MAGISTERARBEIT

Titel der Magisterarbeit

Comparison of Simulated Societal Structures in Agent Based Computational Economic Models

Verfasserin

Katharina Thelesklaf Bakk.

angestrebter akademischer Grad

Mag.rer.soc.oec

Wien, 2009

Studienkennzahl lt. Studienblatt: A 066 926

Studienrichtung lt. Studienblatt: Magisterstudium Wirtschaftsinformatik

Betreuer: Univ.Prof. Mag. Dr. Hardy Hanappi

Introduction

In one of the last semesters of my studies I attended a course called "Neuere Methoden formaler Modellierung". In this course we were dealing with a wide variety of computational simulations. My first leading thought was to write this thesis about simulations of pandemic phenomena. The topic, however, had to be more situated in the field of economics. Therefore we chose the wider field of economic simulations. We resolved to do a research in order to determine the influence of social structure which underlies the computational approach of simulations. After reading different papers about the many applications that you can find in economics I was able to choose one of them for conducting my research. My supervising tutors for this thesis also advised me to have a look on the homepage of Leigh Tesfatsion. There I could find various important things and also the executable program and all the classes of code for the chosen simulation. I just hope that some people are able to benefit from this collection of insights which have been given by other researchers and me.

It was much work and I needed some time to get it done. Therefore first I'd like to thank my parents for exercising patience and for their backing in all my years of studying. Next I have to thank my adviser for his help on where to begin, on the building of the work's structure and for the constructive feedback. I also must thank a frien of mine who corrected my flaws in the english language. Last I have to express my gratitude to the developer of the SimBioSys framework who gave some immediate assistance when I was facing problems with the simulation program. With the help of all those people I finally managed to finish this thesis and thus my master studies.

Abstract

This thesis' goal is to conduct and compare the results of agent based computational economic simulations and different structural conditions. Though there are connections/similarities between agents (agent programs) as understood in the computer sciences and the actual implemented simulated agents, these two concepts have different meanings – agents being the representation of thinking (to very different extents) and learning (modelled very differently). The complexity of these agents' thinking and learning patterns/schemes itself is a current research topic.
This thesis however is more concerned with the modelling of agents' interaction patterns and thus the inner workings of whole economies and societies – based on thinking agents. Current scientific literature on this topic discusses different bottom-up (agent based) approaches to different economic and sociological problems, their different sights stemming from different partner sciences (sociology, psychology, physics, computer sciences, law, policy sciences, etc.) and different research foci. These approaches heavily rely on computer based simulation since the decisions and interactions of rather large numbers of agents are simulated, which on the one hand need many calculations and on the other hand use algorithmic methods not available to algebraic models.
One such research question concerns the type and form of the interaction- and learning structures/space in which these agents 'live', since it is believed to lead to different results. In the literature models can be found where agents either live with space – on two-dimensional grids (regular or irregular) where agents interact with their neighbours – as well as without space (everybody may interact with anyone). Newer models look at the interactions between agents from another perspective – the sociological – which states that important properties of a society stem from its form – a social network (Small Worlds) which features some important distant links between agents.
This thesis will try to compare an economic/sociological simulation by implementing different parameters and testing the existing combinations on these models to compare the different outcomes and the impact on the structure and behaviour emerging in the simulation.

Contents

1 Agent Based Computational Economics (ACE) 1

1.1 Learning . . . 5

1.1.1 Non-Conscious Learning . . . 7

1.1.2 Routine-Based Learning . . . 8

1.1.3 Belief Learning . . . 12

1.1.4 Suitability for Applications . . . 15

1.2 Determination of Interaction Patterns . . . 16

1.3 Advantages and Disadvantages of ACE . . . 17

2 Characterization of the research problem 22

2.1 Simulations . . . 22

2.1.1 Analysis Using Simulations . . . 23

2.2 Structure . . . 25

2.2.1 History of Network Science . . . 26

2.2.2 Choice of Various Networks . . . 28

2.2.3 Small-Worlds . . . 32

2.2.4 Distribution of Network Ties . . . 37

2.2.5 Spatial Models . . . 43

2.2.6 Social Structure . . . 43

2.3 Game Theory and Social Dilemmas 45
2.3.1 Game Theory 46

3 Research Problem in Simulation **52**
3.1 Framework for the Simulation 53
3.1.1 SimBioSys Framework 53
3.1.2 SimBioSys Execution Cycle 58
3.2 Network Simulation 60
3.2.1 Trade Network Game 60

4 Research Problem in Testing **71**
4.1 Finite State Machine Memory = 1 75
4.1.1 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 3 . . . 75
4.1.2 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 3 . . . 76
4.1.3 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 3 . . . 77
4.1.4 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 6 . . . 78
4.1.5 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 6 . . . 78
4.1.6 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 6 . . . 79
4.1.7 Pure Buyers = 6, Pure Sellers = 6, Buyer-Sellers = 3 . . . 80
4.2 Finite State Machine Memory = 10 81
4.2.1 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 3 . . . 81
4.2.2 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 3 . . . 82
4.2.3 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 3 . . . 83
4.2.4 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 6 . . . 84
4.2.5 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 6 . . . 85
4.2.6 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 6 . . . 86
4.2.7 Pure Buyers = 6, Pure Sellers = 6, Buyer-Sellers = 3 . . . 86

4.3 Finite State Machine Memory = 16 87
4.3.1 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 3 . . . 87
4.3.2 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 3 . . . 88
4.3.3 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 3 . . . 89
4.3.4 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 6 . . . 90
4.3.5 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 6 . . . 91
4.3.6 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 6 . . . 91
4.3.7 Pure Buyers = 6, Pure Sellers = 6, Buyer-Sellers = 3 . . . 92
4.4 Conclusion 92

A Inhaltsangabe **102**

1 Agent Based Computational Economics (ACE)

According to the research done by Leigh Tesfatsion and her findings, economies are complex and dynamic. This implies that there are many agents which interact and those interactions result in the expression of different behaviours by different agents. Influenced by those behaviours, interaction patterns and global regularities emerge. So it can be followed that interaction of agents and the emergence of patterns and regularities, which are depending on those interactions, are properties of complex systems. Because agents are also able to adapt to different situations, economies are viewed as adaptive systems, i.e. agents can react to changes in the environment and also try to influence it in order to achieve their goals. To sum up all of the fundamental concepts of complex systems and agent-based modelling: [15] [20] [23]

Adaptation Individuals are able to adapt to changes in the environment through selection processes and learning.

Difference All agents are heterogeneous, i.e. they differ in their behaviour, beliefs, preferences and appearances.

Externalities Positive and negative externalities are highly able to influence

the environment. It is possible that agents create those externalities by themselves.

Path dependence This signifies that the decision of which path to take at present always influences how the future will be shaped.

Geography It is used for the research of location and city formation. By implementing geography to determine with whom it should be possible to interact in the environment, the output can be influenced.

Networks Individuals in the simulated world can be connected by certain networks which are also possibly represented through geography. There are many different types of networks in the world, e.g. friendship-networks, electricity networks, the Internet.

Emergence Emergent phenomena are aggregate outcomes that result from individual actions. Because they are often surprising it is nearly impossible to predict those aggregate phenomena beforehand. This is merely possible by observing the agents' behaviours. Peter Howitt states in the Handbook of Computational Economics [13]

> "The system as a whole is not a macrocosm of its individual parts and the parts are not microcosms of the whole."

and he also states some finding of Schelling in his paper:

> "...,as Schelling has argued..., macro behaviour can depart radically from what the individual units are trying to accomplish."

An example of emergence of aggregate behaviour is synchronization which arises only if the individuals of the environment pay attention to the actions

taken by all the others. Duncan Watts, Steve Strogatz and Tim Forrest examined the phenomenon of synchronization by analyzing the chirping of crickets.

There is no single definition for complex systems. The only possibility how to characterize them is to build a level classification. [9]

First order complex systems These complex systems just impose energy as you can observe the imposition of energy on chemical elements in physico-chemical situations. Modelling techniques are non-linear dynamical and common dynamical mathematics.

Second order complex systems In this case there already is knowledge involved. Knowledge is representing the building of connections with certain elements to get energy. By using this knowledge the energy flow can be influenced. Experience is responsible for the imposition of knowledge.

Third order complex systems In this system knowledge is not just imposed but acquired which doesn't result in a mere experience accumulation but in knowledge involving creativity.

Fourth order complex systems In this case knowledge is interactive in a way which includes peoples' aspirations and future commitments. These aspirations and commitments are understood by others in the environment.

For the modelling of those previously described systems agent based modeling (ABM) is used. A specific feature of agent based computational economics is the fact that the researchers are first building the model. After starting the simulation it mustn't be influenced during a run, i.e. the model is dynami-

cally complete. In other words, after the building process has been finished the modeller is just an observer watching the progress of his simulated world. [20]
Agents are comprised of data and methods that guide their behaviour. They can be private or public. Private data/methods can only be seen by the agents themselves whereas public data/methods can be seen by all other agents in the world. Communication is done via public methods. Agent data is consisting of type attributes (i.e. information of which role an entity plays in the simulation), structural attributes (i.e. where is the agent situated, cost and utility functions) and addresses of other agents (i.e. public attributes of other agents). Agent methods are consisting of behavioural methods that have been derived from social institutions (e.g. market protocols) and behavioural methods which are private to the agent (i.e. behavioural strategies like pricing, production and learning). [20]
Agents can take different roles, e.g. individuals like employers or employees, social groupings like firms and families, institutions like markets, biological entities like nature, and physical entities like infrastructure or the weather. According to the objective of the certain model on the one hand, they are very flexible in learning, whereas learning is used to find certain decisions, and on the other hand, they haven't got learning capabilities at all. Hierarchies of agents are also possible, i.e. agents can be assembled through other agents (e.g. a company is comprised of employees). [20]
Research of agent based computational economics is roughly divided into 4 objectives. [20]

Empirical understanding This strand of research examines the possibility of artificially generating global regularities, which are normally observed in the real world, by constructing agent based models.

Normative understanding The objective is to find out how efficient and fair economic designs can be determined by using models with interacting agents, although those agents are exerting much egoism in their strategical behaviour.

Qualitative insight and theory generation In exploring models and their dynamical behaviours by using many different initial conditions, modellers try to gain more insight into the functioning of economic systems.

Methodological advancement In this part it is determined which tools and methods of investigating economic phenomena are best. Moreover, it is tried to develop them further. This is also including tools for the testing of theories which are generated through experiments. The goal is to compare those theories against data from the real world.

1.1 Learning

Learning models have to be different so that they can be used for various situations of learning in the real world. The baseline of learning is a process used by all species which are possessing a brain. But this process advances depending on the certain species and can get more and more complicated and flexible. The categorisation of learning approaches in economics was highly influenced by psychology. [4]

In history the first psychological approach is the distinction of classic conditioning and operant conditioning. Operant conditioning is more important to economic research than classic conditioning. Classic conditioning means that new stimuli are developed based on existing ones which can then result in a

variation of preferences. Operant conditioning is based on past experiences and resembles the reinforcement learning process. It is derived from those findings that actions, which are resulting in rewards, are repeated more often in the future. Actions, which are resulting in punishment, are gradually stopped. The next approach is called social-cognitive learning theory. In this case scientists found out how certain events, which have been previously observed by agents, and the interactions conducted among those agents have influenced their learning process. The last subject is cognitive learning. Using this process, agents are able to understand coherences and the course of actions/events in the real world. In the following they are able to create and to express their own opinions. A recent application of cognitive learning is neuro-science which investigates information processing speed, stimuli interaction and what parts of the brain are used for certain processes of learning. [4]

It lasted long that economists weren't very interested in learning issues. When they started to show more interest, it was for the normative learning model where the process of learning was described as optimal. Another issue of interest were models where the behaviour reached its optimum in equilibrium. The next approach they were interested in was behaviour that did not converge to an equilibrium but highly differed from it. This divided economists into the ones that only studied learning reaching an equilibrium of behaviour and the others that only studied behaviour which was never reaching optimality. After that many economic researchers developed their own models representing learning because the traditional approaches weren't suitable for their experiments. Some of the models became more prominent and more adopted than the other ones. [4]

To get to know which models are best suited for certain applications or experiments it is helpful to classify them. There are multiple ways to do this, e.g.

classification by origin which includes models based on psychology, rationality, models that are adaptive, models implementing belief learning and artificial intelligence or biological models. As second possibility you can group them according to research fields, e.g. there are certain models used in macroeconomic research, others used by game theorists etc. The classification proposed here by Thomas Brenner is situation based, i.e. he classifies models according to the utilisation in different situations. [4]

Although there have been more fundamental learning processes defined in other fields of research, in reality there exist just two of them. The first one is reinforcement learning which resembles operant conditioning from psychology, i.e. this is learning without consideration of the subject to be learned whereas actions resulting in rewards are repeated more often in the future and actions resulting in punishment are gradually stopped. The second process involves conscious consideration on actions and the understanding of interrelationships in the world which is called cognitive learning. Cognitive learning is further split into routine-based learning, i.e. subjects learn through observed routines like imitation of behaviour, and belief learning/associative learning, i.e. consideration and belief building of observed behaviour. [4]

1.1.1 Non-Conscious Learning

Every unconscious process of learning is called non-conscious learning. In psychology classic conditioning and operant conditioning are counted among this classification. In economics reinforcement learning is applied but this cannot really be understood as a process of learning without consideration. It is somehow resembling the approaches from psychology. Because of that it can be used for modelling non-conscious learning in that field. [4]

Models used for this system are the Bush-Mosteller model, melioration learning and the Roth-Erev model. The principle of melioration learning is that the average of all past behaviour experiences is taken into consideration. The problem is that it is not really adequate for modelling non-conscious learning because reminiscence of past experiences can also include some kind of consideration. The Bush-Mosteller model and the Roth-Erev model both just consider the current situation via a frequency distribution which has been detected in the past for building a new updated version of the distribution. One difference between the two models is that in the Bush-Mosteller model the subjects are able to forget negative experiences of a certain behaviour from the past, if the type of behaviour is suddenly getting positive feedback again. On the other hand, the original Roth-Erev model remembers everything; in an extended version the feature of oblivion is included. There is also some difference in the learning speed whereas the speed is remaining constant in the Bush-Mosteller model and changes through experience in the Roth-Erev model. The last big difference lies in the reception of payoffs. The Roth-Erev model is not able to calculate negative payoffs, this characteristic is only implemented in the Bush-Mosteller model. This is a very important characteristic for modelling reinforcement learning and so it can be concluded that the Bush-Mosteller model is the only approach really suited for the modelling of reinforcement learning. [4]

1.1.2 Routine-Based Learning

Using this learning method, behaviour is derived from observations and experiences and it does not include giving an opinion about observed situations. Models used for this type of learning are models from experimentation, melioration and experience collection, imitation, satisficing, replicator dynamics and

selection-mutation equation, evolutionary algorithms and at last combined models. [4]

Experimentation Experimentation is divided into different approaches. The most common one used is the trial-and-error principle. It has to be specified how many actions will be tried, after how many attempted actions they will be called errors and at last the meaning of an error itself has to be particularized. The core idea is that the subject has to try out some actions. If they result in something positive, they are repeated in the future but if they result in something negative, they won't be repeated. This idea resembles also the idea of reinforcement learning. Thus it can be seen that reinforcement learning itself cannot only be assigned to non-conscious learning but also to routine-based learning. The next method is the principle of S(k)-equilibria which was thought up by Osbourne and Rubinstein. They state that each action of a set of actions is repeated k times. After that the subjects calculate the average payoffs of these actions and choose one according to the highest average payoff gained. The last approach of experimentation is the learning direction theory. In this case there also exists a set of actions. The subjects in the experiments try to order them in a way so that they will be able to recognize, which actions resulted in positive or negative payoffs. At last the only actions chosen are actions which increase their payoff.

Melioration and experience collection Individuals choose their behaviour according to experiences and are also able to gain insight into the connection of similar situations. For this purpose they are accumulating knowledge from various situations. The only model suited for routine-based learning

is melioration learning. As mentioned in the previous subsection the process of melioration is the calculation of the average utility from all past experiences of certain actions.

Imitation For this method there doesn't exist any general approach but there are many different applications in economics. There are two possibilities. First, just a part of the entire population, or second, simply the whole population is taken into consideration for observation. To determine which actions should be imitated you can calculate the average payoff of certain actions or you check, which of the observed crowds have got the highest payoff, to be able to choose a certain action. A further possibility is to check your own achieved utility against only one other subject's payoff at any point in time. Those were just some examples because there are even more possibilities to choose an action for imitation. Some researchers also argue that you could take methods for experience collection to model imitation.

Satisficing The satisficing principle is following the rule of selecting the first observed action that exceeds a certain threshold value of positive payoff. These threshold values are assigned to every situation by the individual itself following its expectations. Threshold values united as a whole are called aspiration level. There have been forged different approaches to initially specify this level. The first is that you can specify it to remain constant over time and second, you can implement it to vary at any point of time according to some special algorithm.

Replicator dynamics and selection-mutation equation The foundation for replicator dynamics is evolutionary game theory. This approach is about

the fitness of behaviours. According to a certain fitness level which is calculated from the average fitness of all behaviours it is decided how often certain behaviours occur. If the fitness of some behaviour is higher than the average fitness level this behaviour is used more often; if the fitness is lower it is used less. The selection-mutation equation is nearly the same as the principle of replicator dynamics but it is containing a mutation process as an additional feature.

Evolutionary algorithms Evolutionary algorithms are based on the selection-mutation equation but they are using a different method for the process of selection and a fundamental description of the subject's evolution. One difficulty of this approach is that the ability to remember past experiences is very restricted. You also have to differentiate between genetic algorithms and evolutionary strategies. Genetic algorithms are coded with binary values whereas evolutionary strategies are coded with real values. This difference has to be taken into account for modelling because mutations and crossovers have to be treated unequally.

Combined models Although it has been mentioned before that there doesn't exist any general models for the representation of different learning processes, there are 2 combined approaches. They are called the Camerer and Ho's Experienced-Weighted Attraction (EWA) model and the Variation-Imitation-Decision (VID) model. The EWA model is for both reinforcement and belief learning. Through observation a subject is able to consider the worth of undone actions whereas actions are chosen according to a logit formulation. The VID model combines all learning processes used for modelling routine-based learning, i.e. all approaches stated above in

this subsection except for replicator dynamics, selection-mutation equation and evolutionary algorithms. Individuals in this model observe certain situations and behaviours and in this way they accumulate knowledge. But these accumulated informations are only used to change the behaviour if there is some dissatisfaction with previously obtained results. If the subjects' behaviours result in positive utility they are remaining the same.

1.1.3 Belief Learning

There is some difference between belief learning from the point of view of the field of economics and cognitive learning from the point of view of psychology. Belief learning does not include all the features cognitive learning does. Thomas Brenner includes not only belief learning models in this category but also artificial intelligence models, machine learning and rational learning models. [4]

Mental models These models are composed of all informations and beliefs obtained from interaction with a certain environment. The knowledge about outcomes of certain behaviours is also included. By using this approach a subject is able to make future predictions. They will choose a behaviour according to their predictions and according to the consequences that will arise in the subject's opinion.

Fictitious play Subjects in this model remember everything which has been performed by all other subjects in the environment. So they are able to calculate the probability whether a certain action will recur and what behaviour will work best in response to other subjects' behaviours. A weakness of this approach is that the adaptation to changes in the environment is very slow and it is using a lot of resources because every happening of

the past is recorded. To alleviate the weakness of recording there exists a modified version of this model where just a finite number of past events is mapped. This modification is also speeding up the adaptation process which is a reaction to changes in the world.

Bayesian learning This approach is treated like a maximization problem. Individuals try to maximize their expected utility whereas utility is resulting from certain actions. Individuals assign probabilities of occurrence to each event. In the initial configuration those probabilities are equal. After gaining more information through the happening of events individuals are able to update the probabilities.

Least-squares learning This is also an optimization approach like Bayesian learning. In this model subjects make predictions about dependencies. These dependencies are consisting of parameters. Subjects compare the values of predicted and observed parameters and try to minimize the statistical standard error, i.e. the difference between estimated and measured parameter values, when they are squared and summed up, should be minimized.

Genetic programming The basis for genetic programming are genetic algorithms. Both approaches are using the same structure, i.e. selection, reproduction, crossover and mutation. The difference lies within the coded objects. For genetic algorithms these objects are actions or strategies and for genetic programming these objects are beliefs about the performance of the environment. Each belief is represented as a simple program. As in least-squares learning, predictions and observations are compared and the same minimization algorithm can be used to choose the programs with the

best outcome. In the crossover process two of the programs are merged. The last step is the mutation of the merged programs.

Classifier systems A feature of mankind is that people always try to classify everything and to put classified objects in classes. Classifier systems are working with condition-action rules. This means that a certain action is conducted if the corresponding condition arises. Each rule has strength as an attribute. Positive outcomes in the past result in more accumulated strength. The condition for the selection of an action is choosing the rule with the highest offered strength. Strength is updated after each round according to the outcome the rule has achieved. There is no evolution of new rules but the weakest rules are erased from time to time. To compensate for this deletion new rules are adopted. They are slightly modified versions of existing rules.

Neural networks Neural networks are a reproduction of the human brain and its structures. They are not really adequate for modelling learning processes because it is unknown how beliefs are built in a human brain and this approach is too complex. Because of that we are not able to understand the reasons for behaviour of implemented individuals who are using this type of model and it is not possible to test the robustness of outcomes produced by it.

Rule learning This method is based on reinforcement learning. Probabilities, which are calculated for each belief, are updated by using the algorithm of this learning approach. The only difference is that in the original approach actions are considered instead of rules.

Stochastic belief learning This approach resembles Bayesian, least-squares and rule learning. The difference is that it doesn't take into account all known beliefs and that each subject only takes into consideration one of the beliefs at any point in time. Which beliefs are chosen from the given set depends on the information obtained until the time of selection is reached. The reception of information is possible via own experiences, observation and communication. When beliefs are updated some of them disappear from the set and some new ones are added according to the situations observed. It is also possible that nothing is changed in the set of beliefs.

1.1.4 Suitability for Applications

Now it is time to consider which model is suitable for which type of application. It has to be determined whether it is sufficient for the agents of the simulation to learn non-consciously or whether they have to learn consciously. You also have to distinguish between routine-based learning and belief learning. What also has to be taken into account is that some of the non-conscious behaviour is due to local circumstances, i.e. this behaviour has always been like that. An example for me as an Austrian is to switch on the lights of the car during daylight. I don't reflect on it, I just do it because I learned it that way in driving school. In other countries this behaviour can be different and people may think that it is weird to switch on the light during day. So for them this would be a conscious decision. Events resulting from unconscious behaviour that lead to dissatisfaction can also inspire people to rethink. You can find recommendations according to Thomas Brenner in Figure 1.1. [4]

1.2 Determination of Interaction Patterns

There are several possibilities to model interactions. The first classification is the differentiation between endogenously and exogenously determined relationships, i.e. if interactions are defined exogenously it is decided by the modeller with whom individuals have to interact. If they are determined endogenously the interaction pattern is derived during the run of a simulation, e.g. because of experience agents have gained through the run. In models with endogenous interaction patterns the links between agents are updated very fast. This is happening according to perceived strategies. In models with exogenously determined interactions the updating is too slow to take into account strategies and so the interaction pattern in those models can be viewed as given. [22]
According to Nicolaas J. Vriend [22] differences between models can be due to

1. the establishment of links for technical reasons, e.g. communication and topology.

2. the establishment and evaluation of links for economic reasons, e.g. learning process.

3. the existence of a certain game played in the network.

4. the algorithms for decision-making if a game exists, e.g. learning process.

5. the importance of interaction patterns that emerged during a simulation.

6. the importance of game strategies that arose during a certain run.

The modelling of random or local interactions on a lattice is an approach where interaction patterns are exogenously assessed. For endogenous processes agents

have to decide themselves what connections to establish. After choosing a certain connection they also have to decide if the link is maintained for more periods or if they break it after one usage. [22]

1.3 Advantages and Disadvantages of ACE

Agent based computational economics has many advantages but also disadvantages if you compare it to standard analytical methods for economic research. But the goal of the computational approach is not to substitute standard approaches. It shall operate as an accomplishment to older approaches. There are always some subjects of research where the standard approaches have more strength than computational approaches and vice versa. [14]
Advantages: [1] [3] [14] [17] [20]

- Every agent has a certain goal. For reaching the goal they are able to engage in competition or cooperation with others.
- Agents have the capability to learn and to adapt their behaviour, beliefs, preferences and interaction patterns to a changing environment. So they are not totally controllable or predictable.
- Agents are very autonomous, i.e. they follow their own goals which can be influenced by the environment.
- Modelling of real world aspects is facilitated. You are able to eradicate many simplifying assumptions that have to be made for standard models.
- Because modelling is not based on the equilibrium approach (i.e. for the participants of the model the assumption of perfect rationality is intro-

duced so that they know everything about their surrounding world; they just manage to find resulting equilibria and examine the look of the economy in equilibrium), it is also possible to build models where equilibria are not existent (also called perpetual novelty), insoluble or where there may be multiple equilibria.

- Social communication between entities is possible.
- There are economic phenomena where no general theorems can be applied. Agent based computational economics is able to deliver some insight into them.
- Reusability of computational models is very easy (e.g. some other researchers want to do the simulation with different parameters).
- Agent based approaches can be used for many disciplines of research because some of the fundamental problems are similar.

Disadvantages: [14] [20]

- The initial specification of the whole model has to be very exact because after starting the simulation it has to run by itself.
- It has to be tested with many different specifications, i.e. it has to be tested on robustness because the results could be strongly affected by changes in the initial specification.
- It is not known whether the models will perform well with thousands of agents.
- It is not easy to validate the results obtained from the simulation against data obtained from empirical testing.

- From the results of standard analytical methods theorems can be derived whereas agent based computational models just produce examples of different outcomes.

To alleviate some of the cited disadvantages there are several possibilities. Because it is not possible to prove a theorem using the computer you can try to calculate the probability, if the hypotheses are true, in finding counterexamples. You can also use sampling methods and regression methods for testing. [14]
The equilibrium approach is based on equations. For being able to solve them the modeller has to implement many simplifying assumptions. Agents are homogeneous and behaviour is displayed by functions of mathematical form. So it cannot be determined how those assumptions influence the outcomes of the models in comparison to the real world. For this purpose computational approaches can be used. The different approaches can be compared and, like it has been already stated above, the computational approach can be viewed as an extension of the older standard approaches. [1]
If you compare experiments conducted with human individuals to agent based simulations, John Duffy argues that experimentation using humans is more restricted. So researchers who want to combine these two approaches mostly use computerized simulations for the explication of results obtained in human subject experiments. But there is still the question of when it would be better to use laboratory experiments and of when to use agent based computational approaches. If outcomes on the individual level are considered human subject experiments are useful to check external validity of computational approaches. If outcomes on the aggregate level are examined it is possible to check the results of laboratory experiments against agent based simulations. In some situations it can be followed that these two approaches are almost perfect substitutes, i.e.

sometimes they can be used in the same way. [8]

Expectations for outcomes in the future of individuals influence their behaviour at present. Further on their cumulative individual behaviours result in some aggregate outcome in the future. They try to forecast this result and this influences future results, i.e. this process is self-referential. For the purpose of handling expectations of many agents, static economic theory has developed the approach of rational expectations. It states how to choose an expectational model for getting a definitive result but this approach is not very useful to represent reality. An approach that didn't have that many unreal assumptions is bounded rationality. It was developed by Herbert Simon more than 50 years ago. But the problem with that one is that there are too many possibilities how to relax the assumptions of rational expectations theory. It is not feasible to determine which assumptions would be right and which would be wrong. For real world representation a generative approach is more useful. [23] [1]

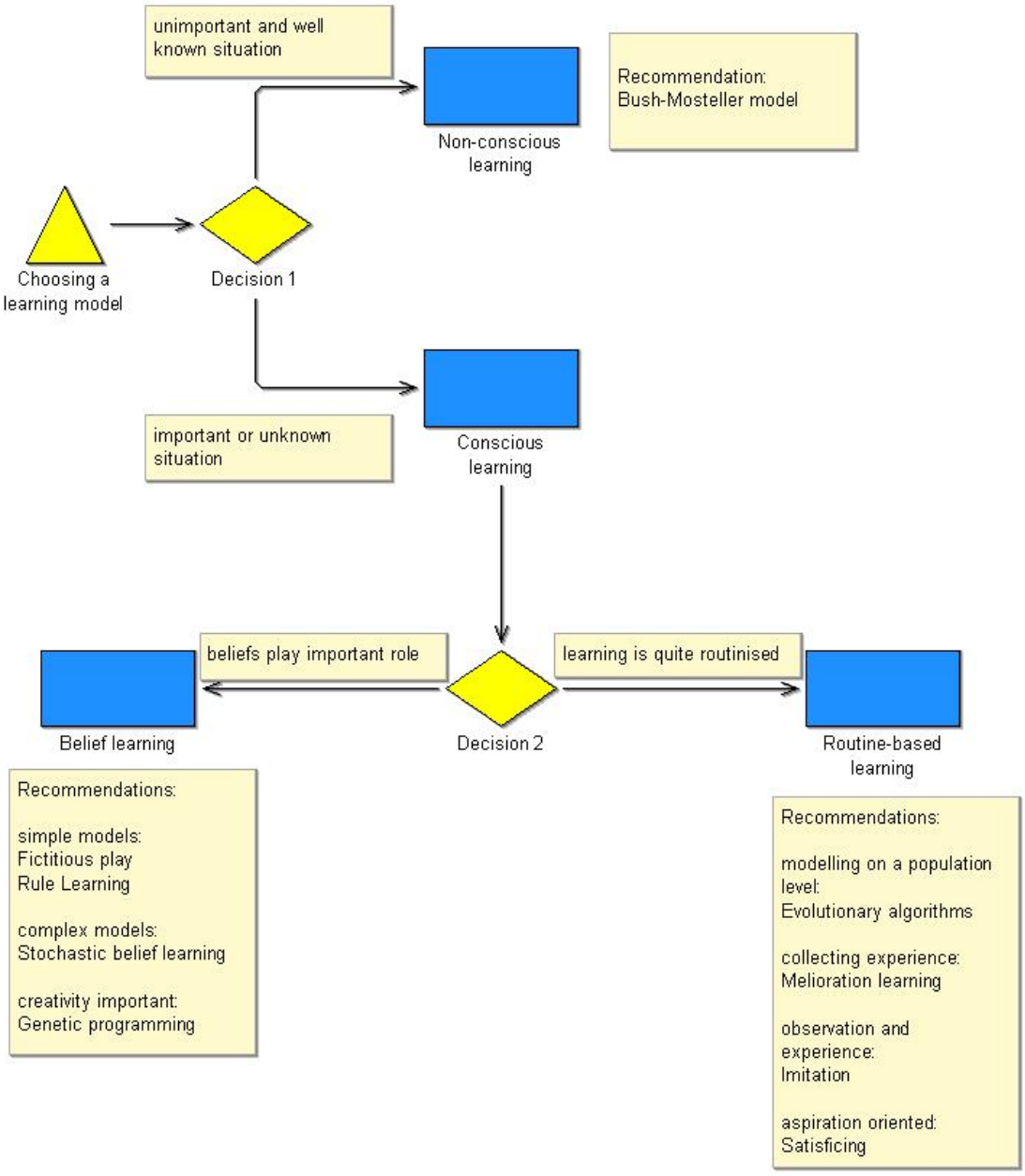

Figure 1.1: Recommendations for choosing a learning model [4]

2 Characterization of the research problem

In this chapter the different parts of the problem are introduced. It starts with the use of simulations and which analysis can be conducted using them. Next a network overview is given. This includes the different representations of structure and also historic approaches are considered, until reaching the theory about small worlds and social structure. Further on some insight into game theory is delivered.

2.1 Simulations

Simulations in this context are meant to be computational programs using certain initial specifications. They are running self-reliantly whereas the modeller is able to observe definitive results. Those results can be used for [2]

Prediction By taking present conditions as initial specifications simulations are able to generate predictions for future outcomes.

Performance Simulations are also used for various assignments, e.g. diagnosis for medical purposes, recognition of speech.

Training In reproducing an existing world it is possible to provide training for individuals.

Entertainment Games representing reality are developed (e.g. flight simulations) for the entertainment of people.

Education Some developed computer games also serve as educational devices (e.g. Sim City) because people are able to observe effects of their behaviour and of introduced policies.

Proof Simulations are able to act as a proof of existence for given situations.

Discovery They are also implemented to detect the coherence between relationships and principles.

2.1.1 Analysis Using Simulations

To perform analysis with simulations you can divide the process into three main steps. First you have to program the model, then you have to analyze the resulting data and at last you have to publish the results so that others are able to work with them or to review them. Another important thing is to replicate given simulations. [2]

For the programming it is important to follow certain principles. Programming has to be valid in generating the results, i.e. results mustn't be influenced by bugs in the code. Next it has to be usable, which means that everyone should be able to understand it. After that everybody should be able to give an interpretation of its results. Extendability shall ascertain that the code can be used by others for the creation of an extended version. It should also be possible to use the program for other purposes. It is very important that a programming language

is chosen which has already some simulation features and which can easily be used by the majority of people, who are involved in studies like that. [2]
As already mentioned in the first chapter the data, which results from a run of the simulation, incorporates path dependency, i.e. events occurring in the past can absolutely influence the resulting data. Because of that many runs using the same initial conditions have to be executed to be able to state, whether the outcome is a standard one or not. The model can be tested with many different initial specifications. This renders it possible to study many different aspects. The effects of the parameter changing can also be researched using statistical approaches. For quantitative changes regression analysis is used and for qualitative changes analysis of variance is taken. [2]
For sharing the simulation with others the first step is to publish it in a magazine to attract the interest of other researchers. Due to the fact that it is not possible to write every detail into an article it is necessary to write a completed documentation including the source code, a model description and instructions, how to run and interpret its results. [2]
The replication of simulation is a subject which is often neglected in the analysis of simulations. But it is important to see whether the results, which have been obtained by a previous researcher, can be reproduced by using other simulation environments. As Axelrod states here [2] he and his colleagues can derive some important lessons from their project of replication.

1. Replication can be executed very fast and easily.

2. You have to consider the level of replication

    a) Numerical equivalence, i.e. all results are the same which is only possible using the same initial specification and generator of random

numbers.

b) Distributional equivalence, i.e. there is no difference in statistical analysis with the same mean values and standard deviations.

c) Relational equivalence, i.e. the results of the simulations have the same shape or characteristic.

3. For the testing of distributional equivalence you have to take a sample size that is large enough. Otherwise it could be possible that the null hypotheses (i.e. that the values are equivalent in this sense) is never rejected.

4. If there is just a little difference between the configuration of the original and the replicated simulation this can influence the result so that no distributional equivalence is attained.

2.2 Structure

To be able to perform a better study of the influence of interactions and to determine who the individuals are, that evoke this influence between agents in an environment, networks are used. This is because their presence in a simulation can have severe impacts on the behaviour of individuals. Furthermore, it can be stated that in this case mathematical theory has too many restrictions, because networks can become very complex; the same is true for empirical studies. So agent based computational models are used. [26]
In earlier times networks were just viewed as fixed but in reality they are more complicated. Networks are structures wherein the nodes are generating something like power, data or behaviour. The structure itself is evolving and changing over time. The network setup influences everything which is happening. Every

different discipline of science has its own form of doing network research. Now a point in time is reached where all the best features of all disciplines have to be combined to understand all the implications of network science. This is not an easy task because each of the disciplines has a different manner of expressing similar topics. So researchers have to learn in advance how to communicate and how to understand each other. [23]

2.2.1 History of Network Science

To study the interrelationship of communication networks, Paul Erdös and his colleague Alfred Rényi developed the theory of random graphs. Links in a random graph are determined randomly. If there are enough concatenations all the nodes in the graph are connected with each other, i.e. you are able to reach any node from any starting point. This type is called a connected graph. But how many bonds have to exist until the graph gets connected? The answer to this question was given by those two researchers. They stated that, if the average number of connections per node was less than one, the graph wasn't connected at all. The outcome here was that you could find many isolated groupings of nodes. But once the number of an average of 1 connection per node was reached, a critical point had been hit. Suddenly the fraction of connected nodes rose very fast. They called that phenomenon phase transition which can be seen in Figure 2.1. But there was a problem with this approach as soon as it was compared to real world networks, because then they were able to see that real networks weren't built at random. [23]

The mathematician Anatol Rapoport asked himself the same question as Erdös and Rényi did. After having found similar results he tried to deal with problems concerning random graph theory. Two nodes which are connected via a

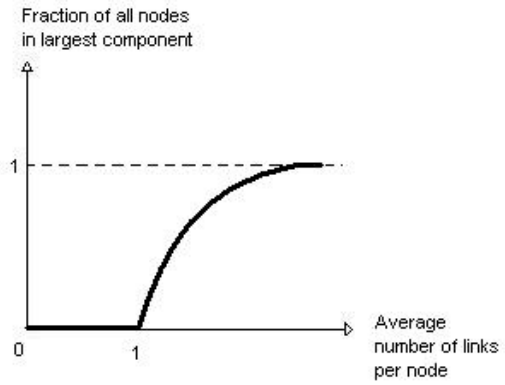

Figure 2.1: Random graph connectivity [23]

link build a relationship (i.e. a dyad). If individuals belonging to 2 different dyads shared a node, Rapoport stated, it was very likely that the 2 other nodes, which were strangers previously, also got to know each other. So they built a configuration that Rapaport called 'triadic closure'. In this theory dynamic evolution of networks was taken into account. This had been the new insight of the triad structure of networks. Basic theory of triads was already developed by a German sociologist called Simmel 50 years earlier. The approach of Rapoport was the first departure from pure random theory but still some properties of it remained. So he called the resulting structure 'random-biased nets'. But the problem was that he couldn't determine a proof or a usage for his modelling approach because of the mathematical and experimental limitations of his time. So it vanished for some time. The only thing he could verify was that it was also possible to develop longer cycles, if triads evolved in a network. [23]

There were researchers from other disciplines that examined nearly the same questions like physicists, but they chose another direction to start from. Sociologists tried to understand the network structure in order to determine social

role patterns of individual players and the group. Physicists however tried to understand what macro behaviour/properties can emerge from micro behaviour/ properties, if all individuals in the network had perfect knowledge of their environment. They found out that at a critical point no center or central authority in the network was needed for universal organization of behaviour. Synchronization could be reached via some random events that were rather small. Those events would have never influenced the overall outcome under conditions of normality. This happened, because actions were viewed as operations that could be seen throughout the whole system, although they took place just locally. This finding could be interpreted as another version of phase transition which was applied for magnetism, freezing of liquids etc. The conclusion reached was that different disciplines with highly different topics of research could have very similar characteristics when complex systems were considered. [23]

2.2.2 Choice of Various Networks

In this section following the arguments of Allen Wilhite we just take fixed or stable networks into account. Normally networks that exist in the real world are evolving over time; so it may not seem to be an adequate assumption for modelling to view them as stable and unchanging. But if it is considered that evolution is working very slowly, this fact may seem more appropriate. The graphs considered in this section only feature undirected links. No node can have a link with itself (i.e. it is simple). Any node is reachable from any starting point (i.e. the graph is connected) and all edges occupy the same value (i.e. they are unweighted). [26]

The complete network (Figure 2.2) In this graph there exists a connection

between each of the nodes. The calculation of the number of edges is very simple because it is following the sequence of triangular numbers. Triangular numbers are binomial coefficients. For the calculation of the number of edges you have to sum up n nodes from *1* to n. [24]

$$X_n = \sum_{k=1}^{n} k = \frac{n * (n+1)}{2} = \begin{pmatrix} n+1 \\ 2 \end{pmatrix} \tag{2.1}$$

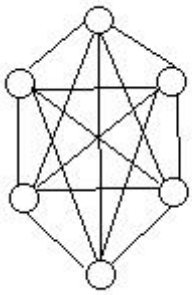

Figure 2.2: The complete network [26]

The star (Figure 2.3) One node is connected to all the other ones but the other nodes themselves aren't linked.

The ring (Figure 2.4) In this network each node has a connection with just a few neighbours, not with all other nodes in the network. If you take for example a ring with 8 nodes, each node is linked with 4 other nodes.

The grid (Figure 2.5) The grid is looking like a chessboard where nodes are situated at each crossing.

The tree (Figure 2.6) The tree network has a hierarchical character. One node is sitting on the top and this node has a few branches which also have branches and so forth.

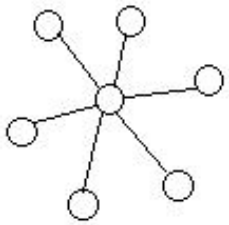

Figure 2.3: The star network [26]

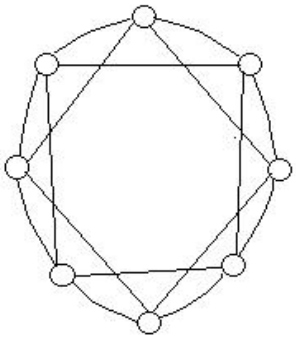

Figure 2.4: The ring network [26]

Small-world network (Figure 2.7) Small world networks are first defined through path length. This is stated by Wilhite [26] as

> "the average number of edges that must be traversed to get from one node to any other node"

. Second they are defined through the amount of clustering. This is defined by Wilhite [26] as

> "the degree of the graph or the average number of edges connected to each node"

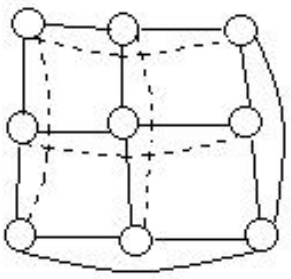

Figure 2.5: The grid network [26]

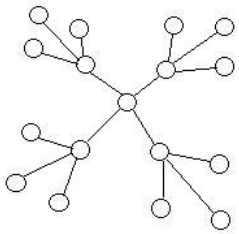

Figure 2.6: The tree network [26]

. For this type of networks the clustering level has to be high and the path length has to be very small.

Power network (Figure 2.8) These networks are also dubbed scale-free networks. They have some single nodes that act as hubs (i.e. hubs have many nodes which are connected to them) and a large number of nodes which only have a small number of links. The distribution of these nodes is a power law distribution.

[26]

The topic we want to investigate here is how different network topologies influence behaviour of individuals in the simulation. Examples are cooperation and

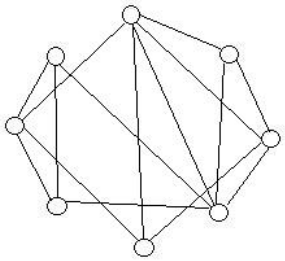

Figure 2.7: The small-world network [26]

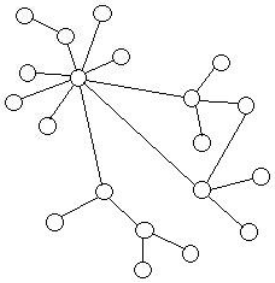

Figure 2.8: The power network [26]

coordination which are behavioural patterns that are often displayed by prisoner's dilemma games from game theory. The utility an individual gains from a certain behaviour (e.g. coordination and cooperation respectively and on the other side defection) is measured. [26]

2.2.3 Small-Worlds

For the development of a model that represents the small world phenomenon Steve Strogatz and Duncan Watts think that it has to incorporate 4 basic features. First there shall be several groups where the nodes have many connections between themselves (e.g. circles of strong friendship). There shall also exist a few links between different groups (e.g. some people have friends of different groups

whereas it is not very likely that they are connected in a stronger way). Second the network representation has to be dynamic, i.e. it has to evolve over time. Third there are different probabilities for the establishment of new connections. These are influenced by the connections at present (e.g. you are introduced to a friend of your friends you didn't know before). At last new links can be established which are only depending on innate preferences. They are not built via already existing links (e.g. decision of changing your workplace or the town where you are living). Decisions we take or behaviour we perform because of our own preferences and abilities are called 'agency' in sociology. Decisions/Behaviour that are/is just affected by the social structure, which can be found around us, is called 'structure'. Behaviour dependent on agency isn't really random but it appears to be. So it is possible to model it using a random network theory approach. But the evolution of relationships which is dependent on structure is constrained by already existent connections. Those 2 features are conflicting and a trade-off between them has to be imposed. So to lay on more weight on either one of these two is a hard thing to determine. To solve this problem the 2 researchers started with the features' extremes and tried to fill the void in between. They developed the 'alpha model' which can be expressed mathematically through an equation. In this equation a single introduced parameter *alpha* is changed from zero to infinity. The parameter states the probability of shortcuts in the network which is not known from the beginning; this was a finding after the development of a second modelling version. If *alpha* is 0 then all newly built links in the network are established via the social structure. When it reaches infinity the connections are totally established at random (see Figure 2.9). [23]

The thought of this approach resembles the model of Rapoport but in this case

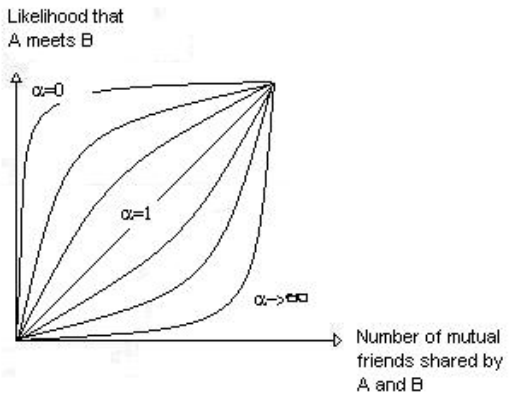

Figure 2.9: Interaction rules for different values of *alpha* [23]

the researchers weren't restricted in testing. They could make use of computers to test their complex system. They tested the model with a high clustering coefficient and short path length. So they saw that at the extreme of *alpha* being 0 there was a high clustering coefficient. At the point where *alpha* was reaching infinity there were very short path lengths. But it didn't look like those two properties could exist both at the same time. Computer testing taught them otherwise. Results show that path length is low at both extremes of *alpha*. At first it rises but when it reaches a critical value the disconnected circles of people get connected. After that path length shrinks again. This can also be viewed as a phase transition like in the approach of Erdös and Rényi. The clustering coefficient is very high for low values of *alpha*. It reaches its maximum after some time of increasing *alpha*. Then it is plummeting very fast. Intentionally you would say that the phase transitions of these two features occur at the same value of *alpha*. But computer experiments show that the path length is already shrinking even before the clustering coefficient reaches its maximum. The resulting curves can be seen in Figure 2.10. The blue part represents the

region of networks having small-world properties. [23]

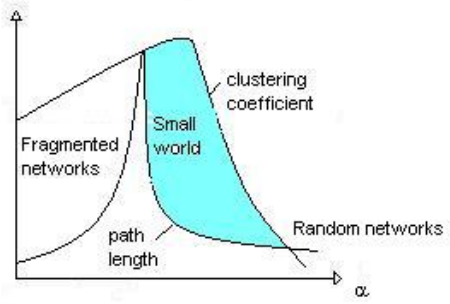

Figure 2.10: Path length and clustering coefficient for different values of *alpha* [23]

In this representation it is only possible to have, on the one hand, many fragmented circles of connected people. On the other hand, there is just one big network where everybody can be reached. In reality social networks seem to be divided into more than one large component. The next topic of research was to get to know the causes for the small-world phenomenon. For this purpose another model was developed and dubbed 'beta model' as a second approach to better understand small-world networks. They used periodic lattices, i.e. lattices where the edges are connected so that there is a passage from one edge to the other, to represent ordered interactions. Another element were random networks as representation for disordered networks. Once again those two versions were the extremes and the researchers tried to determine the stages in between. As in the alpha model a parameter was introduced, now called *beta*. Starting point (i.e. *beta* is zero) is a one-dimensional lattice which looks like a circle. The nodes of this circle only have connections to their two neighbours whereas the neighbours

also have a link between themselves. If *beta* is between zero and one it represents the probability that a link in the network is taken and rewired to another node. For this purpose every connection of the network is considered. If the parameter is one then the resulting network is a random graph. The transition of *beta* from zero to one can be seen in Figure 2.11. The rewiring of connections in the network creates a shorter path length but the clustering coefficient is still high. Those features are once again the characteristics of a small-world network. But as it can be seen in Figure 2.11 only few rewiring-steps are needed to generate a large impact on the path length whereas the clustering coefficient in comparison has a rather slow descent. [23]

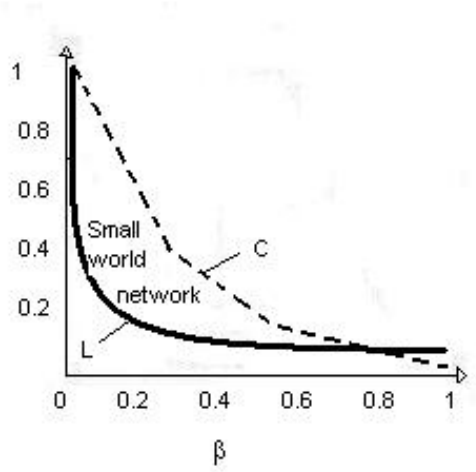

Figure 2.11: Path length and clustering coefficient for different values of *beta* [23]

A characteristic those two models have in common is the fact that individuals aren't aware of the whole world. They are just able to recognize their connections to close friends in a circle. This is an important attribute if you consider disease spreading, spreading of computer viruses or the search of information in

organizations. Another important finding of Watts and Strogatz is that small-world networks can be found in all types of network systems and not just in social ones. This allowed them to find data for testing more easily. The ancient problem was that it was impossible for the representation of social relations to find enough and not falsified data. So when there was the possibility of taking any networked structure, which had been well documented, the problem was resolved. One of the examples is the so-called 'Kevin Bacon Game'. It is taking a movie database from the internet as a footing, where it was recorded which actors played in which films. The game starts with the actor Kevin Bacon and finds out what path length is between any other actor and him during a runtime . This example was also extended to determine the average path length between a number of 225,000 actors. The result of average path length obtained was about 4 and the clustering coefficient was at 79 %. The experiment was also conducted using networks which had not much to do with social networks (e.g. electronic transmission network, neural network of an organism). All of them also had the characteristics of small world networks. However, if you consider any other innate properties they are totally different. [23]

2.2.4 Distribution of Network Ties

Watts and Strogatz forgot about one important topic in their research, namely which probability distribution of the number of connections any node has is forming the basis of the network. They just assumed that the distribution was shaped like a normal distribution. The normal distribution has one maximum signifying the average and there are steep descents on both sides of the it. The descents show that there aren't many different numbers of neighbouring nodes. The most nodes have the average number of neighbours (see Figure 2.12). [23]

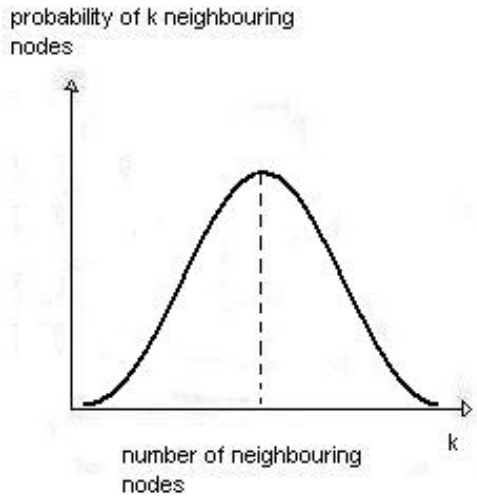

Figure 2.12: Normal distribution of probability of k neighbours [23]

But the two scientists Albert-László Barabási and Réka Albert considered this flaw in the research of Watts and Strogatz. The distribution of random graphs always looks like a Poisson distribution which resembles a normal distribution. So that difference wasn't of much concern but Barabási and Albert found out that many networks in the real world have a completely different looking distribution, the power law distribution (see Figure 2.13). The power law distribution has a different shape than the normal distribution. The maximum does not form an average value, the curve starts at the maximum and decreases very fast until a certain point is reached. After that point the fall of the curve is not as rapid as before but is going on until it has reached infinity. [23]

The interpretation of figure 2.13 is that networks which have power law distribution have a few nodes with many connections to others (also called hubs), and many nodes that just have a few connections to others. As an example you could take the existing network of airports. On the one hand, there are some big

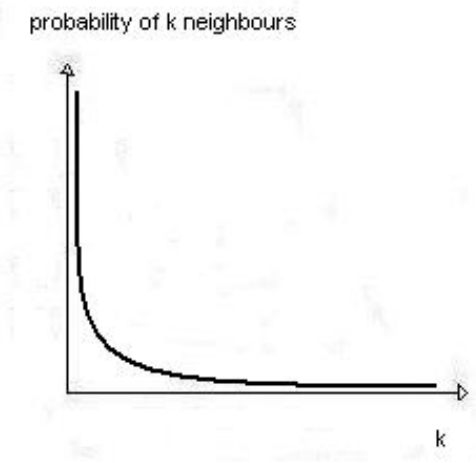

Figure 2.13: Power law distribution of probability of k neighbours [23]

airports in big cities that are directly connected all over the world. On the other hand, there are many small airports with just a few links, mostly connected to hubs. For networks that have this distribution it is not appropriate to calculate an average value and to state that this is the only value which counts for most points in the network. In reality the hubs have strong influence on this value. If you take the example of the distribution of wealth among the population in a certain country few people are very rich and most people are located in the 'middle class'. After calculating the average wealth the average is far too high for most people because the result is highly influenced by the rich ones. [23]
In thinking about the network of airports as having a power-law distribution Barabási and Albert also reached the conclusion that these networks didn't have limitations as had networks with normal or Poisson distributions. They didn't have a limit on how many links a certain node could have which gave those networks the name 'scale-free networks'. Many networks exhibit the characteristics of being scale-free, like the World Wide Web considering the structure of its

links. Also the network of movie actors mentioned in the previous sub-clause can be added to them. The next finding concerning this type of networks was the fact that it was possible that they self-developed. In the beginning of the evolution of a network structure there exists the same probability for each node to get connected to another one of the nodes. When new ones are added over time, it can be seen that the younger nodes have some disadvantage. The older ones have higher probability to get new links to the others. After enough time the distribution of the network is changing from a degree distribution to a power-law distribution. Taking the previous example of wealth in a society it can be seen that it is far more easy for rich people to get richer than for average people to become rich. To sum up every network can become scale-free if it has the features of being a growing network and if it exhibits a higher probability for the formation of new connections by nodes which have already many bonds to other ones. [23]

One problem of the last finding is that it is only valid for infinite networks. Because every network in the real world is finite also scale-free networks have a certain point where the characteristic of being scale-free ends. But it is not sure whether this is because of a finite network or because of the fact that this is a property inherited by the network. This is like the potential of individuals only to be able to maintain a certain number of friendships. It can be followed that in the real world it is impossible to always uphold scale-free properties of networks because links are holding costs. Due to this fact it is not possible to extend networks to infinity. [23]

The next thing Watts and Strogatz had to accomplish in their research was the inclusion of social structure. Scale-free models do not incorporate this from the beginning. Social structure signifies how close or far away the single nodes are

from each other. The two men followed an approach of Harrison White, who was in the first place a physicist and then also graduated in sociology. He stated that individuals formed groups with others who did the same things. He called these activities contexts whereas contexts were constituting the structure of networks. Watts and Strogatz defined the distance in networks between any nodes using the contexts of Harrison White. They decided that individuals who shared many contexts were very close and individuals who only shared a few or no contexts had a larger distance between themselves. So they followed that you had to consider two different structures. On the one hand, the structure of the network itself and, on the other hand, the social structure. [23]

Because they thought of this topic to be too complicated for themselves alone they gathered a new member for their research group, Mark Newman. They created a network called affiliation network which served to show the social structure of networks. Instead of being made up of a single type of nodes (in the following called unipartite or single-mode networks) it was comprised of two different versions of nodes and was called a bipartite or two-mode network. Only different types of nodes could be linked. An example was a network consisting of actors and the movies they appeared in. The first type of nodes were actors and the second type were movies. If an actor appeared in a certain movie a link was drawn from the actor to the movie. It was possible for actors to be in multiple movies and, vice versa, it was possible for movies to have multiple actors. Unipartite networks only had a single distribution underlying their nodes. Bipartite networks incorporated 2 distributions, e.g. the number of actors belonging to one movie and the number of movies each of the actors appeared in. As can be seen in Figure 2.14 a bipartite affiliation network can be split into two unipartite networks, e.g. two of the actors are linked if they are in the same movie and

two of the movies are linked if a certain actor is acting in both of them. The network which represents the actors' relationships is called affiliation network. The one which represents the connections in movies is called interlock network. The scientists discovered that if a bipartite network was built completely at random from the two unipartite networks it still showed all the features of a small-world network. By using this approach it was possible to demonstrate the network's dynamics and the development of structure regarding social and networking facets. [23]

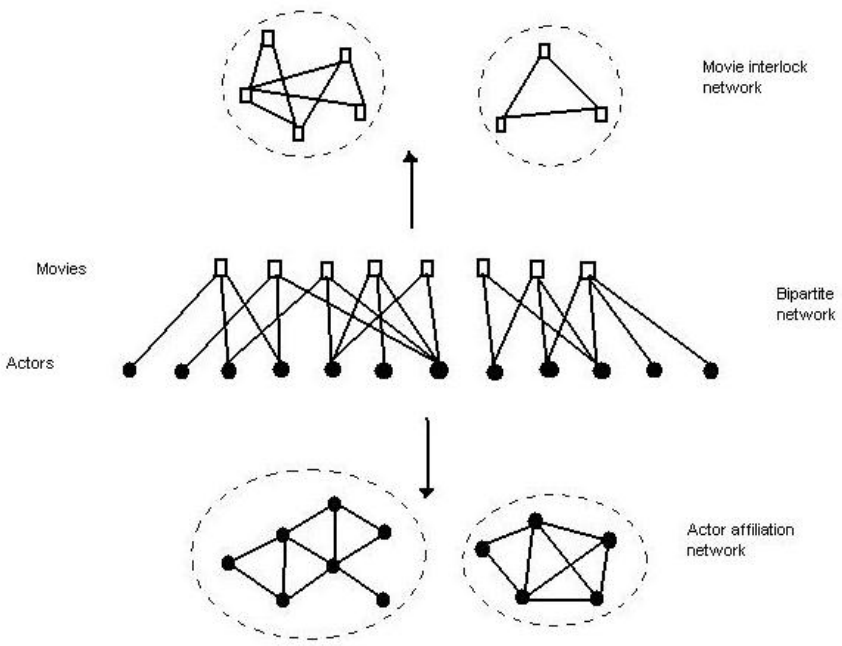

Figure 2.14: Affiliation network [23]

2.2.5 Spatial Models

.

In a spatial environment it is possible for the individuals to move. It can also be seen as a structure for interactions. All spatial models should be able to run in aspatial conditions to be able to compare what behaviours and results can be derived from initial specifications of the model and what can be derived from their interaction structure. In spatial models there are also additional costs, e.g. costs for shipping between the different nodes of the network. [7]

2.2.6 Social Structure

Social structure is influencing outcomes of economic processes, but why? One reason for that is that people rather tend to believe other people they know personally. Punishment and rewards also are more effective when carried out by known individuals. The last reason is the trust that someone who has strong incentives to do things amiss for the community only for getting some reward for himself is doing the other thing which is better for all of them. [12]
There are four core principles that have been developed by sociologists. [12]

Norms and network density Norms are defined as thought regulations for good behaviour which are common in the network. The strength of these norms is highly depending on the density of the underlying network. If the network has higher density the norms are stronger and if it has lower density they are weaker. The reason for this is that, with more density, norms are discussed more frequently. So behaviour that does not fit in is more easily encountered for the imposition of some punishment. The more punishment is noticed by so called free-riders (i.e. individuals that want

to gain just the reward for themselves and because of that work against the community) the less is the free-riders' occurrence. But it cannot be followed that the networks are more dense if they are bigger. If they get too large people will not be able to maintain many links to others for the keeping of high density.

The strength of weak ties Ties in the social network can be strong or weak. This represents the closeness of people who know each other. Strong ties are kept with close friends and weak ties are upheld with acquaintances that aren't very close. The funny thing here is that it is easier to get new informations through weak ties. This phenomenon arises due to the probability that your close friends are also close to your other close friends. This signifies that they build a circle where it is hard to get recent news because everyone residing in this circle has the same status of information. So you have to get new information through a weak tie that is connected to another circle which has different news to offer. An example for this is that it is often easier to get a new job through information received by an acquaintance than through information received by a close friend because it is very likely that you already know everything your friend can tell you. Summarized information diffusion in a social network is highly depending on weak ties.

The importance of 'structural holes' This is some extension of the argument of weak ties. It states that ties which are connected to many different friendship circles in the network gain an important advantage for their possessors. Their possessors are able to exploit the 'structural holes' offered by those weak ties.

The interpenetration of economic and non-economic action This can also be stated as the dependence of economic actions on non-economic actions or institutions. This phenomenon only arises when those actions are mixed. This is called 'social embeddedness' by Granovetter. One negative example is corruption. A positive one is finding a job by means of friends or acquaintances.

After the examination of various models and real world occurrences it is possible to make a classification of three different structures. The first one is the structure where interaction circles have hardly any connections between them, i.e. they are decoupled. In this situation, if interests are very different, the structure has not much influence on the collective outcome. A structure with a few weak ties has more cooperation but tends to be influenced by a single powerful entity; with many weak ties there is also a lot of cooperation but in this configuration it is hard for one entity to exercise that much power to achieve coordination from a central position. [11]

2.3 Game Theory and Social Dilemmas

Why do individuals cooperate although they always have very strong incentives not to do it? Some factors that are playing along this question are, considering resources, not to use all of them until destruction is faced. This is mostly regularized by rules which are established by authorities and by trust between the particular agents. But authorities also have the ability to defect. [16]
Cooperation is a characteristic of trust between individuals in a way that they are able to overcome social dilemmas if there is a certain amount of trust established between them. Because of that it is possible that people adopt a strategy where

the result is not optimal for themselves. But it is good for the community in order to achieve better outcomes than others who act rather egoistic. This phenomenon is dubbed the 'paradox of rationality'. It is also important to take distrust into account because it is just the other side of trust. It can also have deep impact on the behaviour of people. Cooperation is also highly depending on relationships between individuals, on past events and on the structure of the network. As it was already mentioned in the previous section the density of networks plays an important role. Collective action in the network would also be difficult if the single groupings weren't connected trough weak ties. [11]
At the beginning there is no information available who you can trust in an environment. This is learned throughout the progression of time. To have some insurance from the beginning it is possible to build safeguards against defection. These two possibilities which are standing vis-à-vis incorporate a certain trade-off. Researchers argue about which one of the both approaches is more costly, to gather information or to protect from the beginning. [21]

2.3.1 Game Theory

Using game theory, which is a mathematical application, researchers in various fields of sciences try to find out which behaviour and respectively interactions people follow in certain competitive situations. The most important point here is the process of reaching decisions. The first approach was built by John von Neumann who also published a book on this topic together with Oskar Morgenstern in 1944. This early version of game theory was about games between 2 individuals which played zero-sum games. Traditionally, the games should determine equilibria. The elements of a game are the agents playing, the strategies which can be executed and the payoffs which are received when a certain strategy is

played. The payoffs can also be seen as the utility from which the players are able to benefit. Their goal in the games is to maximize payoffs. [10] [19] [25]

2.3.1.1 Game Types and Game Representations

Zero- and non-zero-sum games In zero-sum games the sum of the payoff individuals are able to receive is always zero. So if one of the players wins, then the other one has to lose and vice versa. In non-zero-sum games this is not the case; the result of the overall payoff can be any number.

Cooperative and non-cooperative games Cooperative games have the feature that agents can make agreements. In this case it is compulsory to stick to them. In non-cooperative games cheating is possible although arrangements are made.

Symmetric and asymmetric games In symmetric games the payoffs do not depend on the individuals themselves but only on the strategies which are selected to be played. One example is the prisoner's dilemma. Although normally there are different payoffs and strategies for the players in asymmetric games it is still possible for the players to have equal strategies.

Simultaneous and sequential games In the case of simultaneous games the individuals do not know, when choosing their strategies, what strategies the game partners have chosen. In sequential games every player knows what actions were taken previously.

Games with perfect and imperfect information In games with perfect information the individuals can remember exactly which actions were taken in the previous steps by all the others. This is one of the most simple ap-

proaches in game theory. It is not possible to have perfect information in simultaneous games. Games with imperfect information can have both sequential and simultaneous activities. A different case is complete information where just the strategies and payoffs of the others are known but none of the things that happened during the game.

Infinite games These are the games which are going on endlessly, i.e. the turns of actions are never coming to an end.

Discrete and continuous games In the case of a certain number of individuals playing, a certain number of strategies and actions taken you are talking about a discrete game. Continuous games for example are the ones with the possibility to select a certain strategy out of a continuous number of strategies.

Extensive Form This representation is looking like a graphical tree (see figure 2.15). The nodes are points where individuals can choose their strategies. The root is usually the point where player number 1 chooses his strategy, the nodes one level down are for the second player etc. At the footing of the tree you can find the payoffs. This form is usually designed for showing games with sequential course.

Normal Form In this case the representation is looking like a matrix (see figure 2.16). The payoffs are denoted in the fields. The ones for the rows are cited as the first digit and the ones for the columns are cited as the second. The chosen strategies of the players are represented by rows and columns. Each of the players is appearing either in the rows or the columns. This form is usually designed for games with simultaneous actions.

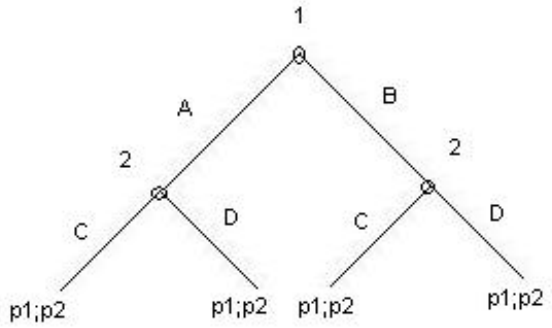

Figure 2.15: Extensive Form [25]

Function Form In the case of functions the payoff for each action which is chosen is calculated from a given function. So the payoffs possible to be earned aren't known beforehand.

[19] [25]

2.3.1.2 Prisoner's Dilemma

The prisoner's dilemma usually is a game with 2 participants and 2 possible strategies. It is named like that because the paradigm, which forms the basis of this approach, was about two individuals that had been arrested. There was not enough proof to convict them of the serious crime they really had conducted but there was enough for proving a minor one. Because the authorities wanted to get them for the more serious misdeed they were providing them with a few possibilities to choose from. If one of them was admitting that they did it and the other one was not, the first one would be free and the second would be imprisoned for a high number of years (possibility 1). If both were giving in then both would go to prison for less time than in the previous possibility

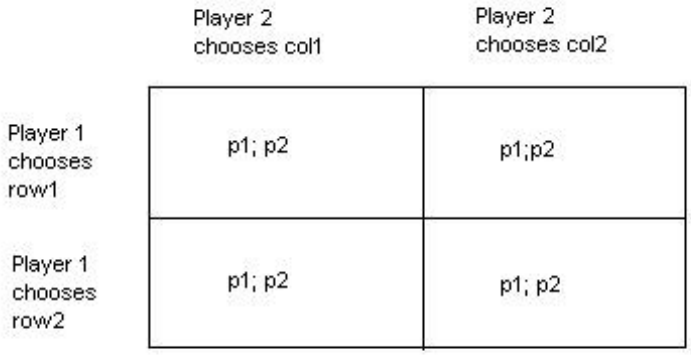

Figure 2.16: Normal Form [25]

(possibility 2). If none of them was confessing they would be in prison for the minor charges of the other crime (possibility 3) which was less time than in the case with both criminals giving in. This course of events can be pictured like in figure 2.17. A payoff of zero represents the worst situation, a payoff of one stands for possibility 2, a payoff of 2 is a representation for possibility 3 and the highest payoff is gained for being free after choosing the first possibility. Of course this is modelled as a simultaneous game so that the players don't know the decision of their partners beforehand. [19]

In the following research included in this thesis this form of game-playing is used for the determination of trading partners. In this case the agents gain or lose via their conducted trades. They have to choose between the strategies of cooperation and defection.

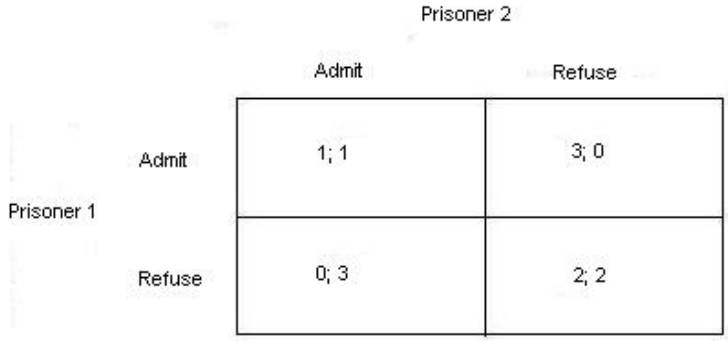

Figure 2.17: Normal Form of the Prisoner's Dilemma [19]

3 Research Problem in Simulation

In the following chapter you can find a detailed description of the simulation which was taken as a medium for being able to get results for this research. Anybody who is interested in more detail and also wants to study the code of the simulation can find details on the homepage of Leigh Tesfatsion (http://www.econ.iastate.edu/tesfatsi/tnghome.htm).

There are several reasons for choosing this simulation. This is an approach which incorporates some of the characteristics which have been described in chapters 1 and 2. In this case a genetic algorithm is used as the method for learning. Furthermore, the selection of trading partners is determined endogenously. This means that the agents themselves identify their partners via the payoff which can be expected in an interaction. After a certain trading partner is accepted the interaction itself takes place as a prisoner's dilemma. The program offers a very detailed documentation which makes it easier to understand the code. The simulation builds networks through the interaction of agents. The links which are established are written into the output. From the output you are able to study the building process of these networks. Another display of the output is the behaviour of the agents; they can show cooperation or defection. The last

important output is the readout of all payoffs which are received by the individual traders in each of their trading interactions. The code of the simulation can be downloaded from the internet and is already fully executional. You can choose between different versions of development. For this research I chose a simple approach without any graphical user interface. The output is written in lists where you are able to find the important details mentioned in the previous lines. It is also certain that the program has already gone through many tests. So you can be rather sure that the major flaws have already been determined. Therefore the output becomes more trustable.

3.1 Framework for the Simulation

As a framework for the simulation program, SimBioSys is used. This framework has been developed by David McFadzean for biological simulations and was his thesis to reach a master's degree. The program is comprised of several different classes. The classes used in the trade network game are derived from the ones developed in this approach.

3.1.1 SimBioSys Framework

The original purpose for designing and developing this class framework was to use it for the modelling of evolutionary biological systems, e.g. sexual selection/reproduction. The modelling of biological systems includes the choice of the most suitable learning algorithm to consider different features. Using the SimBioSys framework it is possible to implement genetic algorithms, genetic programming, evolutionary programming, cellular automata, neural networks, neuronal networks and artificial life/intelligence. So it can be followed that you

are able to build many different simulations which represent activities that appear in the real world. [18]

As can be seen in Figure 3.1 the framework consists mostly of abstract base classes. It is not possible to create instances from these base classes. They provide objects and functions for their subclasses which can be instantiated and extended. [18]

On the top of all classes is the class *bioObject*. Among classes of the SimBioSys framework, the classes *bioSimulation*, *bioWorld*, *bioPopulation*, *bioThing*, *bioProgram* and *bioGType* are derived from *bioObject*. They inherit all data and methods from this class. The only class derived from *bioObject* which is not designed for the basic framework is *bioList*. It is responsible for the storage and manipulation of instances which are generated from the subclasses of *bioObject*. The class *bioObject* itself has the duty to keep free space for the data and methods which belong to all its subclasses. [18]

bioSimulation stores a pointer to an instance of *bioWorld* and several instances of *bioPopulation*. It acts as a counter for cycles in the simulation and it is able to activate simulation cycles. In the default implementation action cycles are just executed until the default implementation is overridden. The action cycle invokes the method NextStep() for the *bioWorld* instance. After that the counter for the number of action cycles is incremented. When the breeding cycle is entered, an invocation of the method Breed() is sent to all *bioPopulation* instances. As the next step the counter for the breeding cycle is also manipulated. In the case of the simulation, which is researched in this thesis, the environment cycle changes only internal counters for the diverse cycles except for the breeding cycle. [18]

The class *bioWorld* monitors all objects which belong to the environment and it is responsible for the execution of action cycles. Because of that it also handles the

perception of the local environment and the determination of intentions which both belong to instances of *bioThing*. It stores a list of all instances of *bioThing* and an extra list of instances that are already dead in the simulation but not yet deleted. It incorporates methods for the setting and manipulation of the positions of *bioThing* instances situated in the world. There are also methods for the removal and deletion of those instances. The execution of the action cycle (i.e. method NextStep()) starts with the copying of all stored *bioThing* instances into a local list (i.e. instance of *bioList*). Then the system tells those instances to get their perception of the local environment. After that it translates those perceptions into intentions and finally deletes all the things that are already situated in the list of dead instances. [18]

A subclass of *bioWorld* is *bioCellWorld* which implements all methods from *bioWorld* that are only defined there. It stores a representation of the world. This representation can be comprised of the number of cells and the size of the window where the world is situated, etc. There are methods for the creation of an empty world with predetermined size, for the setting of instances of *bioThing* to an appointed position, for the removal of such an instance and the finding of a free cell for an instance if no initial position is given. It also has the ability to return information about the local environment (i.e. position contents), it is able to inform instances of *bioWorld* about the transformation of a certain region and it is translating user actions (e.g. mouse click) into coordinates which are situated on the world. [18]

For the storage and manipulation of *bioPType* instances together with their *bioGType* instances responsibilities rest with class *bioPopulation*. The instances of *bioPType* are called phenotypes and represent single agents in the simulation. They are built from genotypes (instances of class *bioGType*). For the simulation

in this thesis a genetic algorithm is chosen which is implemented in *bioPopulation*. Data, which are stored in this class, are phenotype instances, the size of the array where those instances are situated, the quantity of phenotypes of a generation who perform best and get into the next one, the rate at which phenotypes are mutated and the total, maximum, minimum and average fitness. The class creates space and it initializes a population which consists of phenotypes. Those phenotypes have certain genotypes. The class is also responsible for the comparison of the fitness scores of 2 *bioPType* instances and for the sorting process afterwards. The method for the execution of the genetic algorithm first checks up whether the phenotypes are already sorted by their fitness. If they aren't sorted it invokes the sorting method. Otherwise it proceeds with the creation of new space for the next phenotype generation. Best performers of genotypes are directly transferred into the next generation. From the remaining ones 2 parents are selected and a new genotype is generated as crossover of these parents. If there exists a mutation rate, the genotype is mutated after its creation and new phenotypes are built from the changed genotype. [18]

bioThing acts as an interface for all existing objects. Those objects can be active, like agents inhabiting the world, or they are passive, like food that is traded between them. The class stores a pointer which is directed to an instance of *bioWorld* and it stores also an object's position and orientation. [18]

Class *bioAgent* is derived from *bioThing*. The instances of this class are controlled by their internal programs. An instance stores a pointer to its internal program and also an appointed intention which is represented by an integer value. The class is comprised of several methods whereas one of them is a pure virtual one. This virtual method normally has the duty to return the local environment's current state during a run. [18]

There exists also a subclass of *bioAgent* which is *bioPType*. As it has already been mentioned before, the instances of this class are constructed from genotypes and a certain number of them can build an instance of *bioPopulation*. *bioPType* stores the pointer to a genotype instance. Belonging to each *bioPType* instance it also stores an id, the name and the fitness score. It also has the ability to return the genotype value of a phenotype instance. [18]

bioProgram is a pure virtual class and it includes only definitions of methods and no data. It acts as an interface for the communication between any agent and the corresponding program. The methods defined here are implemented in the subclasses which can be, for example, a finite state machine. [18]

The class *bioFSM* is a subclass of *bioProgram* and it implements all the methods which are defined there. For this purpose it uses a finite state machine program which is run via a table. For the table the current state and the input are combined and construct the index. The next state and output is available via the contents of the table at a certain index. Data which are stored in this class are the current table, input and output, the number of bits in a state and the size of the table. The size of the table is calculated by *2* to the power of n, whereas n is the sum of the number of state bits and the number of input bits. The class initializes the table by the calculation of its size and by making space available for it. After that the table can be populated via the transformation of an array into this table. [18]

The instances of class *bioGType* are owned by class *bioPType* and manipulation is executed by *bioPopulation* instances. *bioGtype* stores no data and its instances are used for the construction of new phenotypes using crossover and mutation. [18]

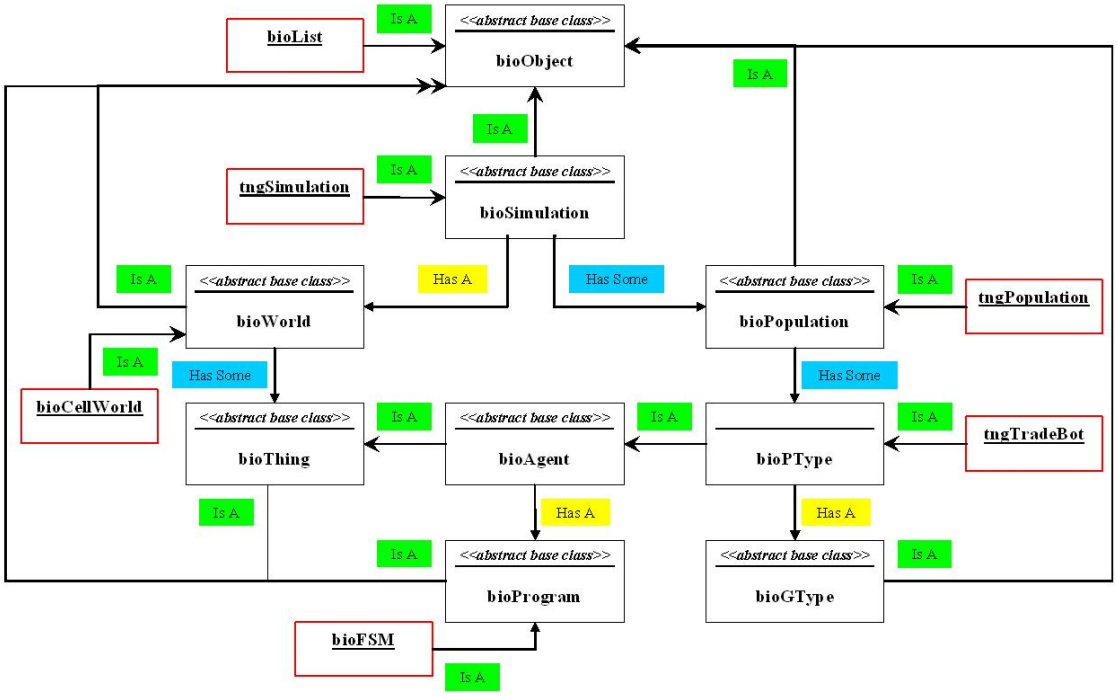

Figure 3.1: Class diagram of the SimBioSys framework including classes of the Trade Network Game [6] [18]

3.1.2 SimBioSys Execution Cycle

In Figure 3.2 you can see the course of events in a simulation cycle of the SimBioSys framework. The first step is the initialization of the simulation in creating a world. Then the simulation creates a population which inhabits the world. After that different cycle loops are entered. The first loop which is executed is the action cycle loop. An agent retrieves information about their local environment and delivers this information to their internal program. The program comprehends how to deal with the environmental information. Next it reads out what action an agent has to perform when they consider the prior input. After all the agents have decided upon their activities, the action cycle

loop is left behind and the environment cycle loop is entered once again. This means that the action cycle loop is situated within the environment cycle loop. In the environment cycle statistical data about the environment is accumulated and saved. Then each agent's fitness is retrieved and the operating figures (e.g. average fitness) are calculated. If processes which are not in collaboration with agent actions occur, it is possible that the environment is modified. After leaving the environment cycle loop, the breeding cycle loop is reentered (i.e. action cycle loop and environment cycle loop are situated in the breeding cycle loop) and a breeding step is executed. At this step at first the phenotypes (i.e. agent programs) pass through a sorting process according to their fitness. Some are elected as parent phenotypes which are then genetically altered corresponding to a certain algorithm. The altered phenotypes then replace the former ones in the environment. [6]

```
int main{

        Initialize world and agent population;

        for (B=0, ···, BMAX-1) {      //Enter breeding cycle loop
                for (E=0, ···, EMAX-1) {      //Enter environment cycle loop
                        for (A=0, ···, AMAX-1) {      //Enter action cycle loop

                                Do agent actions;
                        }

                        Environmental step;
                }

                Breeding step;
        }
        Return 0;
}
```

Figure 3.2: Pseudocode for the SimBioSys framework [6]

3.2 Network Simulation

In this section of network simulation the classes which are derived from the overlying framework are explained and the initial parameters are specified. Next follows a description of how the whole simulation is executed step by step.

3.2.1 Trade Network Game

The Trade Network Game is based on an ACE model which was introduced by Leigh Tesfatsion. This simulation is a combination of evolutionary game play and endogenous partner selection. The individuals which inhabit the system have to develop trade strategies. Those strategies mature all the time within the process of going through different generations of traders. Trade partners are chosen via the presumption of an expected payoff. The agents are able to receive this payoff through the interaction with others. The interaction between two agents is implemented as a prisoner's dilemma game. The simulation was built based on the SimBioSys framework and implemented in C++. [6] [18]
Class *tngSimulation* is the subclass of *bioSimulation*. It stores many of the parameter values used in the trade network game: the number of generations that are produced in the game, the number of trade cycles which are run through in each generation, the number of offers each buyer is able to direct to sellers, the number of offers each seller can have at most on their waiting list, the payoff level that is initially expected by each trader considering other traders, the payoff a trader receives when getting a refusal to their offer, the payoff for an inactive trader, the payoff after mutual cooperation between traders, the payoff after mutual defection, the payoff a trader gets in the case when they defect and the other one is cooperating and vice versa, the number of all traders

in the simulation, the number of pure buyers, the number of pure sellers, the number of buyer-sellers and the number of how many trades have been executed in one trade cycle. Instances of *bioPopulation* are stored as a population of pure buyers, a population of pure sellers and a population of traders who are both buyers and sellers. Furthermore all tradebots, the buyers and sellers, are stored in different lists (i.e. instances of *bioList*). There are methods in this class for the initialization of parameters, for the initializing of generations, the determination of trade partners, implementation of trades, calculation of fitness scores, the evolution of each generation and the allocation of payoffs, which are received from the prisoner's dilemma game. [6] [18]

tngPopulation, which is derived from *bioPopulation*, stores no data and implements only the creation and initialization of a genotype and also the creation of a trader which is an instance from the class of phenotypes. [6] [18]

At last there is the class for all the traders which is called *tngTradeBot* and is a subclass of *bioPType*. This class needs an auxiliary class called *TraderInfo* to store all information concerning the conducting of trades with other traders. This information consists of the last received payoff, the sum of all payoffs gained via interaction with a certain trader, of how many payoffs have been received, the number of refused offers by the appointed trader, the expected payoff, the last actions taken, how many trades have been conducted and the status of an offer. The status can be that of 'no offer', an 'offer made' and a 'rejected offer'. The class *tngTradeBot* itself stores how much space can be used for the finite state machine program to record past actions, the number of bits for internal states of the finite state machine, how many offers can be at most outstanding and how many can be at most accepted, the number of all traders, the currently outstanding, made and accepted offers, the number of trade cycles where

a trader has been inactive, the number of all payoffs and the sum of all payoffs. Furthermore, it stores pointers to instances of the auxiliary class *TraderInfo*. The methods it makes available are implementations for the trade strategy, for the initialization of generations, for the matching of traders, methods for trading itself and for the output of the simulation. [6] [18]

3.2.1.1 Specification of Initial Parameters

There are two types of parameters that have to be set initially in a configuration file (in this case the file is called tng.ini). The first are parameters which describe the virtual environment. Second, there are parameters which belong to the traders. All of them are implemented in class *tngSimulation* and class *tngTradeBot* respectively. Their description can be found in tables 3.1, 3.2 and 3.3. [6] [18]

3.2.1.2 Execution of the Main Program

As a first step an instance of *tngSimulation* is created which invokes method Init(). In this method the parameter values, which are used by the environment, are implemented and printed out. A new generation of traders (which consists of pure buyers, pure sellers and buyer-sellers) is created. Their strategies which are used in the following trading cycles are assigned randomly (the application flow can be seen in Figure 3.3). [6] [18]
Once this is done the generation cycle loop is entered and method InitGen() invoked. Within this method at first the shortly before created traders are configured with parameter values. The list of traders is filled with converted instances from the populations of pure buyers, pure sellers and buyer-sellers. The list of buyers is filled with pure buyers and buyer-sellers and the list of

Parameters of the virtual environment		
Parameter	**Data Type**	**Description**
GMax	positive integer	total number of generations
IMax	positive integer	number of trade cycles in each trade cycle loop
RandomSeed	unsigned integer	seed for pseudo-random number generator
MutationRate	nonnegative number < 1	probability for bit toggling in genetic algorithm
FsmStates	positive integer	number of internal finite-state-machine (FSM) states (i.e. trade strategies)
FsmMemory	positive integer	FSM memory (expressed in bits) for trade partner actions of the past
TraderCount	integer > 1	total number of traders
PureBuyerCount	positive integer <= total number of traders (TraderCount)	number of pure buyers (sum of PureBuyerCount, PureSellerCount and BuySellCount >= TraderCount)
PureSellerCount	positive integer <= total number of traders (TraderCount)	number of pure sellers (sum of PureBuyerCount, PureSellerCount and BuySellCount >= TraderCount)
BuySellCount	positive integer <= total number of traders (TraderCount)	number of buyer-sellers (sum of PureBuyerCount, PureSellerCount and BuySellCount >= TraderCount)
ElitePercentPB	nonnegative number < 1	percentage of elite traders in the subpopulation of pure buyers

Table 3.1: Description of Parameter Specification 1 [6]

ElitePercentPS	nonnegative number < 1	percentage of elite traders in the subpopulation of pure sellers
ElitePercentBS	nonnegative number < 1	percentage of elite traders in the subpopulation of buyer-sellers
Temptation	real number	highest possible payoff from the prisoner's dilemma game
BothCoop	real number	prisoner's dilemma payoff when mutual cooperation takes place
BothDefect	real number	prisoner's dilemma payoff when mutual defection takes place
Sucker	real number	lowest possible payoff from the prisoner's dilemma game
RefusalPayoff	real number	payoff received by trader who has been refused
WallflowerPayoff	real number	payoff received by trader who has been inactive
`Trader's parameters`		
BuyerQuota	nonnegative integer <= PureSellerCount + BuySellCount	quota for buyer offers a buyer is able to place

Table 3.2: Description of Parameter Specification 2 [6]

SellerQuota	nonnegative integer <= PureBuyerCount + BuySellCount	quota for offers sellers are able to accept
InitExpPayoff	real number	payoff which is initially expected from the traders

Table 3.3: Description of Parameter Specification 3 [6]

sellers is filled with sellers and buyer-sellers. By invocation of method Init() from class *tngTradeBot* traders get their buyer offer quota (which is the same for pure buyers and buyer-sellers) and their seller acceptance quota (which is the same for pure sellers and buyer-sellers). Furthermore, an information list is created to store all information about other traders and waiting lists are constructed by each seller. Those waiting lists are used for the offers which are received from the buyers. For the beginning each trader is endowed with the same expected payoff level towards others except for a trader's own payoff level. This is decremented by 1 so that the trader cannot choose himself for a trading interaction. The generation cycle loop is repeated when all the other processes (i.e. trade cycle loop, environmental step, evolution step) have been executed. [6] [18]

The next step is the entering of the trade cycle loop. The first action is the assessment of trading partners for the individuals by using method MatchTraders(). The precondition for this method is that each trader has a certain expected payoff which was stored for every single one of the others. To determine the trading partners a variation of the 'Gale-Shapley deferred acceptance mechanism' (for more information consult [5]) is used. This method is called 'deferred choice and refusal mechanism' (DCR mechanism). All methods called on in the process of

the DCR mechanism are implemented in class *tngTradeBot*. First the buyers use method PrepareOffers() to set the offer status of all other traders which could be utilized as trading partners to 'NO_OFFER' (i.e. no offer of this trader was directed to them in the current trade cycle). The number of outstanding offers (i.e. without feedback from potential trading partners) and the number of all offers they have already made in this trade cycle are set to zero. With SubmitOffer() all buyers direct their offers to sellers of whom they can expect the highest payoffs (i.e. potential trading partners are sorted according to the expected payoff) and which have status 'NO_OFFER'. One offer per seller is placed until the maximum number (i.e. buyer offer quota) is reached. The offer status of each seller who has received an offer is changed to 'OFFER_MADE'. The offer itself is added to a seller's waiting list via TakeOffer(). All sellers who have received offers invoke AcceptOffers(). There the offers are also sorted depending on payoffs which are expected from the sellers. Offers which produce the highest payoffs are accepted until the maximum number (i.e. seller acceptance quota) of offers is reached. The remaining offers are refused with OfferRejected(). If a buyer receives a refusal, they have to store the refusal payoff. After that they make an upgrade of the payoff they expect from the seller, who has sent the refusal. At last he changes his status to 'OFFER_REJECTED'. As soon as the buyer has got a negative response to a sent offer they can decrement the number of outstanding offers. As long as there are buyers who still have offers to submit and sellers who haven't rejected them yet the process of the making of offers is repeated. After the matching process is finished, the traders who did not accept or manage to place an offer receive a wallflower payoff for being inactive in this trade cycle. All offers on the waiting list that weren't refused are accepted and MatchTraders() is finished. Postconditions are that refusal payoffs

and wallflower payoffs have been stored and that buyer offer quota and seller acceptance quota haven't been exceeded. [6] [18]

The second point in the trade cycle is the performance of trading interactions with the determined trading partners (method Trade()). The precondition is that all trading partners have already been assessed previously. It is the duty of the sellers to start the prisoner's dilemma games with each of the buyers on their waiting list by invocation of PlayPD(). This results in a call of method MediateTrade() for each buyer on the waiting list. The method is implemented in class *tngSimulation*. The first action is to get a reaction from each of the traders. Those reactions are based on the state of the finite state machine. If a certain trader already had interactions with another one, the last FSM state and outcome of the prisoner's dilemma game were recorded previously. Now this influences the present game, i.e. traders develop trade strategies. After that the actions' payoffs according to the traders decisions are retrieved. The information about those payoffs is passed on to sellers via Sell() and to buyers via Buy(). Each trader has to add the current payoff to its total payoff sum. They have to increment the total payoff count, they add the achieved payoff to the payoff sum which belongs to the appointed trading partner, they increment the payoff and trade count which concerns their counterpart, they record the decision which was taken in the PD game and at last they update the expected payoff for this certain trading partner (invocation of method UpdateExp()). The expected payoff is updated every time a trader receives any payoff from another one. This can be either a payoff received trough trading interaction via Buy() and Sell() or a refusal payoff, if the trading partner has refused to trade via OfferRejected(). Postconditions for the accomplishment of this step in the trading cycle are that all the trading pairs have executed their trades and all data concerning trading

interactions (e.g. payoff, action) have been saved. [6] [18]

After the trade cycle loop is finished the program gets back into the generation cycle loop, where at this point an environmental step is executed through invocation of method AssessFitness(). The only precondition is that each trader has to have a positive count of total payoffs. Each trader's fitness is calculated via CalcFitness(). The average payoff each trader has received in all trading cycles is figured out. The fitness score, which is returned by the program, is 2 to the power of the average payoff. After that method Dump() from class *tngTradeBot* is called within AssessFitness(). This method has to print out all data about the traders themselves and about all traders they had interactions with. Information a trader has to display about themselves, is their identifiers, the sum of total payoffs, the number of wallflower payoffs received and their fitness score. Information they emit for traders they interacted with is the list number and the identifier of those traders, the payoff sum of all contacts to a certain trader, their refusal count, the expected payoff, the trade count and the previous actions the trader has taken in matters of former interactions. At last the subpopulations of pure buyers, pure sellers and buyer-sellers are sorted according to their fitness score by the calling of method SortByFitness(). This method belongs to class *bioPopulation*. After the leaving of AssessFitness(), another method Dump() (this time it belongs to class *tngSimulation*) is invoked to print out the resulting fitness scores for all the subpopulations and the population as a whole. [6] [18]

Next there follows an evolution step via invocation of method EvolveGen(). One precondition is that all the traders of this generation have been sorted according to their fitness score. Another one states that the number of elite traders and the predefined mutation rate have to be generally accepted. At first all elements are removed from the lists of all traders. This also includes

the lists of buyers and sellers. Then method Breed() from class *bioPopulation* is called for every subpopulation (i.e. pure buyers, pure sellers, buyer-sellers). This method sorts all the instances according to their fitness, if they haven't been already sorted, and it allocates space for a new generation. As the next step a predefined number of traders, those who are the fittest, are transferred to the next generation without any changes in their trading strategies. For the remaining individuals a genetic algorithm is used. 2 parents are chosen, a crossover is executed and the resulting children are applied as new traders in the next generation. It can be followed that trade strategies that perform best are kept as they were. Unsuccessful strategies are revoked and new strategies are derived from crossover and mutation of former ones. They are adopted to see how they will perform in the next generation cycle. [6] [18]

```
int main{

        Init();  //Construct initial trader generation with random trade strategies

        for(G=0,…,GMAX-1){  //Enter generation cycle loop

                InitGen();  //Each trader is configured with user-supplied parameter-
                            //values (initial expected payoff levels, resource quotas)

                for(I=0,…,IMAX-1){  //Enter trade cycle loop

                        MatchTraders();  //Determine trade partners given expected
                                         //payoffs, record refusals and wallflower
                                        //payoffs

                        Trade();  //Carry out trades and record trade payoffs

                        UpdateExp(); //Update expected payoffs using newly recorded
                                     //payoffs
                }

                //Environmental step

                AssessFitness();  //Assess and output information about traders

                Dump();  //Output fitness statistics for the current trader
                         //generation

                EvolveGen();  //Evolution step: obtain new trader generation with
                              //evolved trade strategies whereas the evolved
                             //generation enters generation cycle loop
        }
        Return 0;
}
```

Figure 3.3: Pseudocode for the Trade Network Game [6]

4 Research Problem in Testing

There are three different groups; namely pure buyers, pure sellers and buyer-sellers. We want to know how they connect and build networks with a different number of individuals in each of the groups. A second issue we want to cover is the influence of past memory on those connections. For this layout there are only changes made in the numbers of agents within their single groups and in the memory of the finite state machine. All other parameters remain the same in each of the scenarios. I thought this approach of testing to be the best way to achieve our objective. The goal of the research is to test the simulation first on how the networks are built. Second it is tested on the emergence of behaviour. So we should be able to find out how a society evolves. We can see how individuals reach a positive outcome and how they keep links within the community which have resulted in positive experiences.

The first focus of this thesis is set on the number of pure buyers, pure sellers and buyer-sellers. The second focus is on how far the memory of the finite-state-machine algorithm reaches. There are 21 scenarios which were implemented for the testing. The same grouping of agents is tested with 1, 10 and 16 bits of memory for the finite-state-machine.

The next step is to show how the output of the program is reached in single steps. The first step is that each of the agents, that can play the role of a buyer,

chooses one of the sellers and makes an offer to the chosen one. As an example a market is chosen with 3 pure buyers, 3 pure sellers and 3 buyer-sellers. Assuming that the symbol $\rightarrow$ signifies that a certain buyer makes an offer to a certain seller (i.e. *is making an offer to*), the first offering cycle could look like this:

$$PureBuyer1 \rightarrow BuyerSeller2$$
$$PureBuyer2 \rightarrow BuyerSeller1$$
$$PureBuyer3 \rightarrow BuyerSeller1$$
$$BuyerSeller1 \rightarrow PureSeller1$$
$$BuyerSeller2 \rightarrow BuyerSeller1$$
$$BuyerSeller3 \rightarrow PureSeller3$$

Next, the end of the first offering cycle starts and all those agents that play a seller's role have to tell the buyers whether their offers are accepted or refused. Buyers that get rejected have to send new offers to different sellers. The process of sending new offers triggers off the second offering cycle in the first trading cycle. Also buyers whose offers were accepted in the previous offering cycle have to participate in sending the same offers again. By doing so sellers are able to find out whether there could be a better offer in the new ones. This renders it now possible to bring forth the refusal of an earlier accepted offer. In this example of the first trade cycle, which takes place in the first generation of traders, the offering cycle has been performed six times.
The resulting offering cycles can look like this:

1. offering cycle:
   PureBuyer1 $\rightarrow$ **BuyerSeller2** | **accepted**

PureBuyer2 → **BuyerSeller1** | **accepted**

PureBuyer3 → **BuyerSeller1** | **refused**

BuyerSeller1 → **PureSeller1** | **accepted**

BuyerSeller2 → **BuyerSeller1** | **refused**

BuyerSeller3 → **PureSeller3** | **accepted**

2. offering cycle:

   PureBuyer1 → **BuyerSeller2** | **acc** | **acc**

   PureBuyer2 → **BuyerSeller1** | **acc** | **acc**

   PureBuyer3 → **PureSeller1** | **ref**

   BuyerSeller1 → **PureSeller1** | **acc** | **ref**

   BuyerSeller2 → **PureSeller1** | **acc**

   BuyerSeller3 → **PureSeller3** | **acc** | **acc**

3. offering cycle:

   PureBuyer1 → **BuyerSeller2** | **acc** | **acc** | **acc**

   PureBuyer2 → **BuyerSeller1** | **acc** | **acc** | **acc**

   PureBuyer3 → **PureSeller3** | **acc**

   BuyerSeller1 → **BuyerSeller3** | **acc**

   BuyerSeller2 → **PureSeller1** | **acc** | **acc**

   BuyerSeller3 → **PureSeller3** | **acc** | **acc** | **ref**

4. offering cycle:

   PureBuyer1 → **BuyerSeller2** | **acc** | **acc** | **acc** | **ref**

   PureBuyer2 → **BuyerSeller1** | **acc** | **acc** | **acc** | **acc**

PureBuyer3 → **PureSeller3** | acc | acc

BuyerSeller1 → **BuyerSeller3** | acc | acc

BuyerSeller2 → **PureSeller1** | acc | acc | acc

BuyerSeller3 → **BuyerSeller2** | acc

5. offering cycle:

   PureBuyer1 → **PureSeller3** | acc

   PureBuyer2 → **BuyerSeller1** | acc | acc | acc | acc | acc

   PureBuyer3 → **PureSeller3** | acc | acc | ref

   BuyerSeller1 → **BuyerSeller3** | acc | acc | acc

   BuyerSeller2 → **PureSeller1** | acc | acc | acc | acc

   BuyerSeller3 → **BuyerSeller2** | acc | acc

6. offering cycle:

   PureBuyer1 → **PureSeller3** | acc | acc

   PureBuyer2 → **BuyerSeller1** | acc | acc | acc | acc | acc | acc

   PureBuyer3 → **PureSeller2** | acc

   BuyerSeller1 → **BuyerSeller3** | acc | acc | acc | acc

   BuyerSeller2 → **PureSeller1** | acc | acc | acc | acc | acc

   BuyerSeller3 → **BuyerSeller2** | acc | acc | acc

So the resulting trading partners in the first trade cycle of generation number 1 are the connections seen in the sixth offering cycle. Now, that all trading partners for the first trade cycle are selected and accepted, the simulation is able to continue to the second trade cycle of the first generation.

In the next subsections there will follow the outcomes for all executed simulations with all those different parameters, which have been chosen to be altered. The outcomes will be interpreted in order to determine how the different networks, which emerge from those trades, evolve and how the behaviour of all traders towards their chosen partners changes over time.

4.1 Finite State Machine Memory = 1

In the case of the memory of the finite state machine algorithm being just one bit, you can tell from the beginning that the agents in the simulation will not really be able to take their past experiences into consideration.

4.1.1 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 3

This is the case of an equal number of pure buyers, pure sellers and buyer-sellers. As buyer-sellers can play both as buyers and sellers there are 6 possible buyers and 6 possible sellers. Not all 10 trade cycles are executed in each generation. Most of the time the trading stops after trade cycle number six is reached.
Just in generation number one, which goes through all ten trade cycles, you are able to observe similarities in the networks. They appear in-between trade cycles number 3 and 6. In this generation also cycles number 9 and 10 are completely the same. This can be viewed as a coincidental happening.
The payoff the agents receive is negative most of the time. Only traders from the group of pure sellers sometimes get small positive payoffs or remain neutral in their earnings. But the positive share is that small so that it doesn't really affect the average payoff of all traders. The average payoff remains negative throughout all generations.

You can already derive from the payoff which behaviour the agents display in this sample. In the first 2 generations there is mostly defection on all sides. Just rarely one of the pure sellers cooperates. After the first two generations the sellers' side always defects. On the buyers' side pure buyers begin to cooperate. At first there are just some pure buyers which cooperate. But as time passes by a level of full pure buyers' cooperation is reached.
It can be followed that in this sample only random networks are built. This is due to no reminiscence of previous actions and the prevailing negative payoffs received by all the traders.

4.1.2 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 3

Here there could be in total 9 traders on the buyers' side but just 6 traders on the sellers' side. As we restrict the buyer- and seller-quota to 1, the number of trades that can be carried out in each of the trade cycles is limited to six. Also in this case not all possible trade cycles are run through in each generation. But there are more generations than in the previous example that go through all ten cycles.
There are also more weak similarities; they occur in nearly every generation. In generation number six the networks of trade cycles number 6, 7 and 8 look similar. The networks of trade cycles number 9 and 10 are completely the same. Because this happens only in this generation, it can be followed that this pattern just arises by accident.
The payoff for executed trades is negative nearly all the time for all participating agents. As in the previous sample just pure sellers have positive or neutral outcomes. This happens less often than in the case of all agents being the same number in every group. The average payoff for all of them is negative in

all generations. The negativity is higher than for the sample in the previous subsection.

As before, the behaviour emits nearly only defection; there are just a few single cooperations which occur in each of the generations. This attitude, the agents are holding, remains constant through all generations.

The result of the testing of this sample is similar to the previous one. Networks are also built at random. One interesting thing is, however, that there are more similarities than before although the payoffs are more negative. Do those things happen just coincidentally or is the small memory sufficient for remembering some better outcomes?

4.1.3 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 3

There could be 9 traders on the sellers' side but just 6 on the buyers'. So the maximum of trades, which is carried out in each of the trade cycles, is six. All 10 trade cycles are passed through in each of the 10 generations.

From 1 to 6 weak similarities can be seen in the networks in each generation. The similarities get more numerous and overlapping in the middle of the generations, i.e. around generation number 6. When reaching higher generations, the similarities diminish again. This goes on until there are no similar connections in generation 10.

Average payoff for all traders is negative in all the generations. As in the other examples sometimes pure sellers get slightly positive or neutral results in their trades. This does not affect the negative average.

Behaviour is mostly defective; some cooperation arises but it diminishes over time. The pattern is that, at the beginning of each generation in the first trade cycles, no cooperation can be found but it appears again in higher generations.

The networks' similarities mostly emerge for buyer-sellers which interact with other agents. Still it can be followed that the networks are built at random and that the similarities occur accidentally.

4.1.4 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 6

9 trades are possible on both the buyers' and the sellers' side; all trade cycles are run through in every generation.
Just a few single similarities in the networks occur at random, the most can be seen in generations 1 and 8.
Again the average payoff for all the traders is negative. In the first 3 generations pure buyers and buyer-sellers just receive negative results; only the pure sellers have some neutral or slightly positive outcomes. This does not really affect the negative average for all of them. From generation number 4 on all trader groups get strictly negative payoffs.
The agents show mostly defective behaviour. Sometimes pure buyers or some single buyer-sellers produce cooperative behaviour but this diminishes over time until there are no pure buyers left that show cooperation.

4.1.5 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 6

There could be 9 trades conducted on the buyers' side and 12 trades executed on the sellers' side. So the maximum number of trades is 9 in this case. Every trade cycle in each generation is used for the performance of trades.
In the case of a higher number of agents on one side there are much more similarities in the designed networks than in the previous cases. The thing that catches someone's eye here is, that most similarities are to be found in the

interactions between buyer-sellers and buyer-sellers.

Pure sellers only receive positive average payoffs in each of the generations. But the average result for pure buyers and buyer-sellers is constantly negative. The positive outcome of the pure sellers is not able to exceed the negative one of the remaining agents. That is why the average for all traders remains negative all the time. On some rare occasions the payoff for buyer-sellers and pure buyers has a single neutral or positive element which already shows some tendency to achieve better results.

Behaviour of the traders is a mix of defection and cooperation with slightly less cooperative behaviour until the end of generation number 2. After that a little bit more cooperation than defection can be witnessed. All buyer-sellers cooperate and all the others don't. In generation 6 you are able to observe some small differences. Buyer-sellers start to show defective behaviour again. As a counterpart some of the pure buyers have a rise in cooperation. Over time this trend reverses to the previous state of cooperation by buyer-sellers and mostly defection by pure buyers.

Although there are many more similarities and although it seems that at least buyer-sellers as traders choose their trading partners according to the memory of previous interactions, you cannot recognize any pattern that is pursued in all the generations.

4.1.6 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 6

It would be possible to have 12 trades on the buyers' side and 9 trades on the sellers' side; so there are at most 9 trades which are permitted in each trade cycle. The maximum number of trade cycles is carried out in each generation.

This case is completely different to the previous case, with the higher number of

sellers. Here we have just a few single similarities. Those single similarities can only be observed in 5 of the 10 generations. Just once, in generation number 2, the network of trade cycle number 10 is a reduced network of trade cycles number 7, 8 and 9. This seems to happen incidentally because in trade cycle 10 there is just one trade interaction (i.e. just one link in the network) left.
Received payoffs begin to come into being negative and they remain negative for pure buyers and buyer-sellers. Pure sellers start with a positive result but slide down into negativity after generation number 1. So the average for all traders is negative all the time.
Concerning the behaviour at the beginning of the generations there is mostly defection mixed with a few cooperations. Defective behaviour increases over time until there are just 2 to 6 traders per generation which show cooperation.

4.1.7 Pure Buyers = 6, Pure Sellers = 6, Buyer-Sellers = 3

9 trades can be conducted on each side of the market and all possible trade cycles are carried out.
A few similarities can be found in each of the generations but not too many. In most cases those similarities can be said to occur randomly. Just in one generation you are able to find 2 trade cycles which show an approximation to the building of the same network (2 differences in 4 links). In this case you cannot see any likeness of a certain group of traders to build similar connections throughout the generations.
Pure buyers and buyer-sellers only receive negative payoffs from their trading activities. Just the pure sellers show a tendency to more positive or neutral results. In the first 7 generations, however, their average payoff remains negative. In the last 3 generations the payoff rises again to a positive status but it cannot

outrun the negative numbers earned by the other agents. Because of that the average result for all agents is negative in all generations.
The agents exhibit mostly defective behaviour with a few cooperations in each trade cycle. This behaviour does not vary over time.

4.2 Finite State Machine Memory = 10

The memory of the finite state machine algorithm is augmented to 10 bits. This leads us to the assumption that traders will tend to establish similar trading connections if it is compliant with received payoffs.

4.2.1 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 3

There are just some generations which run through all 10 trade cycles; generations number 3, 6, 7, 8, 9 and 10. Generations which already stop after trade cycle 7 are generations 1, 2, and 5. Generation number 4 stops after reaching trade cycle 8.
Similarities do not arise in any of the generations. This just happens in those which go through all possible trade cycles. Although there aren't many of them you are able to detect that, by the time a later trade cycle of a generation is reached, the networks become more and more the same. In this sample in generation number 3 trade cycles 9 and 10 are nearly the same. Cycle number 10 is a reduced version of trade cycles 9 and 7. In generation number 7 trade cycle 10 is a reduced version of cycle number 7. In generation number 8 trade cycles 7 and 8 only differ in one link out of 4. From cycles 7 to 10 you are always able to find the same connections between two buyer-sellers. In generation number 9 cycles 9 and 10 are completely the same. They consist of 3 links. In generation

number 10 cycles 8 and 9 are the same. They consist of 4 links. Trade cycle number 10 is a reduced version of those two cycles.
The average payoff in each of the generations is always negative for all traders. In some generations pure sellers and buyer-sellers reach a positive result in average for themselves. But it is not high enough to change the average which was built for all of them.
The agent's behaviour is a mix of defection and cooperation. Pure buyers tend to cooperate while all the others defect at the beginning. Buyer-sellers sometimes show cooperative behaviour which returns them some positive results concerning the payoffs. Pure sellers show a stable mix of cooperative and defective behaviour all the time. This often gives them some advantage for the reception of positive payoffs from their trading interactions.
In this sample you are already able to see that in some cases agents establish the same trading networks although there are still a lot of negative payoffs returned.

4.2.2 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 3

Generations number 1, 2, 3, 8, 9 and 10 run through all possible trade cycles. Generations number 4 and 7 stop after cycle 5. Generation number 5 stops after cycle 8 and generation number 6 goes on until trade cycle number 9.
As before, you can only find similarities in those generations which run through all possible trade cycles. In generation number one trade cycles 9 and 10 are completely the same (consisting of one link). This link can also be found from trade cycles 2 to 8. Generation number 2 shows the same networks from trade cycles number 8 to number 10 (composed of 1 link). You can also find this connection already from cycles 2 to 7. Generation 3 has the same networks from trade cycles 8 to 10 (1 link). Those graphs are reduced versions of cycles 6 and

7. In generation number 8 trade cycles 9 and 10 are the same (1 link). They are reduced versions of cycles number 5, 7 and 8. The trade cycle networks (7, 8, 9, 10) in generation number 9, that are completely alike, are composed of 3 connections. They are reduced versions of cycles 4, 5 and 6. In generation number 10 trade cycles 8 and 10 are completely the same (3 links). They are reduced versions of trade cycle number 6.

In the first 2 generations you are able to observe some positive results for pure buyers and pure sellers. After that pure buyers step into a negative range and pure sellers keep their average payoffs positive until reaching generation 5. Until generation 9 the average results for all trading groups are negative. Then the pure sellers again change to positive. The average result for all agents together remains negative through all generations.

Concerning the behaviour at first you can see a mix of cooperation and defection with a little bit more on the defective side. After generation number 4 there's mostly defection with hardly any cooperation left. From generation number 8 more cooperation occurs again. At the end in generations 9 and 10 a total change in behaviour can be observed. More cooperation than defection takes place.

As can be seen in this case, somehow the behaviour of all traders and the establishment of connections in the network depend on each other. You can also see that the more memory there is at the traders' disposal the more they are likely to cooperate. With more memory and cooperation they tend to keep the same trading relationships.

4.2.3 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 3

This testing sample is going through all trade cycles in each generation.

The similarities here aren't as obvious as in the samples for a finite state machine

memory of 10 before. It is more like in the case of this constellation of trader groups with a memory of 1 bit. There are quite a few more similarities but it is difficult to find them out on first sight. So for example in generation number 1 you are able to see 2 connections which are the same in trade cycles 8, 9 and 10 out of 4 to 6 links.

For the reception of payoffs it is quite the same as in the case with memory of 1. The overall average is remaining negative for all generations. The only thing you are able to see is that sometimes pure sellers have a positive result in average for their group.

Behaviour is also nearly the same as in the sample with a finite state machine memory of 1; just a little bit more cooperative behaviour can be observed but there is still a lot of defection remaining.

4.2.4 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 6

There are a few similarities in the networks in all generations except for generation number 1. In generation 5 you can find 3 connections out of 4 connections which are the same in trade cycles 9 and 10. You can also see 2 of those 3 links in generation 8 and one of them in generations number 5 and 6. Trade cycle 10 of generation 7 is a reduced version of cycles number 8 and 9. In generation 8 the only difference of trade cycles 9 and 10 (composed of 3 links) is one link. You are able to see the network which is established in trade cycle 10 of generation number 9 also in cycles 5 and 8 and half of it in cycles 1, 6, 7 and 9. In generation 10 there are 3 links which are the same in trade cycles number 6, 7, 9 and 10.

Concerning the traders' payoffs the average of pure buyers is negative in each of the generations, the average of pure sellers is positive in each of the generations and the average of buyer-sellers is negative in all of them. The positive result of

the pure sellers cannot exceed the other negative ones. So the overall average is negative.

At the beginning of the generations there is mostly defection but you are able to observe more cooperation than in the case with a memory of 1 bit. Cooperative behaviour grows over time but, as soon as generation number 7 is finished, defection increases a little bit again.

4.2.5 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 6

The similarities that can be seen in this case aren't too numerous; it looks like there are less of them than in the sample where memory of the finite state machine algorithm is just one bit. However, more of them appear in the last trade cycles of each generation. The similarities also mostly appear in trading interactions between buyer-sellers.

The overall traders' payoff average is negative. Pure buyers and buyer-sellers just receive negative payoffs in average. Pure sellers get some positive and some negative results. So in this case you are able to see more negative results than in the sample with memory being just one.

There is a mix of defection and cooperation which is nearly the same as in the case of a memory of 1. However, you are able to observe slightly more defective behaviour. From this you can derive that the more defection happens the higher become the negative payoffs which are received by the agents. Cooperative behaviour mainly comes from buyer-sellers. They are also those traders which tend to keep their trading relationships.

4.2.6 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 6

In this sample you are able to find some similarities in the networks on first sight. The first time, where the agents decide to keep the same trading relationships in the last few trade cycles, happens in generation number 4. There the cycles 8, 9 and 10 are completely the same (consisting of 3 links). This also occurs in generation number 7 (1 link). This connection is a reduced version of trade cycles 5 to 7. In generation number 5 cycles 7, 8 and 9 look similar and trade cycle number 10 is a reduced version of cycles 3 to 9. The same is true for trade cycle 10 in generation 9, but only for the cycles from number 6 to number 9.
In the first generation the averages of all trading groups are negative. Beginning from the second generation this occurs only for pure buyers and buyer-sellers; pure sellers receive a positive average payoff. However, the payoff is not as high as to change the overall average payoff to a positive value. This remains negative in all generations.
The agents' behaviour in the first generation is mostly defective but after that cooperation increases until a stable mix of defection and cooperation is reached. Cooperation can be primarily found on the buyers' side and defection is mainly on the sellers' side.

4.2.7 Pure Buyers = 6, Pure Sellers = 6, Buyer-Sellers = 3

In generation 3 you can see that the networks of trade cycles number 8, 9 and 10 are very similar. There are 3 to 4 connections which are the same out of 5 to 6 links. It is possible to find some similarities in the cycles 6, 7 and 8 of generation number 4, in trade cycles 7, 8 and 9 of generation 5 etc. So to speak, you can find them in each of the generations. This begins in generation number 3.

The first generation starts with a negative average payoff for all trading groups. In generations number 2 and 3 the pure sellers' average reverts to positive and the overall average remains negative. In the next generation not just the pure sellers but also the buyer-sellers receive a positive average result but the overall average is still negative. This changes in generations number 5 and 6. There the overall average reaches a positive status although just the pure sellers keep a positive result for themselves. All the others receive negative ones. From generations number 7 to 10 the average for all the groups themselves is the same but the average for all of them together shrinks to a negative payoff again.
Behaviour triggers off similarly as in the sample with a finite state memory of 1 bit. However, cooperation increases over time until defective behaviour nearly disappears. Defection just remains at first on some occasions and then it rises again to a stable mix on the sellers' side.

4.3 Finite State Machine Memory = 16

Here you can find the tests with the most memory available for the finite state machine algorithm in this case. It is possible to observe a tendency to form alike networks of trading interactions and a liability to cooperative behaviour.

4.3.1 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 3

Starting already in the first generation, trade cycles 7 and 8 (consisting of 4 links) and 9 and 10 (consisting of 3 links) are the same. The latter two are reduced versions of the first two. In generation 2 and trade cycles 9 and 10 you are able to see 3 alike connections out of 4, which is also true for generation number 3. Trade cycles 9 and 10 of generation 4 are the same (3 links) and a

reduced version of cycle 7. Trade cycles 8 and 10 of generation 5 are alike (3 connections), cycle 9 is equal except for 1 single link. They are also reduced versions of cycles 4 and 7. A similar pattern can also be seen in the rest of the generations.

Concerning the reception of payoffs in the first two generations the traders start with a negative overall payoff whereas the pure sellers' average is already positive. From the third generation onwards the overall average payoff keeps a positive level. Pure sellers and buyer-sellers are positive in average in generation number 3. After that pure buyers and pure sellers receive positive average results and buyer-sellers get negative ones.

You can observe mostly cooperative behaviour throughout the generations. In generations number 1 and 2 there are just a few single pure buyers and pure sellers which defect. From the third generation on there are only pure sellers with defective behaviour.

4.3.2 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 3

Considering networks, this is one of the best examples for traders which take their previous trading interactions into account. Traders keep those partners in mind, they are satisfied with, and keep up their relationships. Already in generation 1 trade cycles 4 and 5 are the same (consisting of 4 links) and trade cycles 6 to 10 are also completely alike (3 links) whereas they are reduced versions of the previous two cycles. Generation number 2 shows completely the same networks in each trade cycle as generation number 1. In generation 3 the cycles from 5 to 10 are the same (consisting of 4 links). Similar patterns can be seen for the rest of the generations.

A phenomenon that is interesting in this case is that, as soon as payoffs are

concerned, the trading groups receive many negative average payoffs. This is the case because many of the traders, which aren't taken as trading partners, are valued with a negative expected payoff from the beginning. So a lot of positive payoffs is needed to get out of negativity. From generation 1 to 5 the average that is received by pure buyers and buyer-sellers is negative and the average for pure sellers is positive. In the fifth generation the positivity of pure sellers is that high that also the average for all groups together is positive. In generations 6, 7 and 8 also the buyer-sellers' average is positive and so the total average is also positive or neutral. Generation 9 shows the same pattern as the first few generations which is also true for generation number 10 except that the total average there is neutral.
You can find a lot of cooperation from the beginning with some defections which diminish over time. From generation number 6 the behaviour shifts to total cooperation. This drives the average into a positive level at last.

4.3.3 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 3

As there aren't many similarities in the networks of the other test versions with lower finite state machine memory you cannot find a total equality like in other samples with the same memory. Still it is obvious that, with a rise in memory, you can also find a rise in the alikeness of networks. In generation number 2 trade cycles 9 and 10 have 5 identical connections out of 6. The same is true for generation 3. This pattern can be tracked throughout all the generations. In generation 10 you can find the similarities already from trade cycles 7 to 10.
Considering the reception of payoffs the only negative overall average can be found in generation number 1. After that the average for all trading groups together remains positive. In half of the cases the pure buyers' average is negative

and the pure sellers' and buyer-sellers' average is positive. In the other half all of them receive positive results.
The traders start with a mix of cooperation and defection. More agents are on the side of cooperative behaviour. Then defection at first disappears from the buyers' side and increases a bit on the sellers'. After that it vanishes from both sides. In the last generation behaviour reaches a level of total cooperation.

4.3.4 Pure Buyers = 3, Pure Sellers = 3, Buyer-Sellers = 6

The networks of each generation do not converge to total equality but you are able to find many similarities. Beginning with generation number 1 there are 4 to 5 identical connections from trade cycles 5 to 10 out of 6 to 8 links. In generation two the networks of cycles 7 to 10 consist of 7 connections whereas at least 5 links are the same. This also goes on in the rest of the generations; in generation 7 trade cycles 9 and 10 are completely the same.
In the first 3 generations the average payoff for pure buyers is negative, the average for pure sellers is positive and the average for buyer-sellers alternates from negative to positive and then to neutral. So in generation 1 and 3 the overall average is negative and for all the other generations it is positive. From generation number 4 all averages for every group are positive.
Concerning the agents' behaviour there is mostly cooperation at the beginning with just a few defections until generation number 5. After that behaviour shifts to total cooperation.

4.3.5 Pure Buyers = 3, Pure Sellers = 6, Buyer-Sellers = 6

You can find more similarities in this case than in the sample with less finite state machine memory. However, they aren't too obvious because there are just a few of them hidden in many more connections. The most similarities here can be seen in trading interactions between buyer-sellers.

The overall average payoff is always negative except for generation number 6 where it is positive. Pure buyers get a negative average payoff in each of the generations, pure sellers have negative results in the first few generations and then they shift to positive. Buyer-sellers receive mostly negative and sometimes positive averages.

You are able to see a mix of cooperation and defection whereas cooperative behaviour rises a little bit over time. In this sample not just the buyer-sellers show a tendency for cooperation but also on some occasions the pure sellers in the last few generations.

4.3.6 Pure Buyers = 6, Pure Sellers = 3, Buyer-Sellers = 6

In generation number 1 the network of trade cycle 10 consists of 4 connections and is a reduced version of cycle number 9. Generation 2 has similarities in cycles number 5, 7, 8, 9 and 10. There are 2 to 3 equal connections out of 5. Networks which are completely the same can be found in generation 3 in trade cycles 9 and 10 (consisting of 2 links). These networks are reduced versions of cycles 3, 4, 5, 6 and 8. In this testing sample you can see generations without many similarities and also generations with lots of them. This alternates over time.

Considering the payoffs all the averages for each of the groups are mostly nega-

tive. Just in two generations the pure sellers manage to get a positive result. So the overall average payoff is also negative in each of the generations.
Behaviour starts with a balanced mix of cooperation and defection. After that cooperation rises and then diminishes again until defection prevails.

4.3.7 Pure Buyers = 6, Pure Sellers = 6, Buyer-Sellers = 3

In every generation you are able to find some similarities. Beginning in generation 1 you can see 3 alike connections out of 5 to 6 links in trade cycles 6 and 7. There are also 3 equal connections from networks which consist of 4 links in cycles 8, 9 and 10. Trade cycle 10 of generation 2 is a reduced version of cycle number 9 and has also got some similarities with cycle 8. As the pattern in those 2 examples it goes further on in the other generations. There are more equal connections in every second generation but there are also some in the other ones. The overall average for all groups together remains negative throughout every generation. The average payoffs for pure buyers and buyer-sellers are always negative. Only the pure sellers receive positive results on some rare occasions.
From the beginning until the end of the generations you can find a mix of defection and cooperation. At first there is more defective behaviour. Then cooperation rises until it reaches a slightly higher level than defection.

4.4 Conclusion

The first goal of this research was to determine how networks are built when the links of these networks are established via interactions of individuals. In this thesis the interactions took place on a simulated marketplace. Concerning networks the building process was not the only issue which was investigated. We

also wanted to know whether the individuals tended to keep their connections and what were the requirements for keeping them. The second goal was to test the simulation on emergence of behaviour. Here the only two possible stati were cooperation and defection. We wanted to find out when there was more cooperation or more defection or just a mix of the two.

The results were interpretations of the output of the simulation. The program showed all trading interactions between all the agents, what payoff each of the agents had received for a certain trading relationship and what behaviour the agents have had in an interaction. To have an overview of the networks which were built in each of the generations, I plotted all the graphs on paper. Each generation was drawn on an A4 page.

The results were the following; with higher memory the individuals, which interacted in the simulation, tended to keep their relationships. So you can tell that, at first with hardly any memory, the networks were built randomly in later trading cycles of one generation cycle. The more memory they had the more you could observe a convergence to a certain graph in later trading cycles of a generation cycle. Whether a graph really converged to a certain picture was also dependent on the payoff. Agents wanted to keep the relationships which returned them positive payoffs. So the positive links were kept and the negative ones were revoked. In some cases in our testing samples the total average payoff did not reach a positive level although the emergence of similarities and the behaviour showed otherwise. In this case you could see a lot of equalities and cooperation which should have been a signal for the reception of many positive payoffs. However, sometimes the expected payoff for each of the agents was that negative that they were not able to reach a positive outcome in the end. In most cases in our tests the resulting networks and behaviour really fit to the

output. Many negative total averages signified higher randomness in the graphs and more defections in behaviour.

It was interesting that an artificial society also came to the same conclusion as real societies did. When memory was included, the agents tried to reach a high level of prosperity for the whole society. So there were many individuals which got a medium reward instead of a small number which received a high reward. From the point of logic you would say that every agent in the artificial world just tried to get out the best for themselves and did not care for the others. The best reward was reached via defection if the other interaction partner cooperated. However, if both partners defected, the output was lower than the output for a mutual cooperation. So the emergence of behaviour was very likely to display mostly cooperative behaviour.

In the following tables (from table 4.1 to table 4.7) you can find an overview of all results of the simulation.

Pure Buyer = 3, Pure Seller = 3, Buyer-Seller = 3	
Finite State Machine Memory = 1	
Similarity	random networks
Payoff	negative average payoffs
Behaviour	mostly defection
Finite State Machine Memory = 10	
Similarity	alike networks in the generations where the program runs through all trade cycles
Payoff	total average always negative, sometimes positive average for pure sellers and buyer-sellers
Behaviour	mix of defection and cooperation, pure buyers show more cooperation, buyer-sellers cooperate sometimes, pure sellers have a balanced mix
Finite State Machine Memory = 16	
Similarity	equal networks in the last trade cycles of every generation
Payoff	total average in the first generation negative, in all other generations the average is positive
Behaviour	mostly cooperation

Table 4.1: Results for the testing with 3 pure buyers, 3 pure sellers and 3 buyer-sellers

Pure Buyer = 6 , Pure Seller = 3 , Buyer-Seller = 3	
Finite State Machine Memory = 1	
Similarity	random networks but more similarities than in the case with 3 pure buyers, 3 pure sellers and 3 buyer-sellers
Payoff	higher negative payoff than in the case with 3 pure buyers, 3 pure sellers and 3 buyer-sellers
Behaviour	defection with hardly any cooperation
Finite State Machine Memory = 10	
Similarity	alike networks in the generations where the program runs through all trade cycles, but more than in the case with 3 pure buyers, 3 pure sellers and 3 buyer-sellers
Payoff	the average for pure buyers and pure sellers is sometimes positive, the total average is always negative
Behaviour	mix of cooperation and defection, at first there is more defection and in the last two generations there is more cooperation
Finite State Machine Memory = 16	
Similarity	fast convergence to equal networks in each of the generations
Payoff	in the first 4 generations the total average is negative, then it switches to a positive or neutral level
Behaviour	mostly cooperation, from generation 6 onwards there is total cooperation

Table 4.2: Results for the testing with 6 pure buyers, 3 pure sellers and 3 buyer-sellers

Pure Buyer = 3 , Pure Seller = 6 , Buyer-Seller = 3	
Finite State Machine Memory = 1	
Similarity	random networks, the most similarities occur between buyer-sellers
Payoff	the total average is negative
Behaviour	mostly defection
Finite State Machine Memory = 10	
Similarity	some equalities in the networks but less than in the case with 6 pure buyers, 3 pure sellers and 3 buyer-sellers
Payoff	sometimes the average payoff for pure sellers is positive, the total average is always negative
Behaviour	there is mostly defection with some cooperation
Finite State Machine Memory = 16	
Similarity	there are many equal networks in the last 2 trade cycles of every generation
Payoff	the total average of the first generation is negative, then it switches to positive
Behaviour	mix of cooperation and defection with a higher level of cooperation, in the last generation there is total cooperation

Table 4.3: Results for the testing with 3 pure buyers, 6 pure sellers and 3 buyer-sellers

Pure Buyer = 3 , Pure Seller = 3 , Buyer-Seller = 6	
Finite State Machine Memory = 1	
Similarity	random networks with a few single similarities
Payoff	the total average is always negative
Behaviour	mostly defection
Finite State Machine Memory = 10	
Similarity	some equalities in the network connections
Payoff	although the average for pure sellers is positive, the total average is always negative
Behaviour	mostly defection, but less than with a memory of 1 bit
Finite State Machine Memory = 16	
Similarity	many similarities
Payoff	the total average for generations 1 and 3 is negative, for all the other generations it is positive
Behaviour	mostly cooperation, after generation 5 total cooperation

Table 4.4: Results for the testing with 3 pure buyers, 3 pure sellers and 6 buyer-sellers

Pure Buyer = 3 , Pure Seller = 6 , Buyer-Seller = 6	
Finite State Machine Memory = 1	
Similarity	still no real pattern, but many more similarities than in the previous cases with a memory of 1 bit
Payoff	although the average for pure sellers is positive, the total average is always negative
Behaviour	mix of defection and cooperation
Finite State Machine Memory = 10	
Similarity	networks are very random, there seem to be less similarities than in the case with a memory of 1 bit
Payoff	pure sellers sometimes show a positive average, but the total is negative, there seems to be more negativity than with a memory of 1 bit
Behaviour	mix of defection and cooperation with a little bit more defection, cooperation comes mainly from buyer-sellers
Finite State Machine Memory = 16	
Similarity	there are many equal links but they are not too obvious
Payoff	the total average is negative, except for generation 6 (positive)
Behaviour	mix of cooperation and defection, cooperation rises, buyer-sellers and pure sellers tend to cooperate in the last few generations

Table 4.5: Results for the testing with 3 pure buyers, 6 pure sellers and 6 buyer-sellers

Pure Buyer = 6 , Pure Seller = 3 , Buyer-Seller = 6	
Finite State Machine Memory = 1	
Similarity	random networks with just a few similarities
Payoff	the total average is negative all the time
Behaviour	mostly defection
Finite State Machine Memory = 10	
Similarity	many similarities on first sight in some generations
Payoff	in the first generation all averages are negative, after that the pure sellers' average is mostly positive but the total average remains negative
Behaviour	at first there is mostly defection, then cooperation increases until reaching a balanced mix of defection and cooperation
Finite State Machine Memory = 16	
Similarity	the occurrence of equal links alternates in every generation between many and not so many connections
Payoff	the total average is always negative
Behaviour	mix of cooperation and defection, at last there is a higher level of defection

Table 4.6: Results for the testing with 6 pure buyers, 3 pure sellers and 6 buyer-sellers

Pure Buyer = 6 , Pure Seller = 6 , Buyer-Seller = 3	
Finite State Machine Memory = 1	
Similarity	random networks with just a few similarities
Payoff	pure sellers tend to receive a positive average, the total is always negative
Behaviour	mostly defection
Finite State Machine Memory = 10	
Similarity	equalities in the networks in every generation after generation 2
Payoff	there are some generations with a positive total average, the other generations have a negative one
Behaviour	in the case where the total average is positive defection nearly disappears, when it is negative there is a balanced mix of cooperation and defection
Finite State Machine Memory = 16	
Similarity	the occurrence of equal links alternates in every generation between really many and many connections
Payoff	the total average is negative
Behaviour	mix of cooperation and defection, at last there is a higher level of cooperation

Table 4.7: Results for the testing with 6 pure buyers, 6 pure sellers and 3 buyer-sellers

A Inhaltsangabe

Das Ziel dieser Arbeit ist es, die Ergebnisse einer agentenbasierten ökonomischen Computersimulation unter verschiedenen strukturellen Bedingungen zu vergleichen. Obwohl es zwischen Agenten (Agenten-Programmen) aus der Informatik und den hier tatsächlich simulierten Individuen Verbindungen beziehungsweise Änlichkeiten gibt, haben diese 2 Konzepte unterschiedliche Bedeutungen – im Fall hier sind Agenten die Verkörperung von Denken (in sehr unterschiedlichen Ausmaßen) und Lernen (dieses wird stark unterschiedlich modelliert). Die Komplexität des Denkens dieser Agenten und die Lernmuster, die sie innehaben, sind laufend Themen der Forschung. Aber hier in dieser Arbeit richtet sich der Fokus mehr auf die Modellierung des Musters, das durch die Wechselbeziehungen der Agenten entsteht. So will man herausfinden wie eine Gesellschaft von innen heraus funktioniert – die Basis dafür sind denkende Agenten. Aktuelle Wissenschaftsliteratur dieses Themas diskutiert verschiedene Ansätze für unterschiedliche ökonomische und soziale Problemstellungen. Die Ansätze sind agentenbasiert und folgen einem "bottom-up"- Konzept. Die stark abweichenden Ansichten in dieser Forschung entstehen dadurch, dass viele unterschiedliche Fachrichtungen (Soziologie, Psychologie, Physik, Informatik, Recht, Politikwissenschaft, etc.) daran arbeiten, die alle andere Forschungsziele verfolgen. Diese Ansätze sind deshalb nur mehr mit computergestützten Simulationen

auszuführen, weil Entscheidungen und Wechselbeziehungen einer sehr großen Anzahl von Agenten simuliert werden. Das erfordert sehr viele Berechnungen und es werden algorithmische Methoden verwendet, die für algebraische Modelle nicht mehr möglich sind. Eine Frage der Forschung beschäftigt sich mit dem Typ und der Form der Wechselbeziehungen, mit der Lernstruktur und dem Raum, wo die Agenten 'leben'. Angeblich führen unterschiedliche Voraussetzungen in diesen Bereichen zu unterschiedlichen Ergebnissen. In der Literatur gibt es Modelle bei denen die Agenten im Raum leben – nämlich auf zwei-dimensionalen Flächenrastern (gleichmäßig oder ungleichmäßig)– wo sie mit ihren Nachbarn interagieren. Weiters gibt es Modelle, die nicht räumlich sind. Dort kann jeder mit jedem in Wechselbeziehung treten. Neuere Modelle betrachten die Beziehungen zwischen den Agenten aus einer neuen Perspektive. Diese ist soziologisch und stellt die Behauptung auf, dass wichtige Eigenschaften einer Gesellschaft von ihrer Form abzuleiten sind. Diese Arbeit soll die Ergebnisse einer ökonomischen bzw. soziologischen Simulation vergleichen. Es werden verschiedene Eingangsparameter bestimmt und untersucht, wie sich diese auf die entstehende Struktur und das Verhalten der Agenten auswirken.

Bibliography

[1] W. Brian Arthur. *Out-of-Equilibrium Economics and Agent-Based Modelling*, volume Handbook of Computational Economics 2, chapter 32, pages 1551–1564. North-Holland, 2006.

[2] Robert Axelrod. Advancing the art of simulation in the social sciences. *Japanese Journal for Management Information Systems, Special Issue on Agent Based Modelling*, 12(3):–, August 2003.

[3] Robert Axelrod. *Agent-Based Modelling as a Bridge Between Disciplines*, volume Handbook of Computational Economics 2, chapter 33, pages 1565–1584. North-Holland, 2006.

[4] Thomas Brenner. *Agent Learning Representation: Advice on Modelling Economic Learning*, volume Handbook of Computational Economics 2, chapter 18, pages 895–947. North-Holland, 2006.

[5] L. Shapley D. Gale. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[6] Leigh Tesfatsion David McFadzean. A c++ platform for the evolution of trade networks. Technical Report 39, Iowa State University, June 1999.

[7] Catherine Dibble. *Computational Laboratories for Spatial Agent-Based Models*, volume Handbook of Computational Economics 2, chapter 31, pages 1511–1548. North-Holland, 2006.

[8] John Duffy. *Agent-Based Models and Human Subject Experiments*, volume Handbook of Computational Economics 2, chapter 19, pages 949–1011. North-Holland, 2006.

[9] John Foster. From simplistic to complex systems in economics. *Discussion Paper*, 335:–, October 2004. School of Economics, The University of Queensland.

[10] Gerald J. Lieberman Frederick S. Hillier. *Introduction to Operations Research*, chapter 14 - Game Theory, pages 659–679. McGraw-Hill, 8th edition, 2005.

[11] Mark Granovetter. A theoretical agenda for economic sociology. *to appear in Economic Sociology at the Millennium*, -:–, June 2000.

[12] Mark Granovetter. The impact of social structure on economic outcomes. *Journal of Economic Perspectives*, 19(1):33–50, 2005.

[13] Peter Howitt. *Coordination Issues in Long-Run Growth*, volume Handbook of Computational Economics 2, chapter 35, pages 1605–1624. North-Holland, 2006.

[14] Kenneth L. Judd. *Computationally Intensive Analyses in Economics*, volume Handbook of Computational Economics 2, chapter 17, pages 881–893. North-Holland, 2006.

[15] Scott E. Page Ken Kollman. *Computational Methods and Models of Politics*, volume Handbook of Computational Economics 2, chapter 29, page 1433. North-Holland, 2006.

[16] Elinor Ostrom Marco A. Janssen. *Governing Social-Ecological Systems*, volume Handbook of Computational Economics 2, chapter 30, pages 1465–1509. North-Holland, 2006.

[17] Robert Marks. *Market Design Using Agent-Based Models*, volume Handbook of Computational Economics 2, chapter 27, pages 1339–1380. North-Holland, 2006.

[18] David McFadzean. Simbiosys: A class framework for biological simulations. Master's thesis, Department of Computer Science, Calgary, Alberta, September 1994.

[19] Don Ross. Review of game theory. *http* : *//plato.stanford.edu/entries/game − theory*, January 2009. in the Stanford Encyclopedia of Philosophy.

[20] Leigh Tesfatsion. *Agent-Based Computational Economics: A Constructive Approach to Economic Theory*, volume Handbook of Computational Economics 2, chapter 16, pages 830–880. North-Holland, 2006.

[21] Bart Nooteboom Tomas B. Klos. Agent-based computational transaction cost economics. *Journal of Economic Dynamics and Control*, 25:503–526, 2001.

[22] Nicolaas J. Vriend. *ACE Models of Endogeneous Interactions*, volume Hand-

book of Computational Economics 2, chapter 21, pages 1047–1079. North-Holland, 2006.

[23] Duncan J. Watts. *Six Degrees - The Science of a Connected Age*. Vintage, 2004.

[24] Wikipedia. Triangular number. *http* : $//en.wikipedia.org/wiki/Triangular_number$, June 2007.

[25] Wikipedia. Game theory. *http* : $//en.wikipedia.org/wiki/Game_theory$, January 2009.

[26] Allen Wilhite. *Economic Activity on Fixed Networks*, volume Handbook of Computational Economics 2, chapter 20, pages 1013–1045. North-Holland, 2006.

List of Figures

1.1 Recommendations for choosing a learning model [4] 21

2.1 Random graph connectivity [23] 27
2.2 The complete network [26] 29
2.3 The star network [26] 30
2.4 The ring network [26] 30
2.5 The grid network [26] 31
2.6 The tree network [26] 31
2.7 The small-world network [26] 32
2.8 The power network [26] 32
2.9 Interaction rules for different values of *alpha* [23] 34
2.10 Path length and clustering coefficient for different values of *alpha* [23] 35
2.11 Path length and clustering coefficient for different values of *beta* [23] 36
2.12 Normal distribution of probability of k neighbours [23] 38
2.13 Power law distribution of probability of k neighbours [23] 39
2.14 Affiliation network [23] 42
2.15 Extensive Form [25] 49
2.16 Normal Form [25] 50
2.17 Normal Form of the Prisoner's Dilemma [19] 51

3.1 Class diagram of the SimBioSys framework including classes of the Trade Network Game [6] [18] 58
3.2 Pseudocode for the SimBioSys framework [6] 59
3.3 Pseudocode for the Trade Network Game [6] 70

List of Tables

3.1 Description of Parameter Specification 1 [6] 63
3.2 Description of Parameter Specification 2 [6] 64
3.3 Description of Parameter Specification 3 [6] 65

4.1 Results for the testing with 3 pure buyers, 3 pure sellers and 3 buyer-sellers 95
4.2 Results for the testing with 6 pure buyers, 3 pure sellers and 3 buyer-sellers 96
4.3 Results for the testing with 3 pure buyers, 6 pure sellers and 3 buyer-sellers 97
4.4 Results for the testing with 3 pure buyers, 3 pure sellers and 6 buyer-sellers 98
4.5 Results for the testing with 3 pure buyers, 6 pure sellers and 6 buyer-sellers 99
4.6 Results for the testing with 6 pure buyers, 3 pure sellers and 6 buyer-sellers 100
4.7 Results for the testing with 6 pure buyers, 6 pure sellers and 3 buyer-sellers 101

Lebenslauf

Persönliche Daten

Katharina Thelesklaf
Eichelauweg 555
8911 Admont

geb. am 28. 05. 1982

Schulausbildung

1992 -2000
Stiftsgymnasium Admont (humanistischer Zweig) mit Abschluss der Matura

Hochschulstudium

2000 -2001
Studium der Technischen Physik an der TU Wien

2001 -2005
Studium der Wirtschaftsinformatik (TU Wien, UNI Wien) mit Abschluss des Bakkalaureats

2005 -2009
Studium der Wirtschaftsinformatik (TU Wien, UNI Wien) mit Abschluss des Magisters

Feb. 07 -Juli 07
Auslandssemester in Toulouse (Frankreich)