Charles University in Prague

Faculty of Mathematics and Physics

BACHELOR THESIS

Jiří Hanuš

Agent-based modelling in economics

Institute of Formal and Applied Linguistics

Supervisor of the bachelor thesis: Mgr. Jan Raab

Study programme: Computer Science, General Computer Science

2011

I would like to thank my supervisor for his patience and advice.

.

Název práce: Agentové modelování v ekonomii.
Autor: Jiří Hanuš
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí bakalářské práce: Mgr. Jan Raab, Ústav formální a aplikované lingvistiky
e-mail vedoucího: raab@ufal.mff.cuni.cz

Abstrakt: V této práci se zaměříme na jeden pohled na modelování v ekonomii - agentové modelování a výpočetní ekonomii založenou na agentech (Agent-based computational economics), což je výpočetně náročná metoda, která simuluje interakce mezi subjekty v ekonomice, místo aby se soustředila na hledání stabilních rovnovážných bodů. Představíme si úvodní myšlenky, které stojí za teorií komplexních systémů, která vysvětluje mnohé jevy, které můžeme v ekonomice vidět. Vysvětlíme, proč tyto jevy způsobují, že je pro ekonomy těžké vytvářet modely vysvětlující chování lidí ve skutečném světě. Ukážeme si jeden možný model interakcí mezi spotřebiteli a firmami a jeho implementaci, kterou se budeme snažit udržet v chodu po nějaký čas a ve stabilním, nicméně dynamickém stavu.

Klíčová slova: Agent, modelování v ekonomii, komplexní systémy

Title: Agent-based modelling in economics.
Author: Jiří Hanuš
Department: Institute of Formal and Applied Linguistics
Supervisor: Mgr. Jan Raab, Institute of Formal and Applied Linguistics
Supervisor's e-mail address: raab@ufal.mff.cuni.cz

Abstract: In this thesis, we present one view on economic modelling - Agent based modelling and Agent based computational economics, which is computationally intensive method that simulates interactions between entities in economy instead of focusing on stable equilibria. We will provide introduction into Complex systems theory, which explains many phenomena we can see in economies and explain why these phenomena make it hard for economists to design models explaining behavior of people in real world. We will show one possible model of face-to-face interactions between consumers and firms and its implementation, where we can see whether it can sustain for a period of time in dynamic but stable state.

Keywords: Agent, modelling in economics, complex systems

Contents

Chapter 1

Introduction

Economics – “a social science that analyzes the production, distribution, and consumption of goods and services” (Wikipedia).

At first, it was an area of study firmly standing among other humanities, but it soon began to incorporate elements of exact sciences, especially of mathematics. This shift gave birth to many controversies and disputes among economists and all people which somehow felt affected by their teachings.

People felt very affected after the last financial crisis which broke out in 2008 and many immediately began to question economics as a science and its trends. This mood in society was reflected by many articles in newspapers and books, for example [1]. The exact and mathematical nature of economics was once again in need to justify itself because of misunderstanding. Misunderstanding of people creating unsatisfactory economic models, for example credit risk models in banks [2], and misunderstanding of people using these models in applications where their authors never intended for them to be used.

The use of mathematical approach and modeling in economics increased with the invention of computers and computing machines and was used interestingly for example to help in economic planning in the Soviet Union [3].

One type of economic models, are the general equilibrium models, which are a tool of general equilibrium theory – theory, that "seeks to explain the behavior of supply, demand and prices in a whole economy with several or many interacting markets, by seeking to prove that a set of prices exists that will result in an overall equilibrium" (Wikipedia).

The growth of computer capabilities in the past years paved the way for an entirely new approach – the Agent-based Computational Economics (ACE). It studies economic processes as dynamic systems of interacting agents (autonomous entities pursuing their goals), rather than equations. These agents do not have to have perfect information or rationality. Hence this approach does not use equilibria, which do not have to be maintained at all, but focuses on processes and phenomena that occur when agents interact.

When using equation models, all the complexity of economy must be directly embedded in the equations, therefore even when modeling relatively simple economies, the equations are relatively complex. However, when using agent-based modeling, the agents are relatively simple – they have to be, since subjects in the economy (and in the world) aren't all knowing and all powerful. The complexity of economy and its interesting properties arise from the numerous interactions between agents.

Chapter 2

Agents and complexity

The interesting properties of economies, arising from the interactions between simple agents, are manifestation of a phenomenon called emergence, which is studied by complex systems theory. These emergent properties can be very different from the properties of the individual agents.

System is regarded as complex, when it consists of connected but interdependent diverse entities and these entities adapt according to the changes in their environment.

They are interesting because of the aforementioned emergent phenomena, for example when system organizes itself in some kind of order. They often can not be predicted from the behavior of individual parts and this organization can be maintained even if some shock events happen to the environment. You can see from this description, that economies are such complex systems. By getting to know the properties, virtues and dark corners of complexity, one can properly appreciate the hardships any modeler faces.

2.1 Landscapes

In order to learn more about complex systems, we will use the idea of landscapes as a metaphor for the environment agents live in. This metaphor is used by Scott E. Page [5]. We will be interested only in the height of the point agent is in. Point is some state of the environment defined by the variables and assigned values. The height means how good the agent is doing.

Lets divide landscapes into three types

1. Simple landscape – one single hill with a top.

2. Rugged landscape – has many peaks with different height.

3. Dancing landscape – many peaks with different heights and the landscape changes in time.

Property if peaks is their height. Simple landscape has one peak, which is global peak – the optimum if we were searching for the highest one. Rugged landscape has many peaks – local optima – and one or more global peaks. It is usually computationally hard to find them, if the landscape is defined by some function or system of equations. Dancing landscapes are even harder (not that NP hard was not hard enough), if we want to find the global peak, since they change in time and what was global peak once can be local peak in the next moment or not a peak at all.

Simple landscapes

Problems, whose search space is simple landscape are simple to solve. Classical example is the optimal size of shovel explored by Frederick Winslow Taylor in the early 20th century [4]. With the increasing size of shovel the productivity of worker was increasing to a certain point and then decreasing.

Rugged landscapes

However, the number of problems that lead to simple landscapes is limited and optimizing the economy or personal profit in it is undeniably more complicated than that. The ruggedness of the landscape stems from the fact, that agent has usually more choices he can make at once and the outcome of one choice influences the outcome of second choice.

For example when agent has two possible types of goods to buy, the decision to buy one good and its consumption affects the happiness from the purchase and consumption of the other goods. The more interconnected choices agent has, the more rugged the landscape.

Dancing landscapes

This ruggedness, however, does not mean complexity as we have defined it. To get complexity, we need the landscape to become dancing. This happens, when there are other agents, that our "problem solving" agent has to take into account and their choices interact with his choices. In this way, agent has very limited time for decision and knowledge about what choice produces what outcome, because the outcomes are dependent on other agents' choices too.

For example, we have two firms producing some kind of good and they have to establish or improve their delivery chain. The first firm decides to expand in a place, where demand exceeds supply and that would seem like a reasonable action. This firm takes a lot of money hires many very clever people who work for a long time to tweak the delivery chain so that it would exactly match the expected demand at that place. After many resources spent, the deliveries in that area begin, only for the firm to realize, that the second firm had the same idea and already supplies in that area and the demand therefore dropped significantly. The landscape has changed, because of the other firm's (agent's) action. If the other firm was not there, it would be only a problem represented by rugged landscape (dependencies of choices are likely).

This landscape metaphor helps us to understand the fact, that sometimes systems can achieve optimal solution and sometimes can't.

2.2 Complexity arises

As we have previously said, the system is complex, if it consists of interdependent, connected, diverse and adaptive agents. But to which extent do these properties have to exert?

Interestingly, we can have too much of a good thing and at the extremes, complexity does not occur.

But to have better insight in what really is complex behavior and what is not, we divide behavior of systems into four categories.

1. **Stable single-point equilibria** – the classic example is the ball at the bottom of the bowl, standing still. If we push it in whatever direction, it comes back and settles into rest. This is usually the case of the aforementioned general equilibrium theory economic models, although they are somewhat more complicated than a bowl.

2. **Periodic orbits** – state of system changes, but in a periodic way.

3. **Chaotic behavior** – this is not just any chaotic behavior, but chaotic in a strict way defined by chaos theory. This behavior can be periodic and settle down to predictable path in state space which is called an attractor. State space is a virtual space where each possible state of the system is incidental to all states that the original state can become if one variable changes its value. Chaotic behavior is by definition very sensitive to initial conditions, as was discovered by Edward Lorenz. He has shown, that it is impossible to predict weather in the long run, because the weather system is so sensitive, that a flap of butterfly wings in Brazil can set off a Tornado in Texas (hyperbole) [6].

4. **Complex behavior** – behavior which is complicated, but has some inner regular structure and reason that arose from the interaction of parts of the system. This structure makes perfect sense, if the system is simple enough so that it can be described and comprehended. If the

system is not simple enough, it probably would make sense if we were smarter beings.

2.2.1 Interdependence

So when we get what behavior? Let's say we try the interdependence level. If the agents are independent, the behavior is not very interesting. If they are totally dependent on each other, we get chaos, because even a slight change causes an avalanche of changes throughout the system.

2.2.2 Connectedness

Next is connectedness. If agents are disconnected, they cannot observe their neighborhood and for example the level of interconnectedness is irrelevant. So this behavior is not interesting.

If connectedness is kept too low or too high, the system quickly converges to some equilibrium. However, with reasonable middle course of connectedness, interesting patterns begin to emerge, as noted for example in [7], [8]. In these papers, agents are let play rock-paper-scissors game connected in a network, players have fixed strategy (player always uses either rock or paper or scissors) and when player loses, he is replaced by a copy of the winning player. When players are intermediately connected (for example 3 connections), cyclic dominance of strategies is observed, i.e. strategy is dominant only for a while and changed by another in a periodic fashion. This is really observed in nature in Escherichia coli bacteria, which come in three types and fight with each other for space.

2.2.3 Diversity

The same stands for diversity. When there is not enough diversity, the system is simple. If there is too much, it is a mess with no interesting patterns at all. Really high diversity can survive mostly because of the moderate levels of connectedness each diverse type has – this can be found for example in nature, where there is very high diversity, but species are limited to their niche.

What is the cause of diversity in real life and in models? First, it is the interdependence and connectedness of agents – if one agent changes its behavior, the landscape for the other agents has changed and they have to act accordingly to survive. Other reason is, when there is, on the other hand, not enough pressure for selection and agents can become diverse with no fear of punishment for the lower efficiency they deliver [9].

2.2.4 Adaptation and learning

If we want to adjust adaptation, we adjust the intelligence of agents. If there is no adaptation, the system stays in equilibrium. If agents are all-knowing and super-intelligent, the system once again tends to stay in equilibrium – every agent finds its optimal place and behavior. Again, the best results lie in the middle, where agents are trying to figure out how to interact with the others to create complex system.

2.3 Exploration and exploitation

An ubiquitous problem in agent modeling is the trade-off between exploration and exploitation. By exploration, we mean the search for new and better behavior and solutions. Exploitation is, when agent wants to stay in place and take advantage of the found behavior or solution. For example, agent can ask himself the question “Should I still buy goods from this company, or does some other company offer better prices. And if not, will it always be like this in the future?” It is obvious, that agent must somehow find balance between trying new things and sticking with old, even with respect to time.

A classic problem where this phenomenon is studied is the two-armed bandit. It is a variation of a traditional slot machine. It has two levers and each lever provides a reward according to some probability distribution of that lever. The person or agent pulling the levers is faced with the explore/exploit problem, because he wants to pull only the lever that pays more money. How to determine, which lever is better? Many strategies have come up through time. Some of the simpler ones [10]:

- Epsilon-greedy strategy – simple greedy strategy would use only the lever, which has been better so far. This strategy still chooses the worse lever with probability ϵ and the better lever with $1-\epsilon$. This way, it the algorithm misjudged, the better lever now considered worse, may earn its proper appreciation, because it is still given a chance to earn more.
- Epsilon-first strategy – if there is fixed amount of trials T, agent earmarks his first ϵT trials for exploration (random lever pulled) and the rest, $(1-\epsilon)T$ for exploitation of the better lever.
- Strategy similar to epsilon-greedy, but the value for ϵ decreases in time.
- Strategy similar to that above, but epsilon decreases quickly if agent observes big changes in the expected distributions of the payoff of the levers. This allows agent to explore. Once the expected values of distributions do not change much, epsilon drops down to allow exploitation.

This model of two-armed bandit can be used even in economics. If firm starts producing some good and does not know its demand curve, it can try to set some prices according to some strategy and observe the profit the prices generated to determine the best price [11].

2.3.1 Simulated annealing

Now we'll explore more general algorithm, dealing with the trade-off between exploration and exploitation and we will illustrate it using our concept of landscapes. We want to get as high on the landscape, as possible, with limited time. If we make a step only if it takes us higher (exploit), we would quickly become stuck in some local optimum. This outcome happens usually when we try to program some greedy algorithm into our optimizing agent.

Much better approach is some algorithm that both explores and exploits. Typical representative is the simulated annealing algorithm [12]. Annealing (not simulated) is a process used by metallurgists and in materials science, used for altering the material by heat to change its properties such as strength and hardness. This is achieved by aligning all particles in the material in one direction. If metal is slightly heated, the previously randomly aligned particles align with their neighbors, which creates local neighborhoods with equally aligned particles. Then the temperature is cooled down gradually, so that the boundaries between neighborhoods cause some neighborhoods to switch particle alignment and assimilate into other neighborhoods. After some time, all particles are in one direction.

In modeling, this approach is used similarly. The temperature, which is higher at the beginning and gradually drops, is represented by willingness to explore and hence make mistakes. So at the beginning of simulation, the agent explores a lot. On a landscape, that means he moves both up and down. With time, the temperature goes down and the agent starts to exploit more. This means he goes only up the hills in the landscape and eventually stops at some local optimum. It is only local optimum and it is not secured that it is the global one; nevertheless it is a good solution. The rate at which the temperature falls depends on the ruggedness of the landscape.

Example of a landscape where simulated annealing can be used is the landscape represented by Rastrigin function [1] in figure 2.1.

For sample implementation used for optimizing cost function, see [14].

[1] Rastrigin function [13] is often used as a benchmark for the efficiency of optimization algorithms. It is defined for example like $f(x) = 10n + sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i)), -5.12 <= x_i <= 5.12$. where n is the number of dimensions

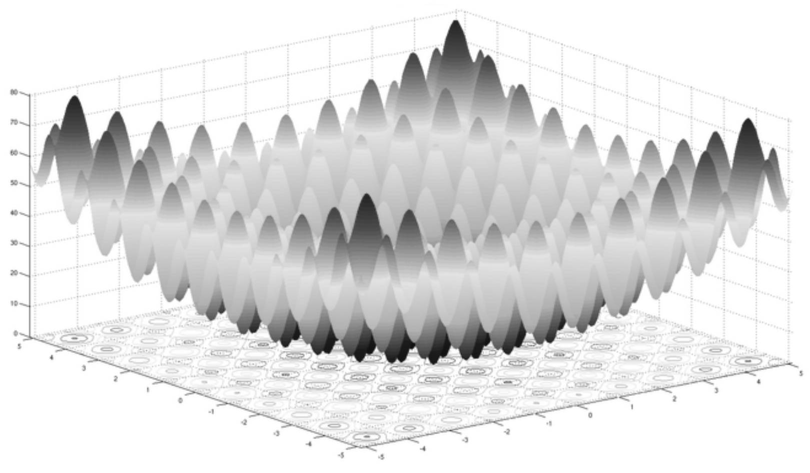


Figure 2.1: Rastrigin function

What about dancing landscapes? When the peaks in the landscape move, agents can not only exploit, they have to maintain some residual amount of exploitation. This exploration itself makes the landscape move for other agents, since one agent's action is connected with another agent's outcome (thanks to interconnectedness). These actions are, however, not random, but optimizing, which causes the emergence of complexity instead of chaos.

2.4 Emergence

Emergence means a spontaneous functionality and order, which was not imposed on the system by modeler or some central planner (in the case of real world) but was created from the bottom up [15] by the individual agents.

Examples of emergence in nature are self-organizing schools of fish or flocks of birds. In economics, fine example is the emergence of stock market behavior that regulates the prices of securities, although investors have limited knowledge about only some companies. But as a whole, this system works and usually produces equilibrium prices and patterns studied by analysts.

2.4.1 Types of emergence

Emergence can be divided into two types – weak and strong.

- Weak emergence – if the emergent property is deducible from its individual parts, even if the property was unexpected.
- Strong emergence – if the emergent property is not deducible from its individual parts.

The fine line between these two types is not set precisely and some scientists argue, that they are the same, because if person is sufficiently intelligent an knowledgeable, he is able to deduce anything from its parts; with the exception of consciousness, which is the only example of strong emergence. [16]

2.4.2 Cellular automata

For one very simple example of emergence, we will use the concept of cellular automata Cellular automaton is a discrete model which consists of grid of cells. Grid can have arbitrary number of dimensions. Each cell is in some state – in our Figure 2.2, there are only two states, ON and OFF and the grid is two dimensional. Each cell has some defined neighborhood, which consists of cells in some distance from the original cell. In our example, we will define the neighborhood as consisting of the 8 cells around the original cell.

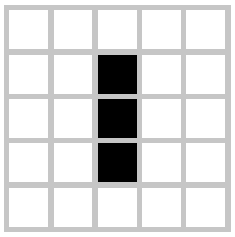


Figure 2.2: Blinker in state 1.

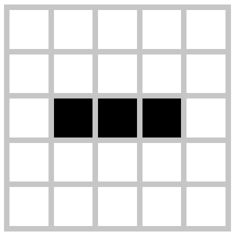


Figure 2.3: Blinker in state 2.

The cells change their state in time, all at once, according to rules. These rules determine the state of the cell in the next step only with dependence on the state of neighboring cells in the last step.

Our rules will be:

1. If cell is OFF and three neighboring cells are ON, cell swithces ON.
2. If cell is ON and two neighboring cells are ON, cell stays ON.
3. Else cell is OFF in next step.

If we start with simple grid looking like the one in Figure 2.2, where the ON cells are black, we get 2.3. In the next step, the system goes back to state in Figure 2.2. This means, that from simple rules, blinking behavior emerged. Much more complex behaviors, with simple rules, were devised in the Game of Life system by John Horton Conway.[2]

[2]Game of Life is a cellular automaton devised by John Horton Conway in 1970 and described first in [17]. More information can be found for example in [18]. It has four rules, different frou ours, and shows very interesting emergent behavior and patterns, like the “gun” pattern, which repeatedly shoots out moving patterns. One pattern that can be shot is the “glider”, which can be used to construct logic gates (AND, OR, NOT) in the system, making it Turing complete [19].

Chapter 3

Agent-based models

We will start with a few definitions:

- Agent (in economics) is the unit which acts and makes decisions in the model.
- Agent (in computer science) – autonomous entity which uses perceptors for sensing the environment it lives in and actuators to make actions affecting the environment.
- Agent-based model – computational model with agents following rules, pursuing their goals and taking actions, whose effect on the environment is then studied.

Agent-based models are computer models, that are used for studying complex systems (like economies). The dependence between complex systems and agent-based modeling is two-way – agent-based models help us understand some problems arising in complex systems, but they also create new problems to study by themselves.

We have seen simple agent-based model in section 2.4.2. There, individual cells were agents, because they were independent entities behaving according to some rules and through their behavior they affected the environment - other cells.

Agent based models can be used for example to study evacuation from buildings – how people behave and where the potential bottlenecks of successful evacuation are [20].

3.1 Walrasian equilibrium model

When modeling competitive markets, economists still widely use the equilibrium model invented by Leon Walras. [1]

This equilibrium model consists of

- Firms producing some good or service and maximizing their profit.
- Consumers maximizing some utility function.

Prices of goods are then set by walrasian auction, where all consumers compute their demand for all possible prices of the good and the price is then set so that the amount of good demanded equals the amount of good supplied by firms [2]. This has, however, some properties that are usually assumed by economists, but are rarely seen in real world – perfect information, no transaction costs and no strategic behavior of consumers or firms, trying to "outwit the others. Consumers are simple passive price-takers with no dependence on the choice of other consumers. The same problem stands for firms.

Moreover, the process of setting the right prices is much more complicated in real world and involving agent interactions. Such processes seldom end in stable equilibria and really, as shown for example by Fisher in [22], there are very specific conditions that have to be met for the walrasian equilibrium to exist. And even though these conditions are met, the equilibrium does not have to be unique or stable.

[1]French mathematical economist living 1834 - 1910, who is considered the creator of general equilibrium theory, as introduced in chapter 1.

[2]More information on how the auction works in detail can be found in [21]

If we remove the walrasian auctioneer, we must now take into account the interactions between individual agents [23], that cannot be described analytically.

3.2 Simulation

The simulation model described here tries to model face to face interactions of firms and consumers on single market instead of using the Walrasian auctioneer construct for finding the clearing price and volume of goods. This kind of model was devised in [24] but never implemented, instead, it evolved into The Trade Network Game Lab (TNG Lab) [25]

3.2.1 Economy

There are two types of firms, one selling abstract good A, the other good B. Each firm can produce only one type of goods. Then there are consumers, who buy goods A and B. Each firm in period T = 0 starts with an amount of money and production capacity.

Each firm has a total cost function that includes fixed costs proportional to its current capacity. Each firm knows the number of firms and consumers currently in the economy. However, no firm has prior knowledge regarding the income levels and utility functions of the consumers or the cost functions and capacities of other firms.

Each consumer in period T = 0 has fixed regular salary (paid from the outside) and a utility function measuring preferences and subsistence needs for goods in each period.

Each consumer is also a shareholder who owns an equal fraction of each firm. The income of each consumer at the beginning of period T = 0 is entirely determined by his money endowment. At the beginning of each period, each consumer's income is determined in part by his money endowment, in part by his savings from previous periods, and in part by his newly received dividend payments from firms.

At the beginning of each period, each firm selects a supply offer consisting of a production level and a unit price. Each firm uses a learning method to make this selection, conditional on its profit history and its cost attributes. This posting is carried out simultaneously by all firms, so that no firm has a strategic advantage through asymmetric information.

At the beginning of each period, each consumer costlessly acquires complete information about the firms' supply offers as soon as they are posted. Consumers then attempt to ensure their survival (subsistence needs) and happiness (amounts above subsistence needs) by engaging in a price discovery process consisting of successive rounds. During each round, the following sequence of activities is carried out:

First, any consumer unable to cover his currently unmet subsistence needs at the currently lowest posted prices immediately exits the price discovery process. Each remaining consumer determines his utility-maximizing demands for goods conditional on his currently unspent income, his currently unmet subsistence needs, and the currently lowest posted prices. He then submits his demands to the firms that have posted these lowest prices. Next, the firms receiving these demands attempt to satisfy them, applying if necessary a rationing method. Consumers rationed below subsistence need for one of the goods can adjust downward their demand for the other good to preserve income for future rounds. Finally, actual trades take place, which concludes the round. Any firms with unsold goods and any rationed consumers with unspent income then proceed into the next round, and the process repeats.

This period-T price-discovery process comes to a halt either when all firms are stocked out or when the unspent income levels of all consumers still participating in the process have been reduced to zero. Consumers who exit or finish this process with positive unmet subsistence needs die at the end of period T. Their unspent money holdings (if any) are then lost to the economy, but their stock shares are distributed equally among all remaining (alive) consumers at the beginning of period T+1. This stock share redistribution method ensures that each alive consumer continues to own an equal share of each firm. At the end of each period T $\geq$ 0, each firm calculates its period-T profits. A firm incurs positive (negative) profits if it sells (does not sell) enough output at a sufficiently high price to cover its total costs, including its fixed costs. Each firm then calculates its period-T net worth (total assets minus total liabilities). If a firm finds it does not have a positive net worth, it is declared insolvent and it must exit the economy. Otherwise, the firm applies a state-conditioned profit allocation method to determine how its period-T profits (positive or negative) should be allocated between money savings, capacity investment or reduction, and dividend payments to its shareholders.

3.2.2 Mathematical equations and processes

Utility function of consumer is defined as

$$Utility_c(a,b) = (a - subNeeds_{a,c})^{\frac{\alpha}{\alpha+\beta}} * (b - subNeeds_{b,c})^{\frac{\beta}{\alpha+\beta}}$$

where a and b are amounts of bought good A and B respectively, $subNeeds_{a,c}$ and $subNeeds_{b,c}$ are respectively the subsistence needs for goods A and B of consumer c and α is preference of A and β preference of B of consumer c.

Fixed costs of firm j in period T are

$$FixedCosts_j(T) = f_j * capacity_j(T) + F_j$$

where f_j and F_j are some given constants and $capacity_j(T)$ is capacity of firm j in period T.

Total costs of production of firm j in period T are

$$TotalCosts_j(T) = R_j * productionLevel_j(T)^2 + S_j * productionLevel_j(T) + FixedCosts(T)$$

where R_j and S_j are given constants and $productionLevel_j(T)$ is the level of production of firm j in period T.

Profit of firm is defined as

$$Profit_j(T) = soldGoods(T) * priceOfGoods(T) - TotalCosts_j(T)$$

where $soldGoods(T)$ is the amount of goods sold in period T and $priceOfGoods(T)$ is unit price for goods in period T.

Net worth of firm j is calculated as

$$NetWorth_j(T) = money_j(T) + capacityPrice * productionCapacity_j(T) + Profit_j(T)$$

where $money_j(T)$ is money firm has at the beginning of period T, $capacityPrice$ is the price for one unit of capacity (firm can buy or sell it for $capacityPrice$) and $productionCapacity_j(T)$ is the production capacity of firm j in T. If firm has negative net worth, it leaves the economy.

Capacity investment

If firm produces at the maximum capacity and sells everything, it allocates some part of profit into capacity investment and from the rest it pays part as dividends to consumers. The rest of money stays with the firm. Money in period T+1:

$$\begin{aligned} &Money_j(T+1) = \\ &1 - dividendAllocation_j) * (Money_j(T) + (1 - capacityAllocation_j) * Profit_j(T)) \end{aligned} \tag{3.1}$$

$$\begin{aligned} &Capacity_j(T+1) = \\ &\qquad Capacity_j(T) + capacityAllocation_j * Profit_j(T)/capacityPrice \end{aligned} \tag{3.2}$$

$$\begin{aligned} &Dividends_j(T+1) = \\ &dividendAllocation_j * (Money_j(T) + (1 - capacityAllocation_j) * Profit_j(T)) \end{aligned} \tag{3.3}$$

Where $dividendAllocation_j$ is part of money to pay as dividends and $capacityAllocation_j$ is part of profit to invest in capacity. If profits are nonnegative, but not all was sold, all profits go to money holdings and dividends are paid.

Learning

Each firm at the beginning of period decides what amount of goods to produce and for what unit price. Amount of goods to produce, called production level, must be smalled thar production capacity of firm. Price is chosen indirectly through markup, which is

$$Markup = (price - MarginalCosts)/price$$

where $MarginalCosts$ are marginal costs of current production.

Firm can try to increase its profit by increasing markup and production level (and capacity).

Learning is implemented as reinforcement learning - tendency to implement an action should be strengthened (reinforced) if it produces favorable results and weakened if it produces unfavorable results.

A firm j can choose from $NumOffers_j$ feasible supply offers in the form of $(productionlevel, markup)$ in each period. In the initial period T = 0, the initial propensity of firm j to choose its feasible supply offer is given by a nonnegative initial propensity $propensity_{ji}(0), \qquad i = 1, ..., NumOffers_j$ These initial propensities are assumed to be equal valued.

After that first period, the choice probability that firm j uses to select particular offer i is given by

$$probability_{ji}(T) = \frac{exp(propensity_{ji}(T)/cooling_j))}{\sum_{k=1}^{NumOffers_j} exp(propensity_{jk}(T)/cooling_j))}$$

where $cooling_j$ is a cooling parameter that affects the degree to which firm j makes use of propensity values in determining its choice probabilities.

As $cooling_j \rightarrow \infty$, then $probability_{ji}(T) \rightarrow (1/NumOffers_j)$ for each i, so that in the limit firm j pays no attention to propensity values in forming its choice probabilities. On the other hand, as $cooling_j \rightarrow 0$, the choice probabilities become increasingly peaked over the particular supply offers i having the highest propensity values, thereby increasing the probability that these supply offers will be chosen.

At the end of each period T ≥ 0, the current propensity $propensity_{ji}(T)$ that firm j associates with each feasible supply offer i is updated in accordance with the following rule. Let i' denote the supply offer that was actually selected and posted for period T, and let $Profit_{ji'}(T)$ denote the profits (positive or negative) attained by firm j in period T following its actual choice of supply offer i'. Then, for each feasible supply offer i:

$$propensity_{ji}(T+1) = (1 - recencyParam_j) * propensity_{ji} + Response_{ji}(T)$$

$$i = i')Response = (1 - experimentParam_j) * Profit_{ji}(T)$$

$$i <> i')Response = experimentParam_j * propensity_{ji}(T)/(NumOffers_j - 1)$$

where the recency parameter$recencyParam_j$ acts as a damper on the growth of propensities over time. The experimentation parameter $experimentParam_j$ permits reinforcement to spill over to some extent from a chosen supply offer to other supply offers to encourage continued experimentation with various supply offers in the early stages of the learning process.

Firm j faces a trade-off in each period T between information exploitation and information exploration. This learning algorithm resolves this trade-off by ensuring continual exploration, typically at a declining rate. Firm j in period T does not necessarily choose a supply offer with the highest accumulated profits to date. Given a suitably small value for $experimentParam_j$, selected supply offers generating the highest accumulated profits tend to have a relatively higher probability of being chosen, but there is always a chance that other supply offers will be chosen instead. This helps to reduce the risk of premature fixation on suboptimal supply offers in the early stages.

Price discovery process

Price discovery process consists of a sequence of rounds. The process comes to a halt as soon as either all firms are stocked out or the unspent income levels of all consumers still participating in the process have been reduced to zero. The total amount of goods actually purchased by each consumer c during the course of the period-T price discovery process are denoted by $bought_{c,a}(T)$ for good A and $bought_{c,b}(T)$ for good B

Suppose at least one firm has not stocked out and that the currently unspent portion of $money_c(T)'$ of consumer c's period-T income is positive. Let $subNeeds'_{c,a}$ and $subNeeds'_{c,a}$ denote consumer c's current net subsistence needs for A and B, i.e., his basic subsistence needs net of any purchases he has made in previous rounds of the period-T price discovery process. Finally, let $lowestPrice_a$ denote the currently lowest posted price for A if any A firms are still posting supply offers, and similarly for $lowestPrice_b$.

Suppose all A firms have stocked out but at least one B firm has not stocked out. If either $subNeeds'_{c,a} > 0$ or $lowestPrice_b * subNeeds'_{c,a} > money_c(T)'$, consumer c exits the price discovery process. Otherwise, consumer c determines his demands as follows:

$$demandForA = 0; \quad demandForB = money_c(T)'/lowestPrice_b$$

Finally, suppose that at least one A firm and one B firm have not stocked out. If consumer doesn't have enough money to satisfy his subsistence needs at the current lowest prices, consumers exits.

If both A and B firms still supply, consumer maximizes his utility subject to his budget and subsistence constraints. Demand functions:

$$demandForA = subNeeds'_{c,a} + \frac{\alpha}{\alpha+\beta} *$$

$$* \frac{(money_c(T)' - (subNeeds'_{c,a} * priceOfGoods_a(T) + subNeeds'_{c,b} * priceOfGoods_b(T)))}{priceOfGoods_a(T)}$$

Where α and β are preference coefficients as in utility function above. Similar equation stands for demand for B.

Rationing

If firm isn't able to satisfy all the demand, it must ration provided goods among consumers. One possible rationing method: Random queue rationing: Given excess demand for my good, I first randomly order my current customers into a queue line. I then attempt to satisfy each customer's demand in turn, to the fullest extent possible. All rationed amounts offered to consumers must be nonnegative.

If consumer is not able to cover subsistence needs under rationing, then he must adjust the demanded amounts to save as much money as possible to try in next round.

Cases:

1. no rationing - what was demanded was also offered by firm and therefore could have been bought
2. all needs met under rationing - rationed amounts are purchased
3. one need not met - for example for B - consumer buys only net subsistence needs for A and rationed amount of B to save money
4. both needs are not met - consumer buys rationed amounts

At the end of the price discovery round, consumer updates unspent money holdings and net subsistence needs.

3.2.3 Implementation

Introduction to architecture

More details about implementation and architecture of the program, with

more easy-to-read and hyperlinked text can be found in the enclosed documentation
Program is implemented in Java and consists of 2 types of agents, represented as instances of the classes TradingWorld.Firm and TradingWorld.Consumer. These live in the TradingWorld.World, which holds the containers of all consumers and firms and also is responsible for the right flow of the program by calling firms' and consumers' trading methods.

Class World holds all instances of classes Consumer and Firm in HashSets. HashSets for Firms are two, TradingWorld.World.AFirms for firms selling A and the TradingWorld.World.BFirms for B. These two types of firms are not different in their instances, but are differentiated just by the different location World stores them. No firm knows what type of goods it is actually selling.

When firm chooses how much and for how much it will sell its goods, it uses the TradingWorld.PropensityManager class. Each Firm has one instance of its PropensityManager, where it stores the data needed for deciding which offer to choose. When firm chooses price and production level by calling the TradingWorld.PropensityManager.chooseOffer(int productionCapacity) method, it only sets TradingWorld.Firm.price and TradingWorld.Firm.productionLevel attributes in its instance. World them must extract the price from inside the firm, pack it together with reference of firm offering the price and create instance of TradingWorld.World.PriceOffer object holding these two pieces of information together. This PriceOffer object is then placed in one of the TradingWorld.World.APrices or TradingWorld.World.BPrices lists of price offers in world.

Consumers then ask World for the PriceOffer of one of the cheapest firms for trading and prepare their order.

Firms take orders from consumers in the form of instances of TradingWorld.Firm.Order class. Each consumer must therefore create an instance and pass it to firms to enter their queues. Consumer knows the reference to firm he wants to trade with, because he has unpacked it from

PriceOffer he got from the World.

Order of consumer contains not only amount of goods he wants to buy but also an instance of class TradingWorld.Consumer.OrderConfirmation, which belongs to ordering consumer. There the chosen firm responds to consumer's order and submits amount of goods consumer can actually buy. Consumer owns two instances of OrderConfirmation. One he provides to chosen firm selling A and the other to firm selling B. There he collects the amounts of goods offered by the firm and redecides the amounts if it needs to buy less to save money for later purchases, because his subsistence needs are not met.

The buying itself is performed by consumer and carried out through the references to firms in firm's PriceOffer, which contains the price for which firm sells and firm's reference.

When consumer wants to be removed from world, he notifies the world about his removal and passes the reference of itself to the world. World stores these references in TradingWorld.World.consumersToKill list of firms which want to be removed and removes them at the end of the round (removal is carried out through comparing ID's of consumers).

Firms who want to be removed are in a more complicated situation. If they pass their reference to the world, it wouldn't know what type of firm it is (selling A or B), because the firm itself does not know. Therefore each firm is provided (at the start) with a reference to TradingWorld.World.FirmArraysContainer, which the firm should use when submitting its removal. These containers are two in the world - one with some lists where firms selling A post some data, the other container for firms selling B. Firm can't do anything with this container reference, it only pases it back to the world when submitting removal and world adds this firm to collection TradingWorld.World.FirmArraysContainer.firmsForRemoval in the container. This collection is processed at the end of the round and all firms added there are removed from the world.

Similar mechanism is used when firms want to withdraw their offers from the list of PriceOffers when they are stocked out - uses TradingWorld.World.FirmArraysContainer.pricesForRemoval.

There is one more type of nested classes in Consumers and Firms, which are classes implementing the Runnable interface. Each of these classes has only one method - the run() method, which acts as a standard method of Consumer of Firm. But because it is wrapped in Runnable, it can run in a separate thread and use paralelization.

Parallelization

Some functions manipulating with Consumers and Firms are computationally intensive and every consumer or firm runs its own instance, manipulating only with the data stored in it. Therefore these functions can easily run in parallel, so that program can take advantage of multi-core processors.

Each of these functions is wrapped in a class implementing standard java Runnable interface and the implemented method run() is the parallelized function. These runnable classes, which are in fact just like regular functions. Some examples:

- TradingWorld.Consumer.EnterQueueRunnable
- TradingWorld.Firm.ChooseOfferRunnable
- TradingWorld.Firm.ProcessQueueRunnable

More detail about what particular runnable does is in the Detailed flow of the program in the enclosed documentation.

These Runnables are always started by the World, which ensures the parallelization by standard java facility for starting Runnables, which is java.util.concurrent.Executor and interface java.util.concurrent.ExecutorService, providing additional methods for control of Runnables.

Propensity manager

Propensity manager serves as learning mechanism for deciding which production level and markup firm (which owns its instance of propensity manager) should choose for production. Provides two main methods - TradingWorld.PropensityManager.chooseOffer(int productionCapacity) and TradingWorld.PropensityManager.updatePropensities(int productionLevel, byte markup, double profit). ChooseOffer method returns some production level and markup according to the embedded learning mechanism. UpdatePropensities updates the mechanism according to profit the previously chosen offer generated.

Propensity manager stores propensities in TradingWorld.PropensityManager.offerMap, which is a TreeMap<Integer, ProductionLevelNode>indexed by production level and storing TradingWorld.PropensityManager.ProductionLevelNode. This production level node consists of list TradingWorld.PropensityManager.ProductionLevelNode.markups , which stores instances of TradingWorld.PropensityManager.MarkupNode, which store propensity to the given combination of production level and markup.

Since at the beginning all the propensities are the same and have the uniform value assigned to them, there is no point in storing this value in each MarkupNode and all these MarkupNodes in ProductionLevelNodes. So each ProductionLevelNode contains also list TradingWorld.PropensityManager.ProductionLevelNode.uniformMarkups of markups, which have uniform propensity. These markups do not have their representation in TradingWorld.PropensityManager.ProductionLevelNode.markups.
Likewise, the ProductionLevelNodes which have all markup propensities uniform do not have to be in TradingWorld.PropensityManager.offerMap, so they are simply not present there. Also capacity of firms can be infinite,

which would mean infinite number of ProductionLevelNodes needed.

When chooseOffer is called, propensity manager first has to choose production level from offerMap and then it chooses some markup from ProductionLevelNode of that particular production level. First manager has to determine the probability of choosing each ProductionLevelNode. This probability is computed as a sum of all probabilities of choosing markups inside ProductionLevelNode. (Method TradingWorld.PropensityManager.ProductionLevelNode. getAltProductionLevelPropensity() is responsible for that.) This probability of choosing one markup (in production level) is computed from propensity of the markup as probability = Math.exp(propensity / coolingCoefficient) (see 3.2.2).

When those probabilities of production levels are determined (by method TradingWorld.PropensityManager.getNonUniformProdLevels(), the production levels are put into array, the first level is assigned its probability and all next production levels are assigned number computed as "probability of the particular level + the number assigned to the level before". This way the last production level has been assigned the sum of all probabilities. (Method TradingWorld.PropensityManager.transformAndGetMissing (ListAndPropensitySum<ProdLevelAltSummedPropensity>nonUniformProdLevels).)

This way, some production level can be chosen by generating random number (from the interval $< 0, number_at_last_production_level >$) and whichever level has summed probability closest to the random number, it is chosen.

One problem with this approach is, that some production level nodes are missing in offerMap. Those, whose markups have uniform propensity. The probability of choosing any of this production levels is the same and can be easily computed from uniform propensity.

Next, it is determined, if production level from uniform ones or nonuniform ones will be chosen - determined by random. If from non-uniform, method

TradingWorld.PropensityManager.chooseNonUniformProdLevel() is called, which choses one production level in the way described above. If uniform production level will be chosen from the list of uniform ones, it is chosen with uniform probability distribution from the array of integers representing production level, because the production level node does not yet exist. After choosing, the new instance of ProductionLevelNode is created and put into offerMap, since if it was chosen, it will be updated later with new propensities.

Markup from ProductionLevelNode is chosen by TradingWorld.PropensityManager.MarkupNode.chooseMarkup(), which uses similar approach as when choosing production level - transformation of propensities into probabilities, if uniform is chosen, it is removed from list of uniform and put in set of non-uniform.

Chosen production level and markup are passed to firm as instance of TradingWorld.PropensityManager.Offer.

When firm uses chosen production level and markup and makes some profit (of loss), it updates the propensities with its propensity manager by calling TradingWorld.PropensityManager.updatePropensities(int productionLevel, byte markup, double profit), which updates propensities according to 3.2.2.

Concluding results

The purpose of this simulation, as was suggested in previous chapters, is to see, whether such system can survive for some period of time. That is, if enough consumers and firms stay in the economy for successive rounds. This depends only on the parameters of the system, which can be tuned in the program.

The first idea about how to setup the program, would be to give everybody as much money as possible and little of costs. This works well for firms - they do not mind what they produce and at what cost, because it is always cheap for them to produce it and consumers are rich, so they buy everything. Propensity manager can be setup any way, because it does not matter what price and amount firm produces. However, this approach has one glitch. The consumers, which come first buy all the goods and so the consumers behind in the queue are not allowed to buy anything.

This is what it looks like in graphs and setup - figures 3.1, 3.2, 3.3, 3.4.

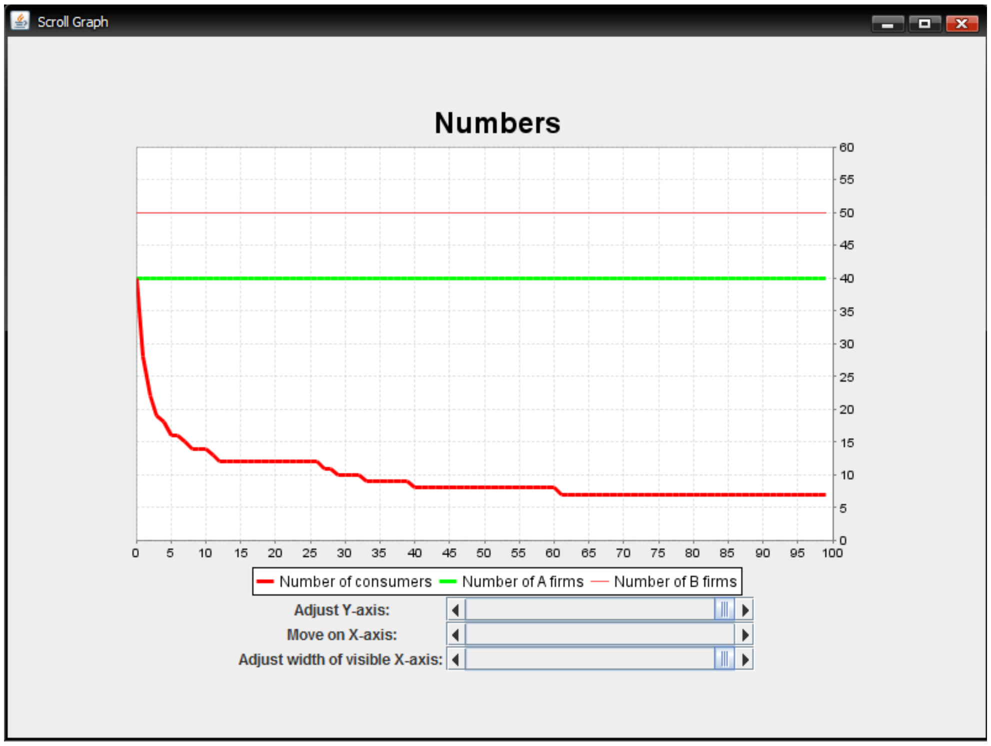
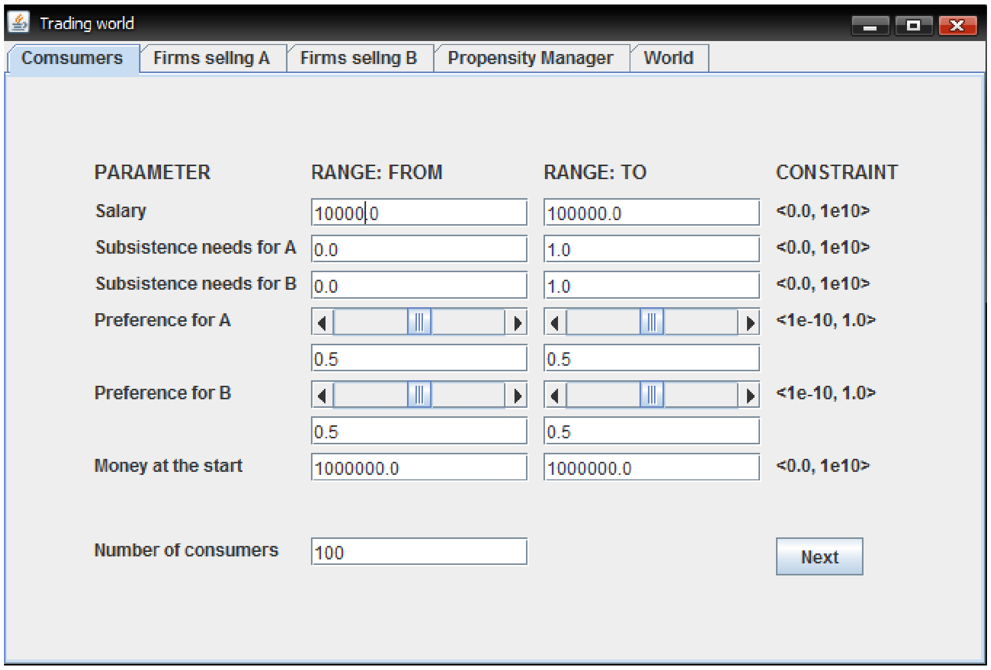


Figure 3.1: Numbers of consumers (bottom line) and firms (top 2 lines)

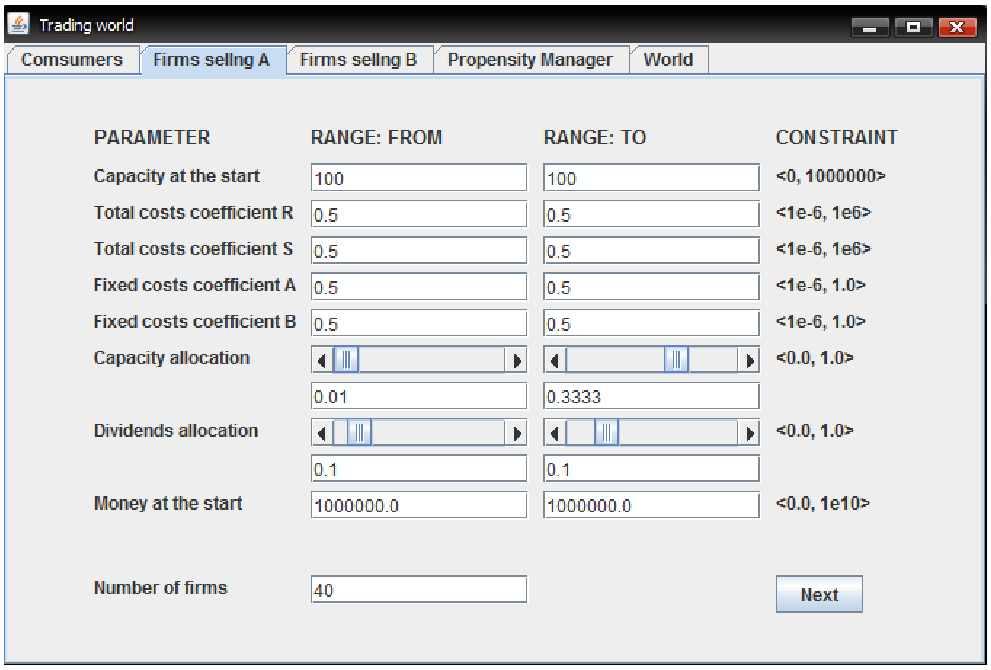
Both the types of firms thrive (stay on the same level, no firm exits the economy), whereas the consumers die quickly, until the strongest (or luckiest) stay, because the firms produce the right amount of goods just for them.

One thing we can do, is to give less money to consumers and firms. If we give only 1000 to both types of firms and 100 to consumers for beginning and 10 as salary, the world quickly ends, because when consumers spend all their starting money, they have almost nothing and die, taking firms with



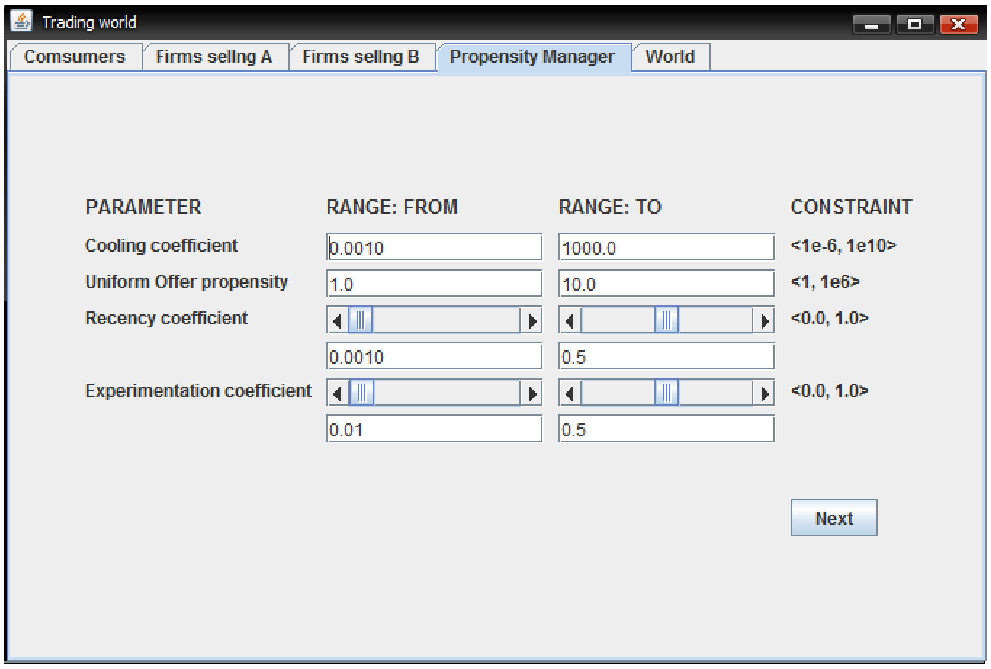
PARAMETER	RANGE: FROM	RANGE: TO	CONSTRAINT
Salary	10000.0	100000.0	<0.0, 1e10>
Subsistence needs for A	0.0	1.0	<0.0, 1e10>
Subsistence needs for B	0.0	1.0	<0.0, 1e10>
Preference for A	0.5	0.5	<1e-10, 1.0>
Preference for B	0.5	0.5	<1e-10, 1.0>
Money at the start	1000000.0	1000000.0	<0.0, 1e10>
Number of consumers	100		

Figure 3.2: Setup of consumer



PARAMETER	RANGE: FROM	RANGE: TO	CONSTRAINT
Capacity at the start	100	100	<0, 1000000>
Total costs coefficient R	0.5	0.5	<1e-6, 1e6>
Total costs coefficient S	0.5	0.5	<1e-6, 1e6>
Fixed costs coefficient A	0.5	0.5	<1e-6, 1.0>
Fixed costs coefficient B	0.5	0.5	<1e-6, 1.0>
Capacity allocation	0.01	0.3333	<0.0, 1.0>
Dividends allocation	0.1	0.1	<0.0, 1.0>
Money at the start	1000000.0	1000000.0	<0.0, 1e10>
Number of firms	40		

Figure 3.3: Setup of A firm (B similar)



PARAMETER	RANGE: FROM	RANGE: TO	CONSTRAINT
Cooling coefficient	0.0010	1000.0	<1e-6, 1e10>
Uniform Offer propensity	1.0	10.0	<1, 1e6>
Recency coefficient	0.0010	0.5	<0.0, 1.0>
Experimentation coefficient	0.01	0.5	<0.0, 1.0>

Figure 3.4: Setup of propensity manager

them - figure 3.5.

However, when we balance the money and give 1000 to firms as starting money and 100 to consumers as salary, the world can finally sustain on some interesting levels. Only 2 of B firms and 3 A firms survived, but there are also 20 satisfied consumers. And this ratio consumers : firms seems quite realistic - figure 3.6.

But other patterns emerge, when we set parameters differently. For example, when there are only small amount of firms left (lets say two) and one of them makes a big mistake when deciding price (large experimentation coefficient helps), it collapses the whole system. Consumers begin dying quickly, since half of the goods supplied is now gone. That means less profit for the one firm left at this moment and if firm does not react quicky by changing price and supply (which it can't, since it uses its propensity manager quite nonadventurously), it goes bankrupt too. See figure 3.7

We will show one more interesting scenario. We set consumers moderately rich, 1000 salary, with similar setting as previously. Firms have 1000 at the

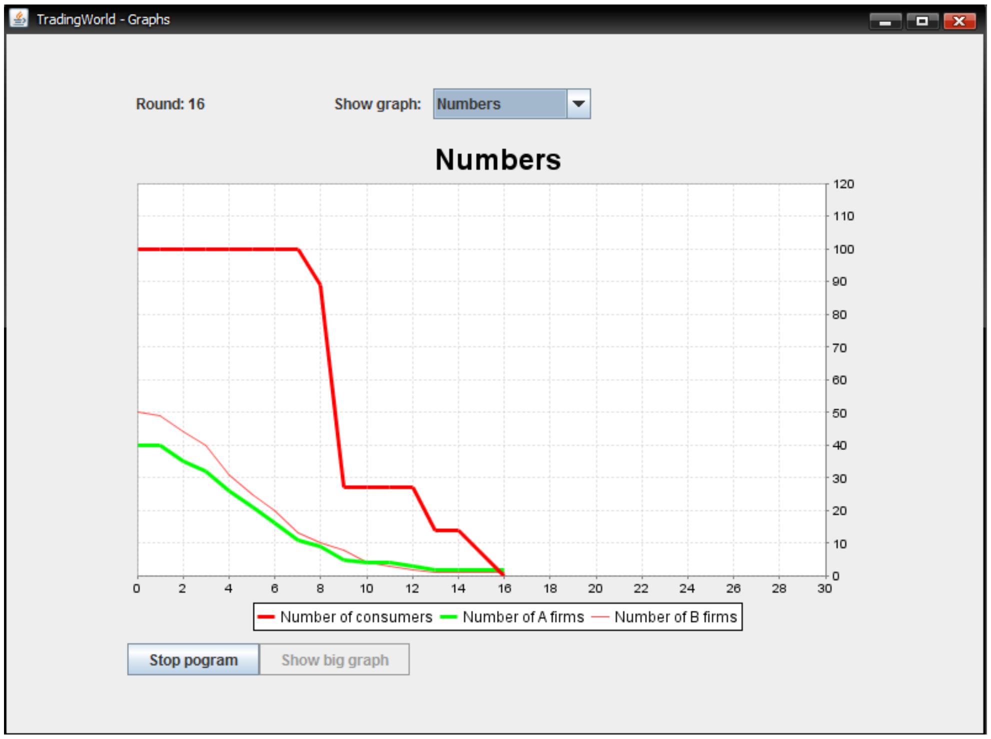


Figure 3.5: Numbers of consumers (top line) and firms (bottom 2 lines)

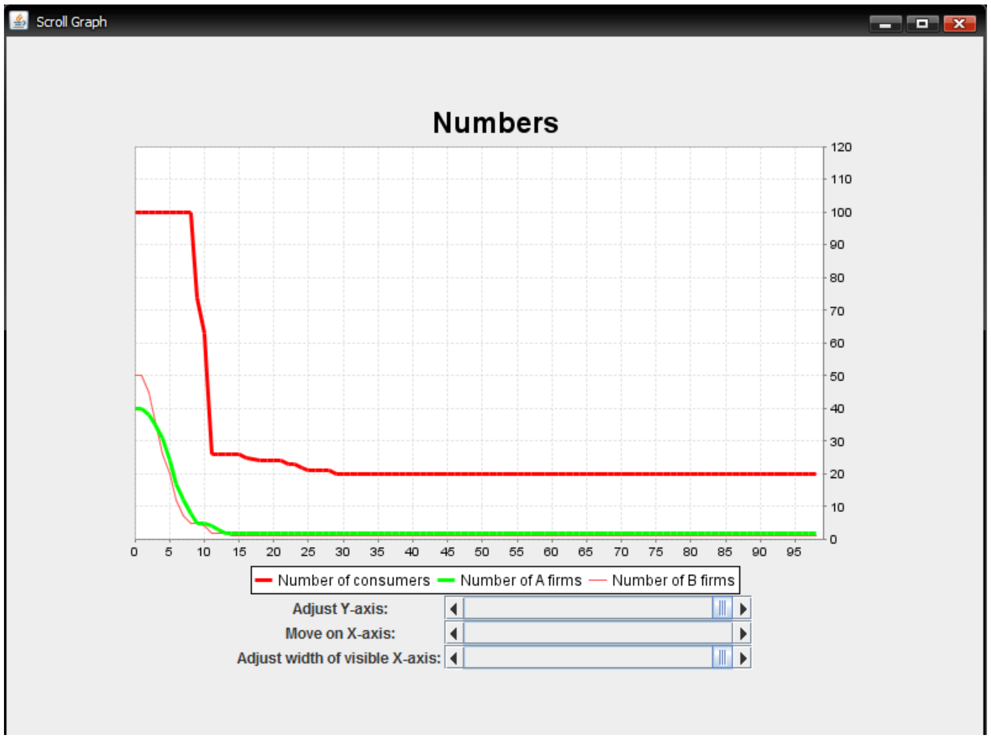


Figure 3.6: Numbers of consumers (top line) and firms (bottom 2 lines)

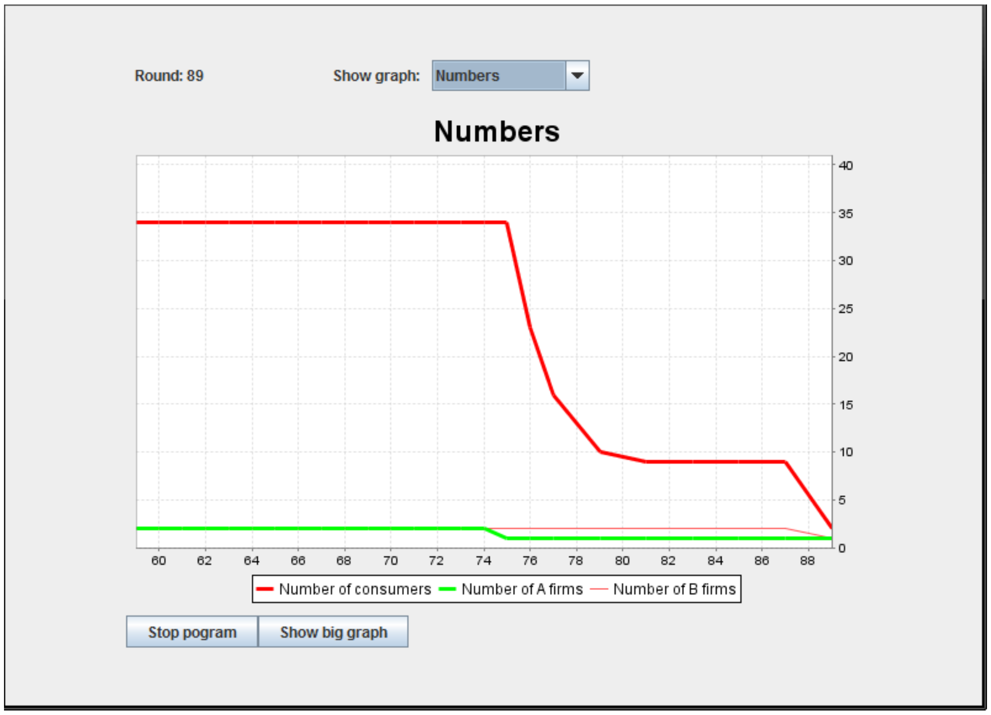
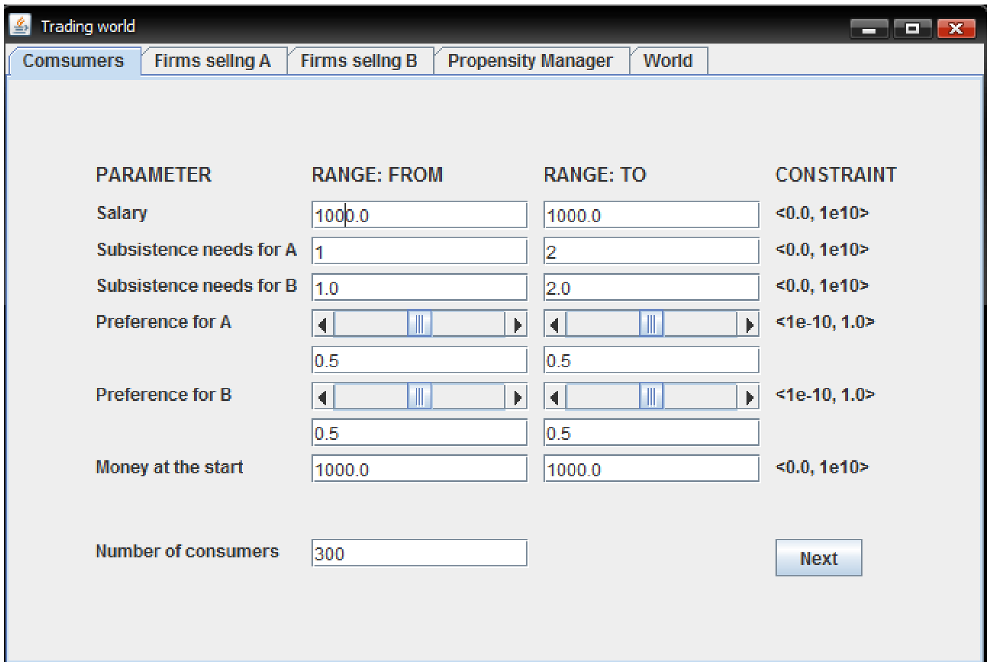


Figure 3.7: Numbers of consumers (top line) and firms (bottom 2 lines)

start too, else everything is normal - see figures 3.8, 3.9.

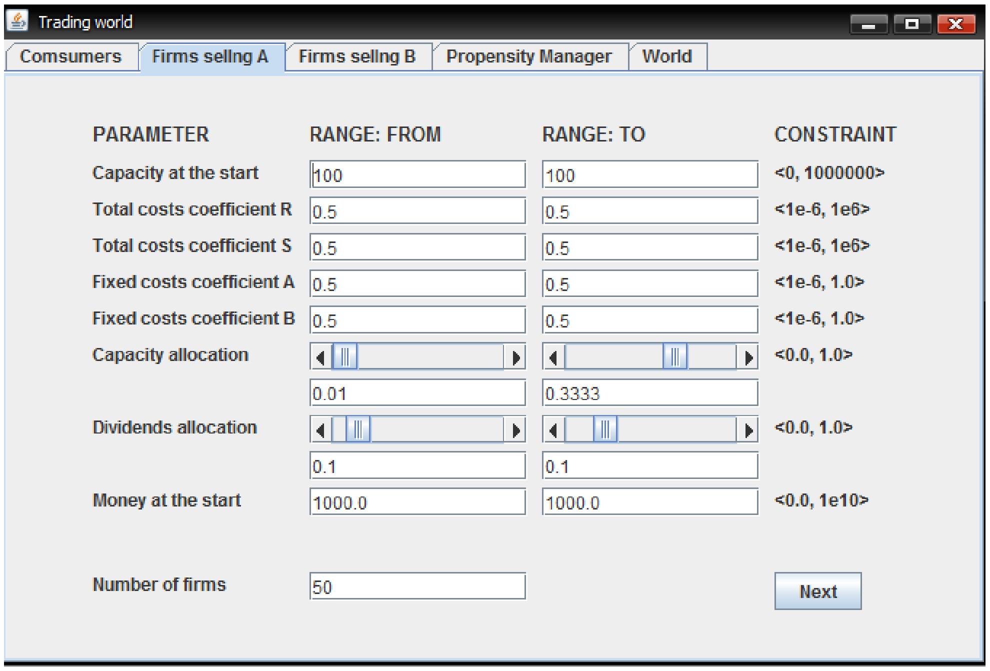
On the graph showing numbers of agents, we can see what seems to be "slowly declining efficiency" - figure 3.10.

If we look at the figure showing money of consumers, we see, that it is exactly zero for all consumers all the time. They are spending all they get - figure 3.11. This is different from the first case, where we had a few rich consumers and the rest dead - the histogram of their money showed a lot of money as reserves. So if consumers are spending all their money, what does it look like for firms? From their price and profit histograms, they have found very competitive price around which all firms oscillate and firms have also very stable profits distributed around zero, but more positive profits than negative ones. See figures 3.12, 3.13.



PARAMETER	RANGE: FROM	RANGE: TO	CONSTRAINT
Salary	1000.0	1000.0	<0.0, 1e10>
Subsistence needs for A	1	2	<0.0, 1e10>
Subsistence needs for B	1.0	2.0	<0.0, 1e10>
Preference for A	0.5	0.5	<1e-10, 1.0>
Preference for B	0.5	0.5	<1e-10, 1.0>
Money at the start	1000.0	1000.0	<0.0, 1e10>
Number of consumers	300		

Figure 3.8: Setup of consumer



PARAMETER	RANGE: FROM	RANGE: TO	CONSTRAINT
Capacity at the start	100	100	<0, 1000000>
Total costs coefficient R	0.5	0.5	<1e-6, 1e6>
Total costs coefficient S	0.5	0.5	<1e-6, 1e6>
Fixed costs coefficient A	0.5	0.5	<1e-6, 1.0>
Fixed costs coefficient B	0.5	0.5	<1e-6, 1.0>
Capacity allocation	0.01	0.3333	<0.0, 1.0>
Dividends allocation	0.1	0.1	<0.0, 1.0>
Money at the start	1000.0	1000.0	<0.0, 1e10>
Number of firms	50		

Figure 3.9: Setup of firm

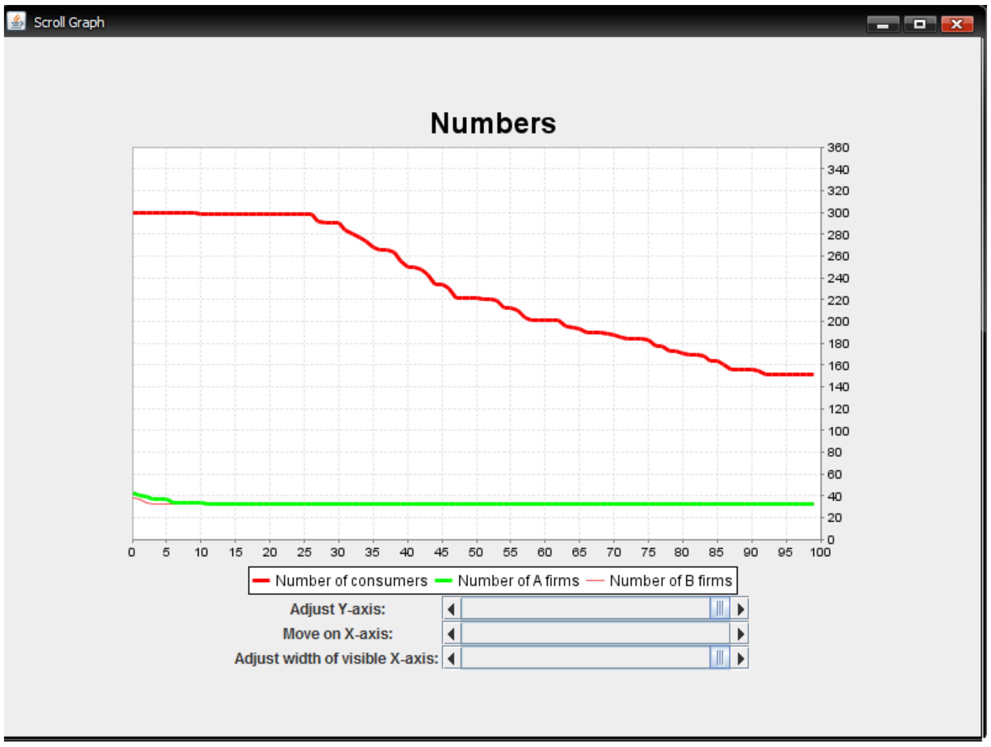


Figure 3.10: Setup of firm

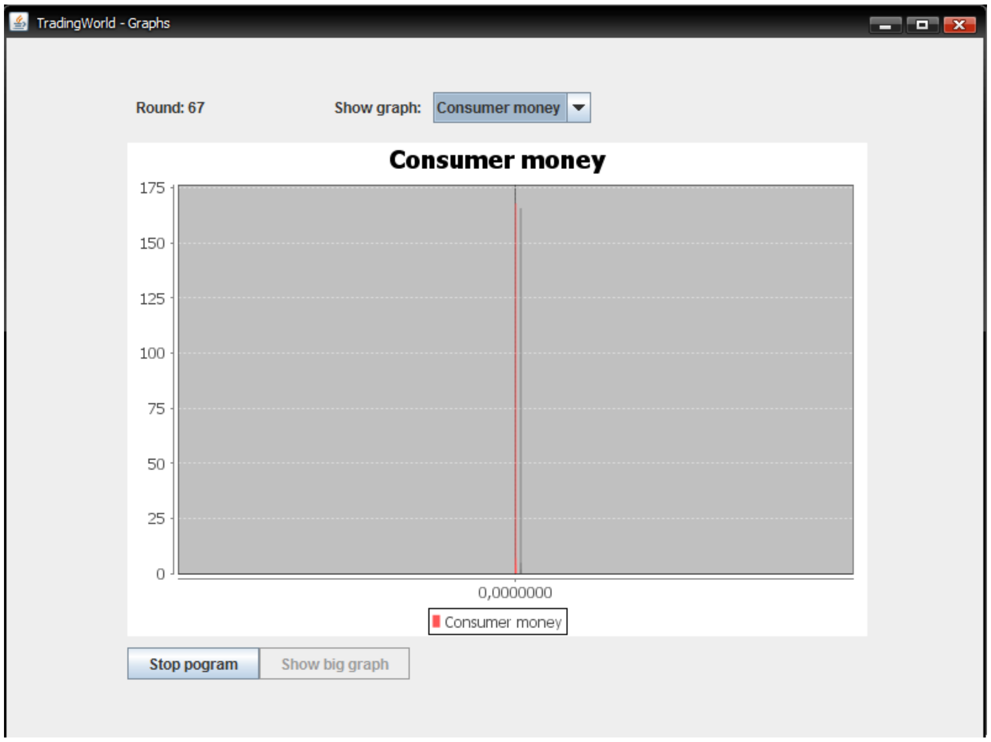


Figure 3.11: Setup of firm

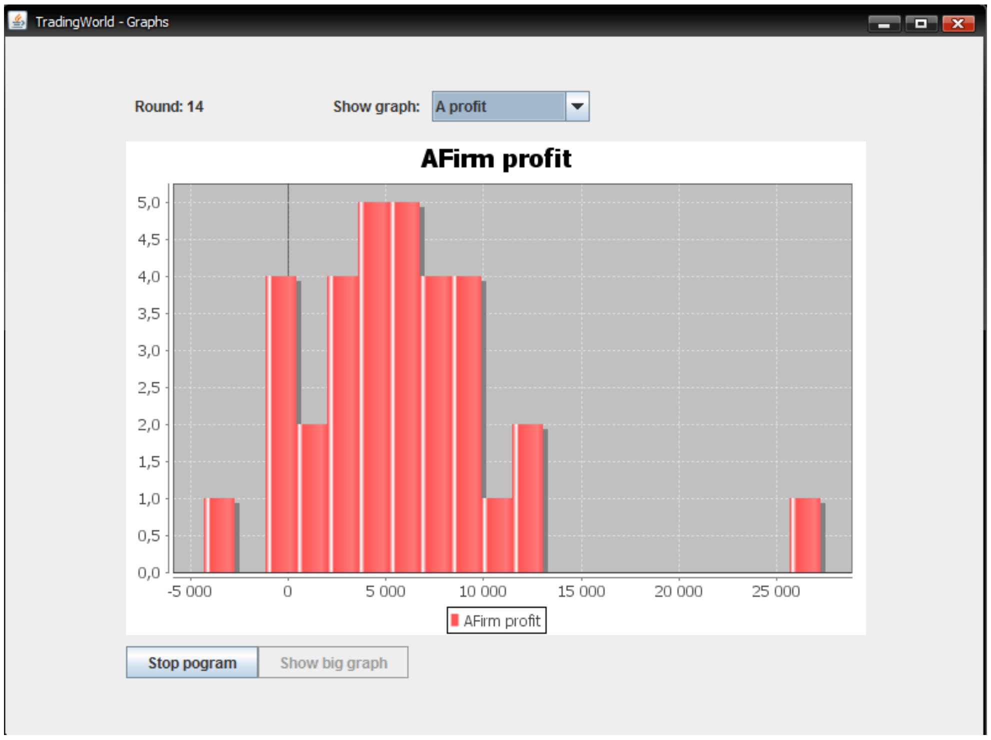


Figure 3.12: Setup of firm

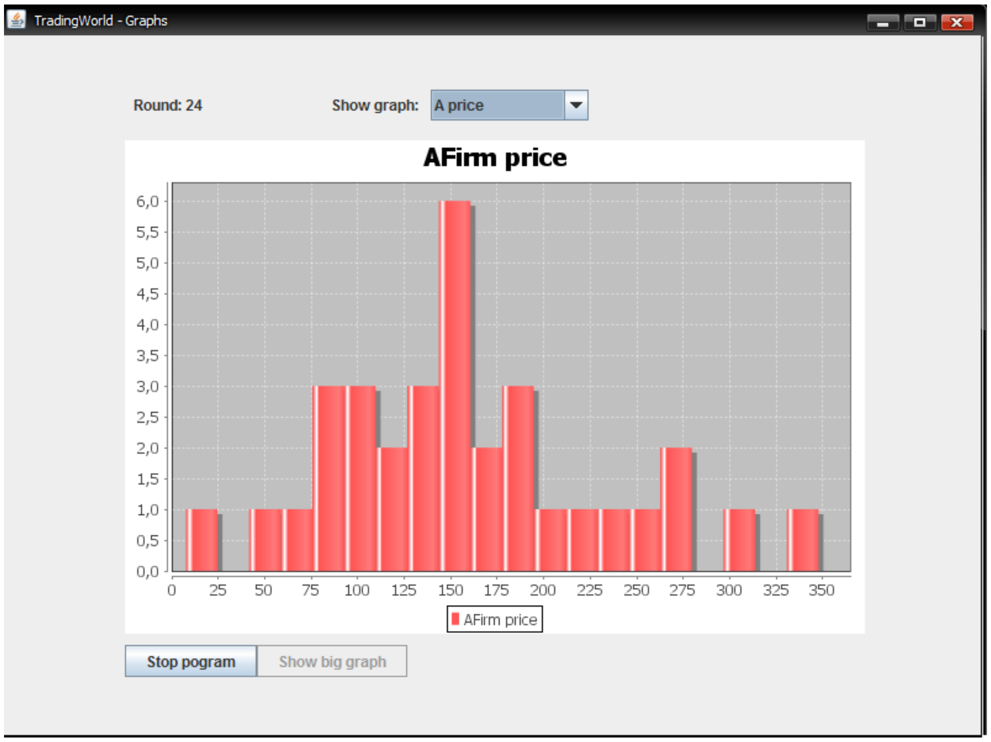


Figure 3.13: Setup of firm

Chapter 4

Conclusion

We have introduced to the reader the new approach in economic modeling. The Agent-based modeling in economics has bright future, since the computing power is increasing rapidly each year and even students can now project and implement sophisticated and computationally intensive models, which help us understand, what economy really is (or at least some of its smaller parts). This approach is not a substitute for classical modeling with differential equations and does not even want to be. It is merely a great complement, where classical models based on equations fail, because of the complexity of the problem or because the complexity of the solution. Since we have shown, that economy is a complex system with all its beauties and pitfalls, to have such a great tool at hand is really exciting. And we have shown, that a simple program with boundedly rational, simple agents, can be complex in its behavior and stable at the same time. Of course, we can't forget that these models are simplifications too and we can't expect miracles from them. But we can have some fun with them and possibly stumble across something of intriguing.

Bibliography

[1] Sedláček, Tomáš. *Ekonomie dobra a zla.* Nakladatelstí 65. pole, Praha, 2003.

[2] Turner, Jonathan Adair. *The Turner Review: a regulatory response to the global banking crisis.* Financial Services Authority, London, 2009.

[3] Fedorenko, N. *Econonico-mathematical models in Soviet economic planning and management.* UNESCO study, No. UNESCO/SS/3.244.1/h/34, 1966.

[4] Taylor, Frederick Winslow. *The Principles of Scientific Management.* Harper and Brothers, 1911.

[5] Page, Scott E. *Diversity and Complexity.* Princeton University Press, 2010.

[6] Lorenz, Edward. *The essence of chaos.* University of Washington Press, 1995.

[7] Keiko Kircher. *Emergent Behavior of Rock-Paper-Scissors Game.* unpublished paper, 2006.

[8] Kerr, B. and Riley, M. and Feldman, M.and Bohannan, B. *Local dispersal promotes biodiversity in a real-life game of rock-paper-scissors.* Nature Vol418, 2002.

[9] Page, Scott E. *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies.* Cloth, 2007.

[10] Vermorel, Joannes and Mohri, Mehryar. *Multi-Armed Bandit Algorithms and Empirical Evaluation.* Lecture Notes in Computer Science, Volume 3720/2005, p. 437-448, 2005.

[11] Rothschild, Michael. *A two-armed bandit theory of market pricing.* Journal of Economic Theory, vol. 9, issue 2, pages 185-202, 1974.

[12] Kirkpatrick, S. and Gelett, C. and Vecchi, M.P. *Optimization by simulated annealing.* Science 220, p. 621 - 630, 1983.

[13] Törn, A. and Zilinskas, A. *Global Optimization.* Lecture Notes in Computer Science, No. 350, Springer-Verlag, 1989.

[14] Bertsimas, Dimitris and Tsitsiklis, John. *Simulated annealing.* Statistical Science, Vol.8, No. 1, 10-15, 1993.

[15] Epstein, J. and Axtell, R. *Growing Artificial Societies: Social Science from the Bottom Up.* MIT Press, 1996.

[16] Chalmers, David. *Strong and Weak Emergence.* Essay in (P. Clayton and P. Davies, eds) *The Re-Emergence of Emergence.* Oxford University Press, 2006.

[17] Gardner, Martin. *"Mathematical Games"* column, Scientific American magazine, October 1970.

[18] Adamatzky, Andrew. *Game of Life Cellular Automata.* Springer, 2010.

[19] Berlekamp, E. R. and Conway, John Horton and Guy, R.K. *Winning Ways for your Mathematical Plays.* 2nd ed., A K Peters Ltd., 2004.

[20] Crooks, Andrew and Hudson-Smith, Andrew and Dearden, Joel. *Agent Street: An Environment for Exploring Agent-Based Models in Second Life.* Paper under review in the Journal of Artificial Societies and Social Simulation, 2011.

[21] Walker, Donald A. *Walras's Market Models.* Cambridge University Press, 2005.

[22] Fisher, F.M. *Disequilibrium Foundations of Equilibrium Economics.* Econometric Society Monographs No. 6., Cambridge University Press, 1983.

[23] Albin, P.S. and Foley, D. *Decentralized, dispersed exchange without an auctioneer: A simulation study.* Journal of Economic Behavior and Organization 18, 1992.

[24] Tesfatsion, L. and Judd, K.L. et al. *Handbook of Computational Economics, Volume 2.* North-Holland, 2006.

[25] McFadzean, D. and Stewart, D. and Tesfatsion, L. A *Computational Laboratory for Evolutionary Trade Networks.* IEEE Transactions on Evolutionary Computation, Vol. 5, No. 5, 2001.

.1 The list of files on the enclosed CD

1. Agent-based modelling in economics.pdf – this thesis
2. Program.zip – simulation program