent based models can be adequate for stock market simulating, reproducing complexity generated by social behaviors like the herding effect.

3 Conclusions

In this work we addressed study and modeling of stock market. And taking inspiration by several previous studies as B.LeBaron [1], it is possible to build an artificial model for stock market trying to reproduce all the so-called stylized facts, and the principal elements which lead price dynamics to assume a complex behavior.

But there are different ways to approach the problem. One is to use analytical formulas from classical financial theory, which considers agents behavior according to the rational markets hypothesis, updating pricing by adopting several portfolio strategies.

Such as in [1],where price is continuously updated at every time by the effect of all buyers and sellers, but this description doesn't correspond to a real behavior.

So we decided to address stock market modeling starting from the microscopical interactions between agents, that are dominated by the social dynamics such as trust relationships which lead to herding effect. And this dynamic is well described by hubs and followers paradigm, as also proved by comparison of real data with our simulated price time series . And we discovered that by varying parameters we can simulate market bubbles and crashes, because in presence of high price uncertainty many investors trust in few specialists causing herding effect.

The aim of this work was not to introduce in our model all elements of a real stock market, but to explain nature of social interactions which lead to those complex behaviors typical of real markets.

In further works we would like to program some hubs to perform specific forecasting techniques, as autoregressive techniques or moving averages, to compare their profits in this simulated market to those obtained in case of completely random price dynamic.

We also would like to study the case in which price is not updated by analytical solutions but using a real market book dynamic, in which traders can exchange only if their bid and ask prices meet.

In order to address other complexity measure we will estimate the heavy tail effect in cumulative distribution, the returns autocorrelation and volatility, as have been done by S.Bornholdt [4]. But we will also focus on searching of a dynamical attractor for stock market, studying time series by dimensionality reduction techniques in order to reduce noise.

References

[1] B.LeBaron, W.B.Arthur, J. R. P.: 1996, Asset pricing under endogenous expectations in an artificial stock market, *Working Papers - Santa Fe Institute* .

[2] LeBaron, W.B.Arthur, J. H.: 1999, An artificial stock market, *Artif Life Robotics* (3), 27.

[3] Pentland, A.: 2014, *Social Physics - How good ideas spread - The lesson from a new science*, The Penguin Press - a member of Penguin Group(USA).

[4] S.Bornholdt: 2001, Expectation bubbles in a spin model of markets: Intermittency from frustration across scales, *International Journal of Modern Physics C* **12**(5), 667–674.

[5] T. Lux, M. M.: 1999, Volatility clusterin in financial markets: a microsimulation of incteracting agents, *International Journal of Theoretical and Applied Finance* **3**(4), 675–702.