Faculty of
Computer Science, Electrical Engineering and Mathematics

Dissertation
in Computer Science

Multi-Criteria Cooperation in Multiagent Systems by Local Adaptation

Markus Eberling

submitted to the
Faculty of Computer Science,
Electrical Engineering and Mathematics

in partial fulfillment of the requirements for the degree of
doctor rerum naturalium
(Dr. rer. nat.)

Paderborn, May 2011

Abstract

Cooperation in multiagent systems is still a challenge. Agents are designed to behave rationally which often means selfish, as different agents may have different preferences over environment states and/or actions. In particular, this is the case if different parties with contrary goals design the agents of a multiagent system. In such systems, the agents cannot rely on receiving cooperation, if cooperation is needed to archive a specific goal. Additionally, malfunctioning agents are a problem, as they may try to exploit cooperative agents to increase their own performance. Especially if cooperation is not for free, agents may tend to behave uncooperatively. This problem often occurs in peer-to-peer file-sharing systems. A file-sharer will not receive any benefit from sharing a file but has to pay some costs for sharing in the form of used bandwidth. However, in human groups cooperation is omnipresent, although it may produce costs, as this may lead to beneficial situations for the whole group. Social scientists have investigated that cooperative behavior likely occurs between similar persons with respect to ideology, look or shared opinions.

In this thesis, a new local adaptation-based mechanism is presented that favors emergence of cooperation. We will consider a multiagent system where agents have to fulfill jobs constructed of smaller tasks, which require specific skills. As each agent is only equipped with a subset of all possible skills, cooperation is often needed to fulfill the jobs that are allocated to the agents. Agents will compare themselves based on rating-vectors. If the similarity is sufficient from the individual agent's view, an agent will provide its help to a requesting agent. Agents are evaluated with the help of the job/task allocation scenario. If the agent does not belong to a local set of best performing agents, i.e. compared to its neighbors in a social network graph, the agent is unsatisfied and adapts itself to a set of ideal agents from its neighborhood. The adaptation is a movement of its current rating-vector into the direction of the rating-vectors of some role model agents. We show that this novel local adaptation-based learning algorithm produces high rates of cooperation. We formally analyze the proposed approach to show basic properties of the algorithm and present a rigorous experimental analysis where we examine the influence of different system parameters. Furthermore, we consider different network structures and the influence of capacity constraints. Additionally, we will consider so-called rewiring strategies, that are used by the agents to change their neighborhood by exchanging old with new links.

Zusammenfassung

Kooperation in Multiagentensystemen ist noch immer eine Herausforderung. Agenten werden erstellt um rational zu handeln, was oft Eigennützigkeit bedeutet, da Agenten unterschiedliche Präferenzen über Umgebungszustände und/oder Aktionen haben können. Dies kann insbesondere der Fall sein, wenn Agenten von unterschiedlichen Herstellern mit unterschiedlichen Zielen erstellt wurden. In einem solchen Fall kann ein Agent sich nicht mehr darauf verlassen, Kooperation von anderen Agenten zu bekommen, wenn diese nötig wird. Ein anderes Problem sind fehlerhaft agierende Agenten, die versuchen, das kooperative Verhalten anderer auszunutzen um den eigenen Profit zu steigern. Insbesondere, wenn Kooperation Kosten für den Helfer verursacht, kann es sein, dass die Agenten sich unkooperativ verhaltenden. Dieses Problem tritt häufig in Tauschbörsen auf. Ein Anbieter von Dateien bekommt keinen Gewinn dadurch, dass er Dateien anbieten, muss allerdings einen Preis in Form von weniger zur Verfügung stehender Bandbreite dafür bezahlen. Dennoch gibt es in der menschlichen Gesellschaft allgegenwärtige Kooperation, da dies zu besseren Lebensumständen für alle führen kann und zu einem Gewinn für die gesamte Gesellschaft. Sozialwissenschaftler haben herausgefunden, dass kooperatives Verhalten besonders zwischen sich ähnelnden Personen auftritt bezüglich ihrer Ideologie, ihrem äußeren Erscheinungsbild oder gleichen Meinungen.

In dieser Arbeit werden wir ein Multiagentensystem vorstellen, in dem die Agenten Aufgaben zu erledigen haben, die aus kleineren Unteraufgaben bestehen, die wiederum bestimmte Fähigkeiten verlangen. Da jeder Agent nur mit einer Teilmenge der möglichen Fähigkeiten ausgestattet ist, sind die Agenten oft auf Kooperation angewiesen um die ihnen zugewiesenen Aufgaben zu erledigen. Die betrachteten Agenten vergleichen sich untereinander auf der Basis von abstrakten Bewertungsvektoren. Bei hinreichender, lokaler Ähnlichkeit wird ein Agent mit einem um Hilfe fragenden Agenten kooperieren. Mit Hilfe des Aufgabenverteilungsszenarios werden die Agenten evaluiert. Wenn ein Agent nicht zu der lokal besten Gruppe an Agenten gehört — das bedeutet, die beste Gruppe unter den Nachbarn in einem Agentennetzwerk — dann ist der Agent unzufrieden und passt sich an eine Menge an Vorbildern aus seiner Nachbarschaft an. Diese Adaptation wird durch eine Verschiebung des aktuellen Bewertungsvektors in Richtung der Bewertungsvektoren der Vorbilder realisiert. Wir werden zeigen, dass dieses neue, lokale, adaptionsbasierte Lernverfahren zu hohen Kooperationsraten führt. Wir werden das vorgestellte Verfahren formal analysieren, um Eigenschaften des Systems zu zeigen und zudem in einer großen experimentellen Analyse den Einfluss unterschiedlicher Systemparameter untersuchen. Des Weiteren werden wir unterschiedliche Netzwerkstrukturen und den Einfluss von Kapazitätsbeschränkungen betrachten sowie Verfahren für die Netzwerkanpassung durch die Agenten vorstellen.

Acknowledgements

As the final step of writing this thesis, I now have the opportunity to express my gratitude to all the people who supported me in many ways.

First, I would like to thank my supervisor Prof. Dr. Hans Kleine Büning for fruitful discussions about my thesis and his support. He let me work on my own pace, showing me the right direction whenever it was needed. Secondly, I am grateful to Prof. Dr. Franz Josef Rammig who agreed to review this thesis. Additionally, I am very grateful to Prof. Dr. Friedhelm Meyer auf der Heide, Prof. Dr. Christian Scheideler and Dr. Matthias Fischer for their interest in my work.

Furthermore, I would particular like to thank my former and current colleagues Dr. Natalia Akchurina, Isabela Anciutti, Heinrich Balzer, Dr. Uwe Bubeck, Dr. Steffen Priesterjahn, Asmir Vodencarevic and Yuhan Yan for the nice atmosphere in the research group *Knowledge-based Systems* at the University of Paderborn. My special thank goes to Dr. Theodor Lettmann, Michael Baumann, and Thomas Kemmerich for their comments and helpful suggestions about my work. Furthermore, I would like to thank Simone Auinger, Gerd Brakhane, Christa Stoll and Patrizia Höfer for supporting me in organizational and technical issues. My supervision of master and bachelor students taught me a lot and I am very grateful for the discussions we had.

I am very proud of all my friends from different areas like sport clubs or other organizations for giving me the time and space for doing something else besides my research. It was a pleasure to have discussions about issues that are much more important in life then receiving a PhD degree. I do not want to thank all of you namely, but you all helped me very much in regenerating after long working days and to get the strength of going further and further.

Last but not least, my thank goes to my parents, Ulrich and Ursula Eberling, who supported me all the years throughout my studies. Without you, this work would have never been possible.

Markus Eberling
Paderborn, May 2011

Contents

1 Introduction

The study of multiagent systems is a field in computer science that emerged in about 1980 (Wooldridge, 2009). Multiagent systems are often used to model so-called *artificial social systems* or systems composed of multiple entities like the Internet (Wooldridge, 2009). One key aspect of multiagent systems is the cooperation between these autonomous entities.

Cooperation can be found in everyday life for instance in groups of humans or in companies that are organized in a network structure. In most scenarios, cooperation leads to higher benefit for the whole group and to higher benefit for the individuals. Mostly, the group members have a common goal but different motivations to join the group (Pennington, 2002) or to stay in it (Buchanan and Huczynski, 1997). Companies build networks like supply chains to achieve their goals (Peitz, 2002). The resulting supply chains are helpful for the companies to produce qualitative products (Schmidt, 1997). Reciprocal behavior is one of the characteristics of such networks (Sydow, 1992). Another aspect is altruism, which is on the one hand helping others without being paid for (Berkowitz and Macaulay, 1970) but which can produce costs on the other hand (Krebs, 1982; Wispe, 1978). The decision to cooperate is often based on different criteria like kin selection or social cues.

We model such systems of collaborative groups with a multiagent system. In the considered scenario, agents should complete some jobs that consist of a number of tasks. Each task needs a specific skill and as the agents are not equipped with all possible skills, they mostly need cooperation partners to complete their jobs. However, completing a job leads to a reward solely for the agent to which the job was assigned. The reward is not transferable and the agents are not allowed to pay others to help them. Moreover, the cooperative behavior produces costs reflecting the use of CPU time or simply speaking some arbitrary resources of the cooperative agent. Such situations often occur in peer-to-peer file sharing systems where a file sharer has no profit for sharing its files and even has to pay some costs in form of used bandwidth (Adar and Huberman, 2000).

We model the process of determining cooperation partners with the help of propositions. The agents rate these propositions and based on the distances of their ratings they determine the agents they are willing to cooperate with. Each proposition leads to a criterion that has to be fulfilled. If all criteria are fulfilled, the agent cooperates with an agent asking for help. The intention behind this step is that humans more likely cooperate with others if the other person is similar to them in form of cultural background, social standing or other subjective criteria (Hinde and Groebel, 1991). The ratings of the propositions may also influence the agent's behavior: a proposition with the meaning "the road is clear" can make a driving agent drive faster if it gives

a high rating to this proposition. A proposition can be simple atomic information as the term is known from logics or it can be an idea, belief, or some preference. We only concentrate on the ratings of these propositions and the propagation through a network of agents.

1.1 Research Contributions

We consider cooperation in multiagent systems based on real-life cooperation mechanisms. We present a system where cooperation is not encoded in the entities, as this may be exploited by agents that do not behave in the desired cooperative way. Thus, we want to construct a multiagent system with emergent cooperation where the decisions to cooperate are based on agent similarities and where the agents are not able to exploit cooperation of other agents. The investigated approach has the following properties:

Local view The agents will not be able to gain a global view on the system in order to promote scalability.

Similarity-based cooperation decisions As in real-life, the decision to cooperate is based on the similarity of agents, which is a phenomenon that is known from social science (Hinde and Groebel, 1991). Therefore, there will be a mechanism that first measures the similarity between agents and second helps to support the decision to cooperate or not.

Minimal knowledge The properties of the agents will mostly be unobservable to other agents. This helps to construct algorithms that do not require much knowledge. With minimal knowledge, we aim to get simpler approaches and simpler entities of the agent system.

Adaptability Agents are able to adapt to others in order to improve their performance and profit.

Simplicity The mechanisms need as less information as possible to receive their design goal, i.e. an increase of cooperative behavior.

Directed cooperation We consider single-sided or directed cooperation decisions. For cooperation decisions that take into account both ways of cooperation other powerful mechanism exist, e.g. negotiations (Wooldridge, 2009).

Dynamic network As inspired by real-life, we allow the agents to rewire their connections in order to get higher profits. Agents that seem to be useless can be exchanged by other agents. This should also model the aspect of new acquaintances.

Emergent cooperation The system will be able to show emergent cooperation in contrast to hard-coded cooperative behavior. This benefits the reliability of cooperation as in the case of hard-coded cooperation, agents have on the one hand no other possibility as to cooperate—thus, they lose their autonomy—and on the other hand it is not possible to

cheat for the agents as emergent cooperation is unpredictable.

1.2 Overview of the Thesis

The thesis on hand is structured as follows:

Chapter 2 defines terms used in this thesis and gives an overview on existing work in the field of cooperation in agent systems and agent networks.

Chapter 3 provides the formal definition of the proposed system that has all the desired system properties mentioned before. Besides this, the considered benchmark-scenario will also be described.

Chapter 4 formally analyzes the probability of having all relevant skills in the agent's vicinity. Additionally, we will analyze the propagation of the introduced value-vectors and show under which conditions convergence to mutually cooperative agents can be guaranteed. Furthermore, we investigate the computational complexity of our approach.

Chapter 5 provides an exhaustive experimental analysis of the proposed system and identifies best values for the most important parameters of the system.

Chapter 6 considers the influence of different network structures under static and dynamic conditions.

Chapter 7 introduces capacity constraints and shows that the system can successively handle the requirements of the benchmark scenario.

Chapter 8 considers different strategies for rewiring of connections in the agent network, which we call *social networking* strategies.

Chapter 9 concludes the thesis and provides an outlook to future work.

The structure of the thesis is illustrated in Figure 1.1. There are several possible reading paths through the thesis. The main path goes along Chapter 1, Chapter 3, Chapter 4, Chapter 5 and finally Chapter 9. Chapter 2 may support the understanding for the formal model and scenario description in Chapter 3. Chapters 6–8 consider additional extensions of the proposed model and evaluate these extensions in an experimental analysis. These three chapters can be read additionally to the main path of the thesis.

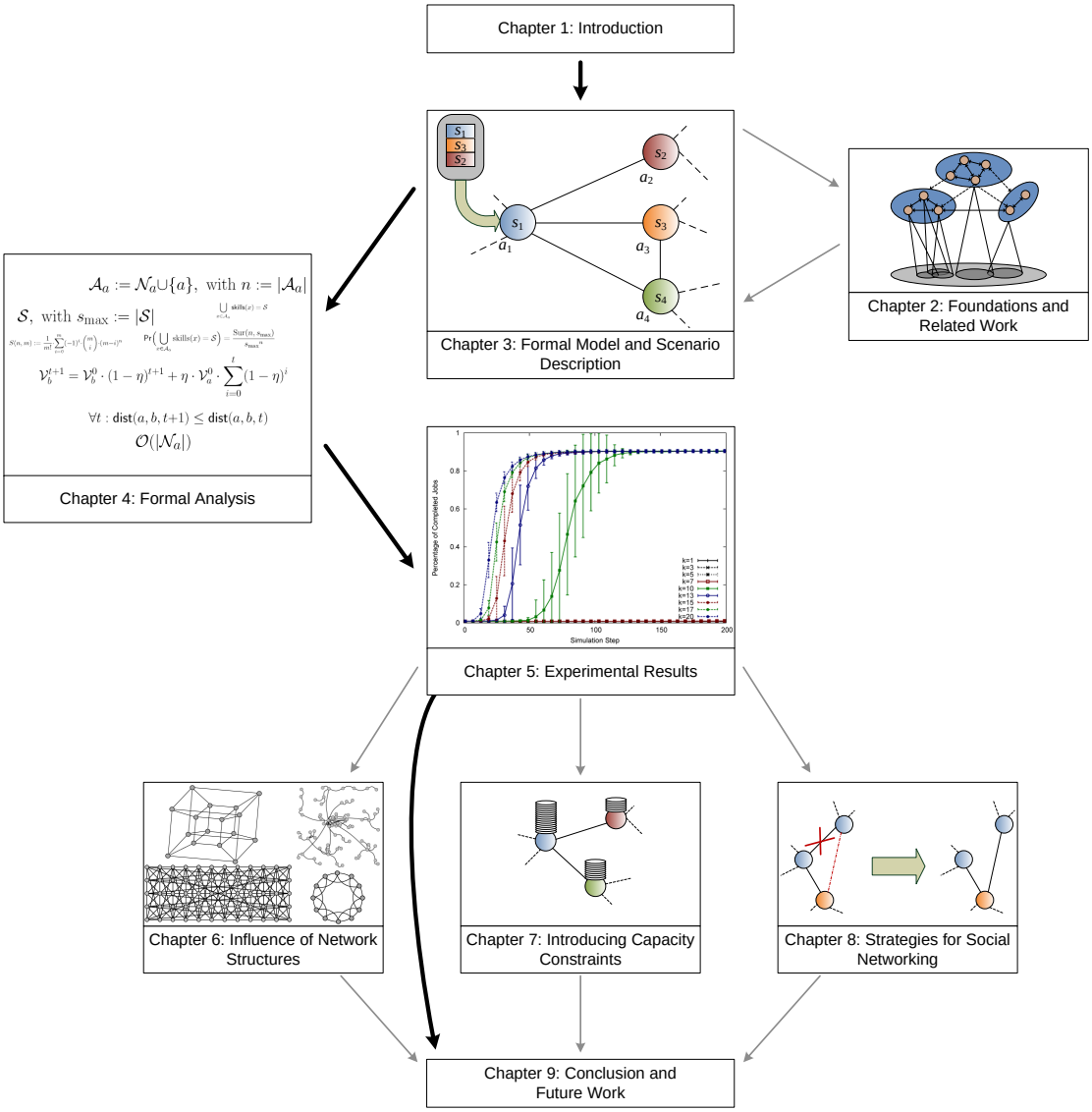


Figure 1.1: Possible reading paths through the thesis.

2 Foundations and Related Work

In this chapter, we present the foundations of this thesis. We introduce the reader to agents and multiagent systems in the first section and then name and define specific terminology in the second section. The last three sections will present related work from three research perspectives, namely *artificial societies*, the *task allocation problem*, and *imitation-based learning and memetics*.

2.1 Agents and Multiagent Systems

Multiagent systems are a discipline in computer science which is classified in the ACM Computing Classification System (Association for Computing Machinery, 1998) to be part of the area I.2.11 *Distributed Artificial Intelligence*. In the area I.2.11 also the areas *coherence and coordination*, *intelligent agents* and *languages and structures* are classified on the same level as *multiagent systems*. Figure 2.1 shows the part where this thesis would be classified according to the ACM Classification System.

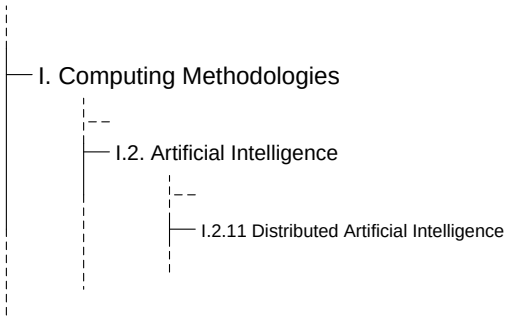


Figure 2.1: Part of interest of the classification system of the ACM (Association for Computing Machinery, 1998).

In literature, many definitions and descriptions on agents and multiagent systems coexist. In the following, we will briefly discuss common properties of agents and after that we will present the definition of multiagent system.

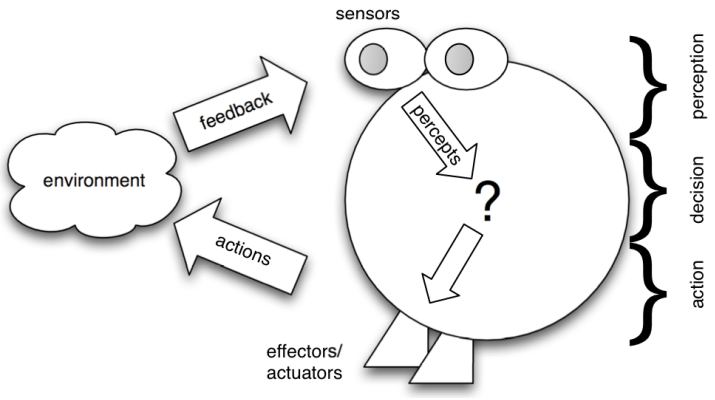


Figure 2.2: The relation between an agents and its environment (Wooldridge, 2009).

2.1.1 Intelligent Agents

One of the most frequently cited definitions on intelligent agents is the following:

Definition 2.1 (Agent (Wooldridge and Jennings, 1995)): *An* agent *is a computer system that is* situated *in some* environment*, and that is capable of* autonomous action *in this environment in order to meet its delegated objectives.*

Although this definition is very vague, it covers the most important facts on agents and agent systems. "An agent [...] is situated in some environment" captures the fact that an agent cannot exist without some kind of environment. This is often called the duality of agents and the environment (Ferber, 1999). Taking this definition as it is, one can identify several agents we are used to. There are web-crawler agents of search engines, which search the Internet and index the websites (Sherman, 2005; Vise and Malseed, 2008), robots cleaning rooms, or humans which can also be view as an agent—though it is hard to decide what the delegated objectives are.

Wooldridge and Jennings have suggested three main capabilities that are needed to call an agent *intelligent* (Wooldridge and Jennings, 1995):

Reactivity Intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.

Proactivity Intelligent agents are able to exhibit goal-directed behavior by taking the initiative in order to satisfy their design objectives.

Social ability Intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

The relation between an agent and its environment is visualized in Figure 2.2. An agent senses the environment state through its sensors. Then, the agent selects an action to be executed in

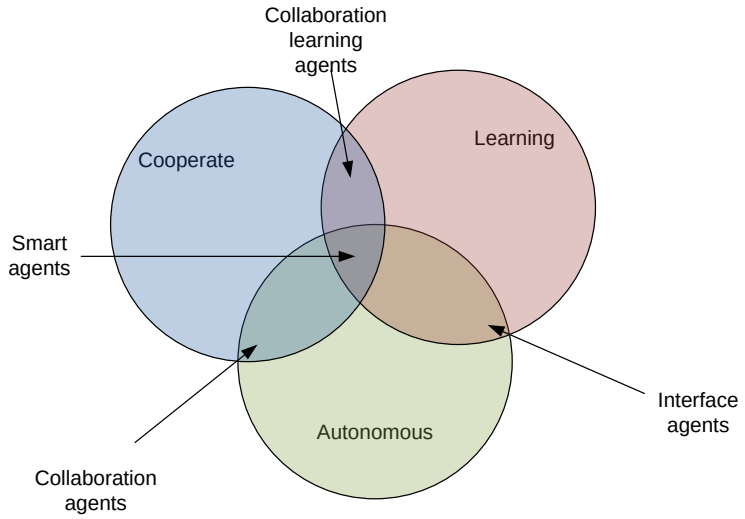


Figure 2.3: Classification of agents proposed by Nwana (Nwana, 1996).

the environment. The action's execution is done through its actuators. The result is another environment state which the agent again receives through its sensors.

Russell and Norvig have specified properties for classification of the agent's environment (Russell and Norvig, 2010):

Fully observable vs. partially observable In a *fully observable* environment the agent is able to obtain correct and up-to-date information about the environment. Especially, the agent has access to the complete state of the environment at each point of time. In *partially observable* environments this is not the case.

Single agent vs. multiagent An environment can be characterized by the number of agents that are situated in the environment. If there is only one agent we call the environment a *single agent* environment and a *multiagent* environment if two or more agents are considered.

Deterministic vs. stochastic Consider an agent that faces the same situation again and selects both times the same action. If the result is identical in both situations we call the environment *deterministic*. Otherwise the environment is said to be *stochastic*.

Episodic vs. sequential If the agent's experience can be divided into atomic episodes, we call an environment *episodic*. One key aspect is then that the next episode does not depend on the actions taken in previous episodes. If short-term actions have long-term consequences then we call the environment *sequential*. Like in the episodic case, the agent does not need to think ahead. Sequential environments are much more complex than episodic ones.

Static vs. dynamic In a *static* environment the only changes of the environment state occur as results of the agent's actions. In *dynamic* environments there can be other processes that may effect the environmental state. If other agents are situated in the same environment, it may seem to be dynamic to one specific agent because environmental state changes occur as a result of the other agents' actions.

Discrete vs. continuous We call an environment *discrete* if there is a finite number of actions and perceptions in it. Otherwise we call it *continuous*. Most real-world examples of agent systems are continuous.

Known vs. unknown In a *known* environment the outcomes of all actions are given. Strictly speaking, this distinction does not refer to the environment itself but to the agent's (or designer's) state of knowledge. In an unknown environment the outcomes cannot be predicted.

Another classification is proposed in (Nwana, 1996). There, the agents are classified based on the three properties *learning*, *cooperate* and *autonomous*. Figure 2.3 visualizes this classification. The most interesting agents are those who combine all three properties. Nwana calls these agents *smart agents*.

As we have briefly presented the properties of a single agent and an agent's environment we will continue with the description of multiagent systems in the next section.

2.1.2 Multiagent Systems

Whenever an agent system consists of more than one agent, it is called a multiagent system (MAS). For so-called single agent systems, the most important research questions deal with the perception of and acting in the environment and all aspects of cognition like thinking, planning, creativity, orientation, imagination and learning abilities etc. of the agent (Thagard, 2010). Jennings presented a canonical view on multiagent systems as it is commonly accepted in the MAS community (Jennings, 2000). An illustration of this view is given in Figure 2.4: Agents build groups and the influence and visibility areas of the agents are shown. The dotted arrows show the dependencies between the groups of agents.

The most important properties of agent systems can also be seen in the figure. Although hierarchies of agents may exist, there is no central instance in a multiagent system. Thus, the distributed character is one of the main characteristics. Another important property is the locality of the view of agents and / or their area of influence area.

Many research on MAS focuses on the following parts:

- Communication
- Cooperation

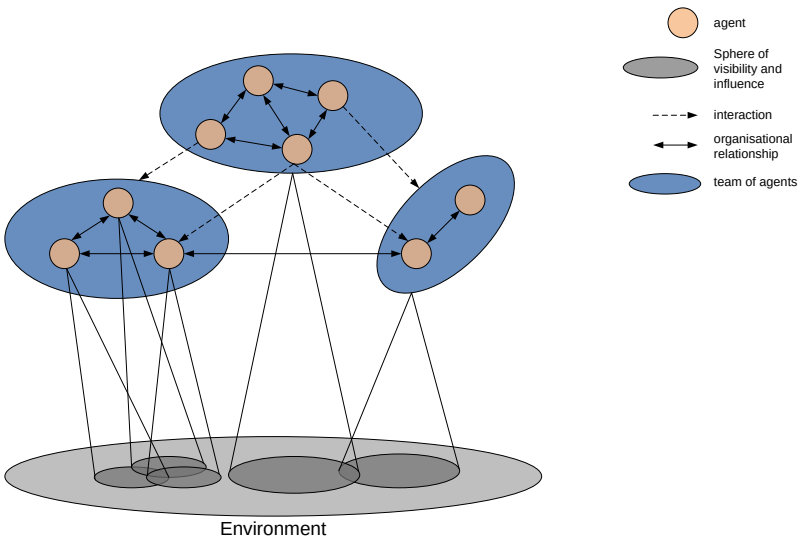


Figure 2.4: Canonical view on a multiagent system (Jennings, 2000).

- Coordination
- Organization
- Interaction

When communication is considered, the communication between the agents is meant. There exist several approaches for agent languages and communication protocols like the Knowledge Query and Manipulation Language (KQML) presented in (Finin et al., 1994a,b), the Knowledge Interchange Format (KIF) described in (Genereseth and Fikes, 1992) or the FIPA agent communication language (FIPA-ACL) developed by the FIPA (FIPA: Foundation for Intelligent Physical Agents, 2002).

Multiagent systems research deals with societies or sets of self-interested agents. Thus, the agents may not have a common global goal although a system-wide global goal may exist. Here, cooperation mechanism have to be used as it cannot be assumed that each agent acts towards the same common goal, as the agents may have been constructed by different parties (Wooldridge, 2009). Thus, the multiagent system community focuses on questions of how and why agents cooperate (Wooldridge and Jennings, 1994), how agents can recognize and resolve conflicts (Adler et al., 1989; Galliers, 1988; Lander et al., 1991) or how agents can negotiate or search for compromises (Ephrati and Rosenschein, 1993; Rosenschein and Zlotkin, 1994).

In contrast to cooperation, coordination deals with how agents can act without endanger the goals of other agents (Wooldridge, 2009). Thus, the research on coordination mechanisms deals with the inter-dependencies between the agents' actions and activities. Coordination can be achieved through partial global planning (Durfee, 1988, 1996; Durfee and Lesser, 1987) or through joint intentions based on conventions (Jennings, 1993). Another possibility is mutual modeling, i.e. agents model the behavior of others and reason about it like it is done in the

MACE system where acquaintance models are used (Gasser et al., 1987) or through norms and social laws. In the latter case we can distinguish between the offline design of norms (Conte and Castelfranchi, 1993; Goldman and Rosenschein, 1994; Shoham and Tennenholtz, 1992b) and norms emerging in the system (Kittock, 1993; Shoham and Tennenholtz, 1992a; Walker and Wooldridge, 1995). All mechanisms are related as they manage the positive and negative relationships between actions (von Martial, 1990). Positive relationships are those where one action enables the execution of another action. In negative relationships, one action prevents the execution of another action.

When considering the organizational part of multiagent system's research we face several facets. On the one hand there is the concept of how parts of an agent are organized (cf. agents formed of smaller agents, i.e. Holonic Agents (Schillo and Fischer, 2003)). On the other hand agents may form coalitions (Shehory and Kraus, 1998) or agree to contracts for specific tasks (Dur and Roelfsema, 2010), hierarchies of agents can be formed (Deng and Papadimitriou, 1999; Jung and Lake, 2008) or agents may form collaborative networks (Boloni and Marinescu, 2000).

Finally, in the interaction part the inter-agent interaction is considered. If agents have preferences about actions or environment states, they tend to select high rated actions or actions that may lead to high rated environment states. Thus, they are assumed to behave individual rational, which means they try to optimize their own performance, or they try to maximize their received profit, if some monetary units are used to distinguish between successful and unsuccessful behavior. In this field of interest, mostly game theory is used to predict the agents' strategies. There, we distinguish between non-cooperative game theory (Nisan et al., 2007)—where we may try to calculate Nash equilibria (Nash, 1950)—and cooperative game theory (Peleg and Sudhölter, 2007)—where the core (Gillies, 1959) or the Shapley value (Shapley, 1953) are solution concepts.

2.2 Specific Terminology

In this section, we discuss the terms *learning*, *emergence*, *altruistic* and *egoistic* behavior.

2.2.1 Learning

The Encyclopædia Britannica defines the term *learning* as "the alteration of behavior as a result of individual experience. When an organism can perceive and change its behaviour, it is said to learn" (Encyclopædia Britannica, 2010). The Columbia Encyclopedia (International Encyclopedia of the Social Sciences, 1968) says:

> [...] learning in psychology, the process by which a relatively lasting change in potential behavior occurs as a result of practice or experience. Learning is distinguished from behavioral changes arising from such processes as maturation and

> illness, but does apply to motor skills, such as driving a car, to intellectual skills, such as reading, and to attitudes and values, such as prejudice. There is evidence that neurotic symptoms and patterns of mental illness are also learned behavior. Learning occurs throughout life in animals, and learned behavior accounts for a large proportion of all behavior in the higher animals, especially in humans.

In artificial intelligence (AI), the most common definition of learning is the process of behavior change based on previous experiences (Barr and Feigenbaum, 1982). Mitchell defines learning as the improvement with experience at some task with respect to some performance measure (Mitchell, 1997). Thus, the community agrees that learning has always something to do with the self-improvement of future behavior based on experiences (Sen and Weiss, 1999).

Several properties characterize a learning process. One distinction is based on the learning feedback, i.e. the performance signal for the agent. Gerhard Weiss gives the following distinction (Weiss, 1996):

- *Supervised learning*: The feedback specifies the desired activity of the learner and the objective of learning is to match this desired action as closely as possible.
- *Reinforcement learning*: The feedback only specifies the utility of the current activity of the learner and the objective is to maximize this utility.
- *Unsupervised learning*: No explicit feedback is provided and the objective is to find out useful and desired activities on the basis of trial-and-error or self-organization processes.

Another important property of the learning method is if the agent learns in an isolated form, i.e. in a single-agent system, or if the agent learns in the presence of other agents (Sen and Weiss, 1999). If the agent has to learn in the presence of other agents, the changes of the environment states are also influenced by the actions of other agents. Thus, the learning process gets harder as the received learning feedback is given for the joint action that may not be observable by the agent. There are two cases that makes it hard for the agent to realize if a selected action was good. It can be the case that the actions of the others favored the execution of the agent's action and, thus, a positive reward was generated. The other case can be that the other agents' actions had negative influence, which results in a lower or even negative reward.

2.2.2 Emergence

One aspect of complex systems composed of several individual entities like agents is the analysis of emergent behavior. Emergence describes the process of how such global behavior arises from the sum of individual behaviors (Bonabeau et al., 1999). One famous example of emergence is present in the well known "Game of Life" described in (Gardner, 1970). There, some rules are defined that activate or deactivate cells of a cellular automaton based on the activeness state of neighboring cells. One can observe different patterns that emerge because of the rules. The interesting thing about it is that none of the rules directly specifies these patterns. However, some stable or moving patterns can be seen as the result of the rules' execution. This behavior

of the system is called *emergent*.

For multiagent systems, emergent behavior can be both, positive and negative. Negative behavior is not intended by the systems designer and may be caused by forgotten dependencies. Positive behavior of course is mostly intended by the designer. Emergent behavior is especially needed if the agents are designed by different parties. Then, one cannot rely on for example cooperative behavior of the other agents due to their design. However, selfishness of the agents may lead to emergent cooperative behavior, which is one form of positive emergence.

As lower-level individual behavior influences the global system's performance one has to search for emergence when analyzing a complex system at hand. Without knowledge about emergent behavior of a system, one could face many problems if the system is running. Then, the obtained results can be contrary to the expected results. If the dependencies of the components are analyzes in a way that helps finding emergent behavior, it may help the developer to redesign the system in order to prevent negative emergent behavior.

2.2.3 Altruistic and Egoistic Behavior

The terms altruism and egoism come from social science (Dawkins, 1982). Somebody who acts egoistically would only do something if the person believes that this will result in a positive reward, which increases the individual welfare. In contrast, altruistic behavior is shown if a person would do something such that somebody else gets a favor. The extreme case of altruism is giving the life for saving another one's life (Wispe, 1978). In the study of social evolution, altruism refers to behavior by an individual that increases the fitness of another individual while decreasing the fitness of the actor (Bell, 2008).

Altruistic behavior can occur in different forms. There is the reciprocal altruism in which agent a only helps agent b at a's costs because agent a assumes to be in a similar situation once, and believes agent b to help a (Trivers, 1971). This is no pure altruism as reciprocal altruism arises from a selfish viewpoint. It is also very closely related to the strategy "tit-for-tat" known from game theory (Axelrod, 1984).

Another form of altruism is the so-called kin selection where individuals tend to helping behavior especially towards individuals that are closely related on the genetic level (Dawkins, 1982).

Prosocial behavior is also worth mentioning, here. Whenever an individual tends to an action that benefits the whole population, this individual is called *prosocially* behaving (Montada and Bierhoff, 1991). This is strongly connected to the *principle of social rationality* (Kalenka and Jennings, 1999): if an agent has a choice of actions, it should choose the action that maximizes the social welfare. Social welfare is defined as the sum of all agents utilities (Wooldridge, 2009). However, the social welfare of a whole group is a global property which is at least costly to compute or can even be incomputable from an agent's view.

2.3 Related Work

We now present related work from three research perspectives. First, we introduce the notion of artificial societies and describe related work from this area. Second, we focus on the task allocation problem and identify work that is related to this thesis. Third, imitation-based learning and memetics is considered. In all of the following sections we compare the related work to the work presented here and discuss differences. Additionally, we also introduce the three research perspectives briefly.

2.3.1 Artificial Societies

One kind of agent-based models are so called artificial societies. With the help of a multiagent system, a society of agents is created and emergent properties within this society are studied (Sawyer, 2003). In contrast to traditional simulation there is no pre-existing system that is modeled and analyzed in artificial societies (Hales, 2001). Moreover, the work done in artificial societies can be seen as form of theory construction in an abstract computational domain (Hales, 2001). Within this area, we can find evolutionary processes similar to those from evolutionary computing or cultural evolution based on thoughts and memes (Fog, 2005). Agents or society members may also die or reproduce like in real-world societies (Langdon, 2005). Models of artificial societies address possible societies of agents, their general processes, dynamics, and emergent properties within the society (Gilbert and Conte, 1995).

In (Castelfranchi et al., 1998), the role of normative reputation is analyzed. With the help of norms, agents' aggression is being controlled. Two sub-populations are considered in an artificial society where the first is controlled by norms and the second by an aggression strategy. The benchmark scenario is a food-discovery experiment. The agents that are controlled strategically attack only agents that are not stronger than themselves. Normative agents do not attack agents that are currently eating. Additionally, a reputation mechanism is used to allow the normative agents to identify those agents who do not follow the normative rules. Normative controlled agents have been proven to work better than strategic ones if they can communicate their experiences. The results show that the strength distribution in the normative sub-population is more equally in comparison to the strategic sub-population.

In (Cecconi and Parisi, 1998), an artificial society is introduced where agents are situated in an environment and need resources to survive. Two strategies are introduced and compared. The first is the individual survival strategy (ISS) and the second is the social survival strategy (SSS). The difference between both strategies is that agents who follow the SSS strategy contribute parts of their resources to a central mechanism that can redistribute these resources to other agents. It is shown that groups following the SSS strategy can survive even in less favorable environments where groups following ISS die. These studies are meant to explain the emergence and maintenance of the SSS strategy of human societies but this goal is not achieved properly as the strategy is shown not to be robust against cheating agents or agents that may decide on their

own which strategy they choose.

Zimmermann et al. (2001) consider the famous Prisoners' Dilemma (PD) in multiagent systems. In PD agents can decide to cooperate or to defect. Mutual defection is punished and mutual cooperation is rewarded. If an agent defects and the other agent cooperates, the defector gains more reward than in the mutual cooperation case. Thus, a dilemma exists as the individual best strategy is to defect in this one-shot game, but the global best strategy is mutual cooperation. In the work of Zimmermann et al., agents are linked to other agents through a network structure, and they play the game sticking to the same strategy with all neighbors. They repeatedly play the game in form of rounds without knowing how many rounds will be played. After receiving the payoff for a round, the agents reconsider their strategy. If an agent has the highest payoff within its neighborhood, it sticks to the current strategy and otherwise it copies the strategy of the neighbor with highest payoff. Additionally, if the agent does not have the highest payoff and is playing the defect strategy it may change its neighborhood with some probability through exchanging a defect-playing neighbor by a randomly chosen agent of the population. The results show that with highly dynamic networks, i.e. high exchange probabilities, the system reaches a stable state with almost all agents playing the cooperative strategy. The work presented here also has a reward and punishment function that leads to an individually best strategy of non-cooperation. However, in contrast to the work of Zimmermann et al., the agents in this thesis do not want to be the locally best agent but they want to belong to one of the bests. Additionally, the agents only change neighbors but do not consider, if the neighbor was cooperative.

Riolo et al. present experiments where the decision to cooperate is based on the similarity of observable markers, so called "tags" (Riolo et al., 2001). Agents interact with randomly chosen opponents and are not likely to meet again. If $|\tau_A - \tau_B| \leq T_A$ is fulfilled, agent A will cooperate with agent B, where τ_A is the observable marker of agent A and T_A is agent A's tolerance value. Both values are drawn from the interval $[0, 1]$. Agents that cooperate have to pay some cooperation costs and the recipient is rewarded with some payoff. After an interaction phase, the agents adapt. Two randomly chosen agents compare their received payoff and the worse agent copies the threshold and tag values of the better performing one. Riolo et al. show that dominant tag groups take over the population and high cooperation rates are achieved. In contrast to the model presented here, they only have a single inequation that has to be fulfilled for cooperation. We concentrate on models that are more realistic and where this decision process is based on a number of inequations. Secondly, we will see that perfect imitation does not lead to the best results. The agents should only be allowed to move their values to a specific direction but they should not able to copy the values. A third aspect is that the agents in our model are not able to sense the others thresholds or even to modify their thresholds. They are randomly chosen at the beginning and fixed over the simulation. Riolo et al. described under which conditions the high cooperation rates can occur but they only used experimental results. We will examine the cooperation probability in our model and show theoretically under which conditions cooperation can emerge.

In (Hales, 2002a,b), experiments are presented where agents are equipped with one skill out of a specific skill set. Agents are given resources that they could harvest only, if the required

skill of the resource matches their own skill. Otherwise, the agent could pass the resource to another agent. The agents also have a so-called tag $\tau \in [0, 1]$ and a threshold $0 \leq T \leq 1$. A *donator* agent D could only give a resource to a *receiver* agent R if $|\tau_D - \tau_R| \leq T_D$. Harvesting resources is rewarded with a payoff of 1 and searching for another agent is rewarded with a negative payoff where the height depends on the searching method. Hales' approach uses an evolutionary algorithm with a kind of tournament selection and slightly mutation. The results show high donation rates for good searching methods. The mechanism of comparing tag value differences to a specific threshold is extended by us in this thesis to model more flexible decision making processes that are not based on a single condition.

In (Hales, 2005), Hales presents a protocol for a decentralized peer-to-peer system called SLAC ("Selfish Link and Behavior Adaptation to produce Cooperation"). There, agents have simple skills and are awarded jobs. Each job only requires a single skill. If an agent does not provide the required skill, it asks an agent from its neighborhood to complete the job. A boolean flag indicates whether an agent behaves altruistically. Altruists will cooperate, if another agent asks for it. Completed jobs are rewarded with a benefit of 1 and cooperation produces costs of 0.25. Agents are allowed to exchange jobs within their neighborhoods, which are modeled as a subset of the population. During the simulation, agents are pairwise compared and the agent with lower utility value in a simulation round is changed. This means, that the neighborhood is destroyed and the agent creates a link to the better performing agent and copies its strategy (altruist or egoist). With some probabilities, the agent also mutates its skill, the altruism flag and the neighborhood after this step. These drastic changes in the network structure lead to good results concerning the job-completing rate. In contrast to the work of Hales, we will not consider such drastic changes in the network. Although the neighborhoods may change over time, the dynamics of the network are lower. From this aspect we expect more stable neighborhoods and adaptation to neighboring agents.

In (Seredynski et al., 2010) a Cellular Automata (CA)-based multiagent system is considered where the agents (cells) play the iterated Prisoner's Dilemma game with the neighboring agents for a number of times which is unknown to the agents. As the CA specifies the interaction possibilities and the game is played a number of times, it is called the spatio-temporal generalized Prisoner's Dilemma. Agents can choose one of the three strategies $all\ C$ (always cooperate), $all\ D$ (always defect) and k-D, where the last means that the player chooses action C until more than k of it neighbors use strategy D. In between the iterations, agents can change their strategy by choosing the strategy of their best performing neighbor. Through this evolution, regions of cooperative cells can emerge. The authors claim that the findings are very promising in the area of cooperative enforcement in ad hoc networks.

2.3.2 Task Allocation Problem

A problem that arises in distributed computer systems is the so-called task allocation problem (TAP). The TAP naturally occurs whenever a set of tasks should be allocated to a set of processing units. There, each task execution produces costs. These costs can be different if the

same task is executed on different machines. Additionally, tasks may produce communication costs if they are executed on different processing units, which can be neglected, if the tasks are executed by the same processing unit. The goal is to find an assignment of tasks to machines that minimizes the overall costs. According to (Lewis et al., 2003), the TAP can formally defined as follows:

Definition 2.2 (Task Allocation Problem): *Given a set* $\mathbf{P} = \{P_1, P_2, \ldots, P_m\}$ *of processing units and a set of tasks* $\mathbf{T} = \{T_1, T_2, \ldots, T_n\}$ *that should be processed.* $c_{t,t'}$ *are communication costs for tasks* T_t *and* $T_{t'}$ *if they are processed by different processing units. If tasks have to communicate and are processed on the same processing unit the costs for communication are negligible.* $q_{t,p}$ *are the execution costs if task* T_t *is executed by processing unit* P_p*. Let* $x_{t,p}$ *be a boolean decision variable that is equal to* 1 *if task* T_t *is assigned to processor* P_p *and* 0*, otherwise;* $1 - x_{t,p}$ *is denoted by* $\overline{x}_{t,p}$*.*
The Task Allocation Problem (TAP) *is to find a task-processor assignment that satisfies*

$$\min \sum_{t=1}^{n} \sum_{p=1}^{m} q_{t,p} \cdot x_{t,p} + \sum_{t,t' \text{ with } t<t'} \sum_{p=1}^{m} c_{t,t'} \cdot x_{t,p} \cdot \overline{x}_{t',p}$$

and

$$\forall t \in [1, n] : \sum_{p=1}^{m} x_{t,p} = 1$$

$$\forall t \in [1, n] \; \forall p \in [1, m] : x_{t,p} \in \{0, 1\}$$

The TAP is known to be efficiently solvable by a polynomial-time algorithm if only a two-processor system is considered (Stone, 1977). However, for an arbitrary number of processors, the problem has been shown to be $\mathcal{NP}$-complete (Lo, 1988).

Dahl et al. (2003) consider task allocations in homogeneous robots, i.e. the robots do not differ. They propose an algorithm based on vacancy chains, typically known from human and animal societies (Chase et al., 1988). In a vacancy chain, a new resource unit coming into a population is taken by a first individual who leaves his/her old unit behind, this old unit is taken by a second individual leaving his/her old unit behind, and so forth (Chase, 1991). The proposed algorithm uses local task selection, reinforcement learning for estimation of task utilities, and a reward structure that is based on the vacancy chain framework. The allocation of tasks to robots is shown to be sensitive to the dynamics of the group while being completely distributed and communication-free. In contrast to their work, we will consider heterogeneous agent systems where the agents differ in their abilities.

Shehory and Kraus use coalition formation to calculate task allocations among agents (Shehory and Kraus, 1995). They use a variant of TAP where multiple agents can work on the same task in order to provide all resources that are needed for the task's execution. Although the TAP has been proven to be computationally exponential, the authors present a polynomial complexity algorithm that yields results which are close to the optimal results and prove that the solution

quality is bounded by a logarithmic ratio bound. The presented algorithm is a distributed one as each agent mostly performs those calculations that are required for its own actions during the process. They distinguish between disjoint coalitions, i.e. each agent can only belong to at most one group at one time step, and overlapping coalitions, i.e. each agent may belong to several groups simultaneously Shehory and Kraus (1996). Note, that in their work the agents only consider the overall system's performance in contrast to their personal payoff (Shehory and Kraus, 1998). The agents in the thesis at hand are not able to calculate the coalitions as for coalition formation all the tasks have to be known a priori in order to calculate an optimal or near-optimal solution. We consider dynamic task allocations and, thus, coalition formation is not applicable, here.

Kraus et al. (2003) present another approach for task allocation through coalition formation. They assume that agents are truthful, i.e. they never lie about individually available resources. The authors use a manager agent as a central instance that distributes the tasks to the agents based on their resources and execution costs for performing the tasks. They propose an efficient algorithm that solves the TAP based on that global view. In contrast to their work, we will only consider local decision making without the need of global knowledge.

In (Manisterski et al., 2006), uncooperative agents are considered. The agents' resources are again known to all agents but their task's execution costs are private knowledge. Thus, the agents are able to lie about the costs. Additionally, they can decide on their own to join a group for a specific task or not. They show that in this general case of the problem no protocol achieving the efficient solution can exist that is individual rational and budged balanced. We will not consider agents, which know the available resources of their neighbors. Although, they can sense the skills of their neighbors, they are not able to detect how many units are available, when capacity constrained agents are considered in this thesis.

Upadhyaya and Lata (2008) consider task allocation in distributed database systems and compare this problem to the standard task allocation in distributed systems. They claim that this is much more complex as data and tasks are distributed in contrast to the standard TAP where only the tasks are distributed. Several additional problems arise such as data fragmentation, data replication or reallocation of data.

Another related problem domain is the Request for Proposal (RFP) proposed by Chan and Leung (2009). There, task managers need to recruit service provider agents to handle complex tasks composed of multiple sub-tasks. The objectives are to assign each sub-task to a capable agent while keeping the costs as low a possible. Chan and Leung propose an efficient multi-auction based approach that can calculate near-optimal solutions. The basic idea is to have a series of reverse English auctions, one for each sub-task. Participating agents are allowed to submit bids for one sub-task at a time. After bids have arrived at the auctioneer, it verifies that two properties hold. First, the sub-task being bid for must be in the capability set of the bidder agent. Second, the bidding amount must be lower than the current outstanding bid and the reservation price for the sub-task. The reservation price is predefined by the auctioneer. It is an upper bound to the allowed payment for task execution. With the help of this additional

verification step, the authors show that their proposed mechanism can reach near-optimal task allocations in private environments, where task execution costs constitute private knowledge.

Other approaches exist for the TAP, e.g. the work presented in (Gupta and Greenwood, 1996). The authors use an (μ, λ) evolutionary strategy (Eiben and Smith, 2003; Fogel, 2006) to compute task allocations that minimizes the overall scheduling time.

2.3.2.1 Dynamic Task Allocation

The dynamic formulation of the Task Allocation Problem is an essential requirement for multi-robot systems operating in unknown dynamic environments (Lerman et al., 2006). It allows robots to change their behavior in response to environment changes or actions of other robots in order to improve overall system performance (Lerman et al., 2006). The main difference to the TAP is that the task allocation is a dynamic process that has to be adjusted continuously in response to changes on the task environment or group performance. The purpose of dynamic task allocation is to increase the system throughput in a dynamic environment, which can be realized by balancing the utilization of computing resources and minimizing communication between processors during run time (Chang and Oldham, 1995). If the agents have different processing times for different tasks' requirements, the problem is said to be heterogeneous and homogeneous otherwise.

Ghizzioli et al. (2004) study algorithms for the homogeneous and heterogeneous case and present an ant-based algorithm that solves the problem. The so-called ATA algorithm is inspired by the division of labor of social insects and based on the work of Cicirello and Smith (Cicirello and Smith, 2001). ATA introduces four specific rules for the agents in order to speed up the adaptation process for particularly unpredictable instances of the dynamic task allocation problem. They considered as a benchmark problem a big factory with painting agents that color trucks on demand. This problem is one of the often-used benchmark scenarios for dynamic task allocation.

Lerman et al. (2006) propose a mathematical model of a general dynamic task allocation mechanism. There, agents have to choose between two types of tasks without communication and global knowledge. The agents estimate the state of the environment through repeated local observations. The authors show that the mathematical model can help the designer of a system to properly choose agent properties in a multi-foraging task scenario.

In (Dai et al., 2009) an approach is presented where tasks arrive dynamically at a contractor who starts a second-price auction on the requested tasks' requirements. Bidding agents submit prices, they want to achieve for accepting a task. The agent with the lowest bid wins but is rewarded with the second-lowest price. This type of auction is also known as Vickrey auction (Shoham and Leyton-Brown, 2009). If a busy agent wins a new task, it has to decommit from its current task and has to pay a decommitment fee. Dai et al.'s main contribution is a mathematical formulation of the problem based on Partially Observable Markov Decision Process (POMDP)

(Kaelbling et al., 1998). They show that with the help of this mathematical model a system's designer can be helped in revealing benefits of a system under consideration. Especially the used strategy of decommitment is shown to be very useful in this context in order to benefit on the global performance level.

Chapman et al. use a Markov game formulation for a decentralized dynamic task allocation problem (Chapman et al., 2009, 2010). They analyze the approach with tasks having varying hard deadlines and processing requirements in the RoboCup Rescue scenario[1]. They approximate these games with series of static potential games and then construct a decentralized solution method for the approximated games using the Distributed Stochastic Algorithm. Within the benchmark scenario of RoboCup Rescue, they show that their approach is comparable to a centralized task scheduler and that it is robust to restrictions on the agents' communication and observation range, which is not the case for the centralized scheduler.

2.3.2.2 Task Allocation in Social Networks

de Weerdt et al. (2007) calculate task allocations with a distributed algorithm in a social network. A social network is a graph of agents as nodes and links between them describing possible interactions. The tasks are assigned to agents with limited resource amounts. In this paper the term *social task allocation problem* (STAP) is introduced. They show that the problem of finding an efficient task allocation, i.e. which maximizes the social welfare, is $\mathcal{NP}$-complete. They do not model cooperation costs, which is different to the work presented here. The agents also know about all tasks before the decision process is started, thus they do not deal with a dynamic environment. In (de Weerdt and Zhang, 2008), they analyze the problem from a mechanisms design perspective and give very good theoretical insights. They show that it is similar to combinatorial auctions (de Vries and Vohra, 2003). There, the agents report their resources and are allowed to strategize over them. They propose a Vickrey-Clarke-Groves (VCG) payment method (Clarke, 1971; Groves, 1973; Vickrey, 1961) to prevent under-reporting.

Kok et al. deal with coordination graphs in RoboCup Soccer Simulation (Kok et al., 2003; Vlassis et al., 2004). Coordination graphs represent the coordination requirements of a system (Guestrin et al., 2002). They assign roles to the agents in order to abstract from a continuous state space to a discrete state space, which allows them to apply existing techniques for discrete-state coordination graphs. They show the successful application of this approach to the RoboCup Soccer scenario.

Abdallah and Lesser define the *coalition formation problem* (CFP) in a formal way (Abdallah and Lesser, 2004). They work with hierarchical organization structures. Their main contribution is an organization-based distributed algorithm for approximately solving the CFP. They use reinforcement learning to optimize the local allocation decisions made by agents in the underlying organization.

[1] `http://www.robocuprescue.org`

Gaston's and de Jardins' research is about dynamic environments where agents have to fulfill tasks. In (Gaston and desJardins, 2005a), *agent-organized networks* (AON) are firstly defined. There, agents have local perception of the global performance. The agents are equipped with a skill σ_i out of the set $\{1, 2, \ldots, \sigma\}$. The paper mainly focuses on different network-rewiring strategies (i.e. structure-based and performance-based). The overall goal is to access the feasibility of designing realistic and efficient strategies for distributed network adaption. In (Gaston and desJardins, 2005b), the focus is on providing a framework for bottom-up AONs to improve the feasibility of applying AONs to improve the performance of an organization of economically motivated agents. They give a formal model of production and exchange. Agents are situated in a social network and are only allowed to trade and negotiate with direct neighbors. They show that the network structure has great influence on the results (Gaston and desJardins, 2008). In the latter work, they give graph theoretical foundations and shows three characteristics which influence the results, namely blocking (topologically and skill), carrying capacities of different graphs and diversity support of the networks. Scale-free networks perform best in this paper, but have the problem of single points-of-failure as they have a structure build of hubs.

Distributed team formation in a networked multiagent system is considered in (Bulka et al., 2007). Agents are situated in a social network and dynamically build teams in order to complete introduced tasks. The social network structure explicitly restricts the set of agents that can form teams. This is done through sensing states of neighbors. The states can be *uncommitted* in the case of an idle agent, *committed* if an agent has decided to join a team for a task but the current team does not provide enough skill capacities or *active* if the agent is currently working on a task within a team. The agents learn team joining and team initiating strategies based on their previous experiences. For this, the agents estimate probabilities for different actions. The results show that this approach outperforms a random selection strategy in most of the considered network structures.

2.3.3 Imitation-based Learning and Memetics

Imitation-based learning is one specific form of learning where agent a tries to replicate the behavior of another person b without knowing the intentions behind this imitated behavior. This learning procedure can also be identified in animals, for example in apes like Rhesus Macaques (Subiaul et al., 2004). Learning by imitation is strongly connected to learning by example (Mitchell, 1997). Whenever an imitator wants to imitate some behavior it has to select an *ideal actor* or a *role model* of whom the behavior should be imitated. One key aspect of imitation-based learning is that this process of learning is very error-prone as the imitator may not be able to perceive all relevant information about the observed behavior like the intention or if the observation was correct. Besides others, robotics is a typical application area of imitation-based learning (Billard and Matari, 2001).

In (Priesterjahn, 2008), an imitation-based learning algorithm is proposed. We present a variant in Algorithm 2.1. After initializing a population $\mathcal{A}$ of n agents a repetition-loop is started until some stopping criterion is reached. Within the loop two parts are important. First, the

Algorithm 2.1 Local imitation-based learning algorithm.

Input: number of agents n

1: **procedure** IMITATION(n)
2: initialize population $\mathcal{A}$ of n agents
3: **repeat**
4: evaluate $\mathcal{A}$
5: **for all** agents $a \in \mathcal{A}$ **do**
6: determine the set of locally best agents $\mathcal{E}_a \subseteq \mathcal{N}_a \cup \{a\}$ of the so-called *closed* neighborhood /* LOCAL ELITE SELECTION */
7: **if** $a \notin \mathcal{E}_a$ **then**
8: select set of ideal agents $\mathcal{I}_a \subset \mathcal{N}_a$ /* ROLE MODEL SELECTION */
9: agent a adapts to $\mathcal{I}_a$ /* IMITATION */
10: **end if**
11: **end for**
12: **until** end condition reached
13: **end procedure**

population $\mathcal{A}$ is evaluated with some mechanism. Then, the local part starts which is executed by each agent. An agent a determines the set of so-called *elite* agents which are those agents that performed best in the evaluation and which belong to the *closed* neighborhood of agent a. The closed neighborhood is defined as it is known from graph theory (Harris et al., 2008), i.e. all neighbors $\mathcal{N}_a$ of the agent a and the agent itself belong to the closed neighborhood. If the agent does not belong to the set of elite agents it selects a set of *ideal* agents which are considered as role models. This selection process can be done in an arbitrary way although selection from the elite set would be reasonable. However, as the set of ideal agents can be greater than the elite set in the local algorithm the ideal agents are selected from the neighborhood in general. Finally, the agents adapts in some way to the set of ideal agents.

Richard Dawkins firstly introduced the term *meme* in his book "The Selfish Gene" (Dawkins, 1976), which deals with cultural evolution. Since then many articles concerning learning with different kinds of memes and similar approaches have been published. A meme is a particular thought or idea, which transfers itself from individual to individual. During this process, a meme could be changed or mutated through imitation. Therefore, Dawkins used the term meme as an analogy to gene. Thus, meme-based approaches are related to the area of evolutionary computation (Eiben and Smith, 2003).

In (Gabora, 1996) Liane Gabora speaks of a "second form of evolution". She points out differences between biological and cultural evolution such as different solution finding strategies. In her opinion, the biological evolution is more like a breadth-first search, because variations are generated randomly. As variants in cultural evolution are generated strategically, she points out that it is similar to depth-first search.

By trying to define “memes ” Blackmore (1998) states:

> *“We may treat the spread of memes as comparable with the spread of infectious or contagious diseases and use models derived from epidemiology.”*

She describes the differences between individual learning such as classical conditioning or operant conditioning and memetic learning. With her definitions, she wants to help to distinguish between what is and what is not a meme.

Hodgson and Knudson also deal with memetics but from the viewpoint of firms as one application domain (Hodgson and Knudsen, 2004). They state, that a replication is a relationship between a copy and some source exhibiting the four characteristics causation, similarity, information transfer, and duplication. Instead of using the term meme, they refer to habits and routines. The firm is an actor in their view. Hodgson and Knudson again deal with replicators in (Hodgson and Knudsen, 2008). They describe a von Neumann’s view of self-reproducing automata as systems that build systems and increase the complexity or the functions of the systems. This is an inspiration for their *generative replicator definition*. They state that a generative replicator must meet four attributes: causal implication, similarity, information transfer, and conditional generative mechanisms.

Kuperman et. al (Kuperman et al., 2006) analyze disease propagation. They say, that the study of diffusion in different media (where transitions are not necessarily limited to local sites, but may include jumps to distant sites), is highly applicable to certain social phenomena, among other applications within computational complexity, neurobiology, and economic exchange. They start with fully connected clusters with mutation probabilities on the links, which lead to networks that exhibit the same properties as small-world networks (Watts, 1999) (i.e. high cluster coefficient and small average path length). To start with fully connected clusters eases the analysis of the diffusion process.

Haggith et al. (2003) give an overview of the theories about diffusion of ideas. They deal with spatial neighborhoods, which is similar to Axelrod’s model of cultural diffusion (Axelrod, 1997). They analyze seven different kinds of social relationships such as arbitrary neighbors in a social network, networks with agent nodes having exactly two neighbors or ten neighbors or hierarchies. Additionally, they point out the difference between copy-the-idea and copy-the-carrier. The mechanism of copy-the-idea tries to copy the specific parts of an individual where the copy-the-carrier mechanism tries to copy the person who has an interesting idea as such.

Henrich et al. (2008) describe five misunderstandings about cultural evolution and give counterexamples to verify their opinions. They state that cumulative adaptive evolution does not depend on replication, fidelity, or longevity. For example, they say that the representation of a meme needs not to be already there when cultural evolution occurs. Another important misunderstanding in their view is that the “cultural fitness” of a mental representation can be inferred from its successful transmission through the population. Memes often have nothing to do with the status of the people who are the carriers. They also say, that a meme‘s mimetic fitness will depend jointly on how attractive its content is to human brains and how it affects an individual‘s

likelihood of being selected as a cultural model by other individuals. This is different to the definition of genetic fitness.

In (Rocha et al., 2005), there is a nice introduction into the topic of memetics. They formally define "learning by imitation" and give mathematical models for it. They also claim, that the development of mathematical tools to deal with the dynamics of meme transmission would be subject of future investigations, related to the meme-gene coevolution of mathematical memeplexes, which are groups of memes.

Richert considered the question how a group of robots can learn to achieve a shared goal and how they can imitate each other in order to increase the performance and the speed of learning (Richert, 2009). A dedicated robot architecture is presented build-up with three layers. In the motivation layer, the overall goal is specified. Feedback on executed actions is provided to the robot, which is used in the strategy layer. Based on the feedback, the current policy of the robot is updated. For each symbolic action of the strategy layer, there exist a low-level representation in the skill layer. A robot is able to learn individually and to switch into some observation state to observe the behavior of others. Based on the observation, a robot extracts information and integrates it into its knowledge base. This imitation step helps the robots to speed up the learning process. Besides this, the robots calculate differences to other robots in the population in order to select the best role model, which is the robot with less difference. With the help of this technique, it is shown that the approach works also in heterogeneous robot system where the robots differ in their capabilities.

Nguyen et al. (2008) deal with memetic algorithms. Memetic algorithms are not algorithms that imitate cultural evolution but use memetics in another way. A memetic algorithm (MA) is a genetic algorithm (GA) with an additional local search to optimize individuals in a non-genetic way. Nguyen et. al name those algorithms 1^{st} generation MAs which do not have an embedded meme transmission. 2^{nd} generation MAs are also named "multi-meme GA" (Nguyen et al., 2008). Here, the memes compete with each other in a pool and simple inheritance mechanisms are embedded into the algorithm. When an algorithm exhibits co-evolution and self-generation of memes they are called 3^{rd} generation MAs. Besides this classification, they introduce a Diffusion-MA (DMA) which bases on the structure of a cellular genetic algorithm.

Azevedo and Gordon (2009) describe a new class of MAs: Terrain-Based MAs (TBMA). They describe vector quantization and present the K-means algorithm. In K-means, the distortion decreases monotonically, since the codebook is updated to satisfy the nearest neighbor rule and the centroid condition. Thus, K-means may be regarded as a hill-climbing technique. After this, they present an accelerated K-means algorithm. Their proposed TBMA is based on the von Neumann-neighborhood, which is called Deme-4 neighborhood in their paper.

2.4 Conclusion

In this chapter we introduces the reader to definitions of agents and their main properties. We also discussed terminology that is used in this thesis. Finally, we presented related work from three research perspectives. We pointed out which related work influenced this thesis and we have presented differences to the existing approaches.

If we consider the presented related work, we can observe that none of the presented approaches satisfies all the properties specified in Section 1.1. One unsatisfied property is one of our main properties, i.e. local view. For example the work on coalition formation requires that the agents can calculate coalition values from a global perspective. For scalability purposes, global knowledge should always be avoided in multiagent system that have a large number of agents. Additionally, the property of minimal knowledge is violated, if a global view is available to the agents.

In the next chapter, we will present our local adaptation approach, that satisfies the desired properties known from the introduction of this thesis.

3 Formal Model and Scenario Description

As we have seen, there exists a number of models for multiagent systems in literature. We are interested in multiagent systems where the agents only have a local view, which means they are not able to know every other agent or the number of agents in the system. Additionally, we are interested in cooperation decisions based on multiple criteria, as it is the case in everyday life between humans or animals. It is well documented that people tend to cooperate with others, who are similar to them (Tajfel et al., 1971). In literature, there are many psychological studies which claim that individuals within groups are highly oriented towards their own group and that they are actively trying to harmonize their beliefs (Kramer and Brewer, 1984; Leyens et al., 1994; Oakes et al., 1994). We present a mechanism that is based on similarities between agents and where agents tend to cooperate, if the difference is not too large in a subjective, agent-oriented view. Furthermore, we consider a scenario where the agents have to fulfill jobs composed of several smaller tasks as a benchmark scenario.

The main results of this chapter are the following:

- We formally define the considered multiagent system.
- The cooperation willingness is based on similarities of agents in this system.
- We propose a new adaptation-based learning mechanism in Section 3.1 to promote the cooperation decisions that meets the objectives presented in Section 1.1.
- Cooperation decisions are one-sided but lead to emergent mutual cooperation.
- We introduce a job/task model as a benchmark scenario for this thesis.

In the following, we give the formal definition of the proposed multiagent system and the algorithms and describe the considered job/task scenario which is used as a proof-of-concept model. The proposed model and benchmark scenario have previously been published in (Eberling, 2009) and used in (Eberling and Kleine Büning, 2010a,b, 2011).

3.1 Agents and Multiagent System

In this thesis, we deal with agent networks that can be viewed as graphs. We use this analogy to define local neighborhoods in our system. We do not consider multiagent systems that are situated in two- or three-dimensional environments, as it can be found in other approaches (Priesterjahn, 2008). The reason for this decision is that in such systems the spatial distribution has to be considered in order to describe the neighborhood of an agent as it is done for example

with the k-nearest neighborhood in (Goebels, 2007). Since we concentrate on the adaptation mechanism, we use neighborhoods based on a network structure for the considered multiagent system. This concept of neighborhoods prevents the agents of having global knowledge, e.g. the whole population of agents. The systems we deal with have thousands of agents, which means that it is not applicable to allow the agents to interact or communicate with all other agents. Therefore, we define a special agent network called *interaction network*.

Definition 3.1 (Interaction Network): *An interaction network* IN *is a graph* $\mathsf{IN} = (\mathcal{A}, \mathcal{N})$ *where*

- $\mathcal{A}$ *is a finite set of agents*
- $\mathcal{N}$ *are links between the agents*

The links between the agents are modeled as undirected edges. An interaction network is called dynamic *if the network can change between successive simulation steps by adding and removing links.*

With the help of the interaction Network IN we are able to formally define our understanding of a neighborhood of an agent. Note also that due to the interaction network the agents' view on the system is local. The agents are not allowed to sense or interact with agents they are not directly connected to. However, with the help of communication patterns agents can get aware of their neighbors' neighbors if this mechanism promises advantages for the agents.

In our system, the agents have to fulfill different jobs consisting of smaller tasks. This allows us to model different granularities of jobs. In literature, one often finds multiagent systems, where only atomic tasks are considered, see for example (Hales, 2001, 2002b). With the distinction between *jobs* and *tasks* we want to model complex cooperation patterns where multiple agents have to work on a single job but on different tasks. With the help of different granulated jobs, we are able to consider more sophisticated job models. Each task requires a specific skill out of a system-wide skill set $\mathcal{S}$. Formally:

Definition 3.2 (Tasks, Jobs): *A task* t *is a pair* $t = (s_t, q_t)$ *where*

- $s_t \in \mathcal{S}$ *is the skill that is required to fulfill task t*
- $q_t \in [q_{\min}, q_{\max}] \subset \mathbb{R}_0^+$ *is the non-negative payoff for task fulfillment*

$\mathcal{T}$ *is the finite set of all possible tasks.* $\mathcal{J} \subseteq Pow(\mathcal{T})$ *is the set of all jobs. A job* $j \in \mathcal{J}$ *is a set of tasks* $j = \{t_1, \ldots, t_n\}$ *where* $t_{\min} \leq n \leq t_{\max}$ *with* $t_{\min}, t_{\max} \in \mathbb{N}$ *are the minimum and maximum number of tasks a job consists of. A job is only fulfilled if all its tasks are fulfilled. The payoff for a job is the sum of the tasks' payoffs if it is fulfilled and zero otherwise. Formally,*

$$\mathsf{payoff}(j) := \begin{cases} \sum_{t \in j} q_t & \textit{if job j is fulfilled} \\ 0 & \textit{otherwise} \end{cases}$$

A task can only be processed by an agent, if the agent provides the required skill.

In the formal model, we only consider tasks that can be processed without time constraints. Additionally, we do not consider any capacity constraints of the agents. At any given time step an agent can work on an arbitrary number of tasks and the agent will finish with all tasks in the same time step. There is no delay due to processing times.

The agents that are considered in this chapter have to search for cooperation partners that help them completing their jobs. As this cooperation process is based on many criteria in real life domains we also consider this determination with whom to cooperate on a set of criteria. The agents share a set of propositions that is part of the environment. These propositions can be opinions about the overall world state or the evolution of the environment. As we do not concentrate on the modeling of such propositions, we only take them as an abstract mean to model multidimensional decision making. A proposition p can represent anything like "The road is clear" in the context of a taxi-driving agent or "The color blue is prettier than black". For our purposes it is enough to know that there are propositions that influence the behavior of the agents and that these proposition form a common knowledge as all agents know about them.

Now we are able to define the environment in Definition 3.3 and to give a formal definition of the considered agents in Definition 3.4. To favor readability, we omitted a simulation step variable at the functions and sets used in the definitions. For the environment, the interaction network may change and the job set is different in every simulation step. Additionally, everything of the agents but the skill set and threshold vectors are changed over time and, thus, would need an index for the current simulation step.

Definition 3.3 (Environment): *An environment $\mathbb{E}$ that contains the agents from $\mathcal{A}$, that are linked in an interaction network IN following some construction mechanism, is a quadruple $\mathbb{E} = (\mathcal{S}, \mathcal{P}, \mathsf{IN}, \mathcal{J})$ where*

- *$\mathcal{S}$ is a finite, non-empty set of skills*
- *$\mathcal{P} = \{p_1, \ldots, p_m\}$ is a set of propositions*
- *$\mathsf{IN} = (\mathcal{A}, \mathcal{N})$ is an interaction network*
- *$\mathcal{J}$ is a set of jobs*

Definition 3.4 (Agent): *An agent $a \in \mathcal{A}$ is a tuple $a = (\mathcal{S}_a, \mathcal{N}_a, \mathcal{C}_a, \mathcal{V}_a, \Theta_a)$ with*

- *$\mathcal{S}_a \subseteq \mathcal{S}$: the set of skills agent a is equipped with*
- *$\mathcal{N}_a \subseteq \mathcal{A}$: agent a's set of neighbors*
- *$\mathcal{C}_a \subseteq \mathcal{N}_a$: set of neighbors, agent a is willing to cooperate with*
- *$\mathcal{V}_a \in [0, v_{\max}]^m \subset \mathbb{R}^m$: proposition rating vector*
- *$\Theta_a \in (0, \Theta_{\max}]^m \subset \mathbb{R}^m$: threshold vector*

The set of neighbors $\mathcal{N}_a$ of agent a is defined through the interaction network, i.e. there has to be a link $e = \{a, b\} \in \mathcal{N}$ for every $b \in \mathcal{N}_a$. Furthermore, only the skill set $\mathcal{S}_a$ and the rating vector $\mathcal{V}_a$ are modeled as observable properties of agent a.

As we defined in Definition 3.4, the agents give values to the system-wide set of propositions.

These values influence the behavior of the agents. If there is a high-valued proposition "The road is clear" in the taxi-driving example, the agent will drive faster than it would do if the proposition has a low rating. Another common interpretation of the agent's proposition-values is confidence with the opinion the proposition stands for. To keep notations simple, we use $\mathcal{V}(p)$ to denote the corresponding vector position that belongs to proposition p. With the help of this concept we want to model similarities between the agents.

Social science analyzes the behavior of animals and humans since many years. One aspect of this research is, that individuals that seem to have common ideas or common interests behave more cooperative than they would if they would not have these similarities (Hinde and Groebel, 1991; Kramer and Brewer, 1984). This aspect is captured in the determination process of cooperation partners in the system. The next definition describes this process formally:

Definition 3.5 (Cooperation Partners): *The set of cooperation partners $\mathcal{C}_a$ of agent a is defined as*

$$\mathcal{C}_a = \{b \in \mathcal{N}_a \mid \forall p \in \mathcal{P} : |\mathcal{V}_a(p) - \mathcal{V}_b(p)| \leq \Theta_a(p)\}. \tag{3.1}$$

This formula means, that every agent b from the neighborhood of agent a has to have proposition ratings such that for every proposition the distance is less than the allowed distance for that proposition, which is defined by the threshold vector.

We also define a cooperation relation $\mathcal{C} \subseteq \mathcal{A} \times \mathcal{A}$ based on the sets of cooperation partners:

$$b \in \mathcal{C}_a \Leftrightarrow (a, b) \in \mathcal{C}$$

According to Definition 3.5, it is easy to see that the relation $\mathcal{C}$ in general is not symmetric. Suppose there are two agents a and b and there is only one proposition p in the system. Let the values be $\mathcal{V}_a(p) = 20$, $\mathcal{V}_b(p) = 50$ and the thresholds $\Theta_a(p) = 40$, $\Theta_b(p) = 20$. As the difference between the values is 30 it holds that $(a, b) \in \mathcal{C}$ and $(b, a) \notin \mathcal{C}$. This means that agent a is willing to cooperate with agent b but not vice versa.

To summarize, we can see that there are different properties our model reflects. First of all, we do not have a homogeneous set of agents as they are equipped with different sets of skills. This means that an agent is not able to fulfill every job that is assigned to it, because the job may require skills that are not provided by the agent. The second aspect is the local view of the agents on the system. They are only aware of themselves and those they are directly connected to. This is an important property as this allows high scalability of the system.

3.2 Scenario Description

We deal with a job/task allocation domain to test our approaches. Therefore, the environment contains a set of jobs (cf. Definition 3.3). These jobs are generated dynamically during simulation. Algorithm 3.1 describes the job generation and the allocation to randomly chosen agents.

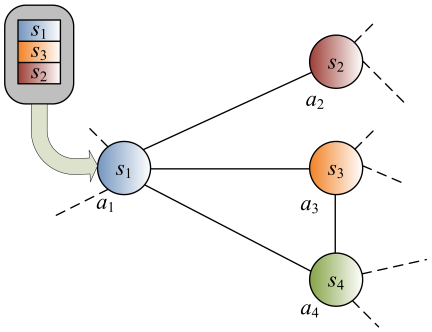
Algorithm 3.1 Job Generation and Processing

Input: set of agents $\mathcal{A}$
skill set $\mathcal{S}$
job factor k
the minimum/maximum number of tasks a job consists of $t_{\min}$ and $t_{\max}$
the minimum/maximum payoff per task $q_{\min}$ and $q_{\max}$

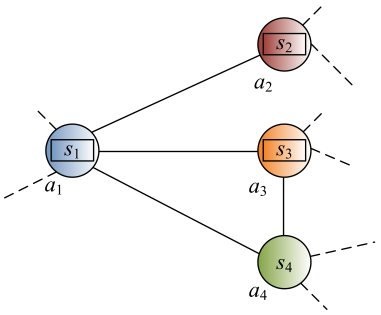
1: **procedure** JOBGENERATIONANDPROCESSING($\mathcal{A}$, $\mathcal{S}$, k, $t_{\min}$, $t_{\max}$, $q_{\min}$, $q_{\max}$)
2: **for** $k' = 1$ to $k \cdot |\mathcal{A}|$ **do**
3: $n \leftarrow \mathcal{U}[t_{\min}, t_{\max}]$ /* UNIFORMLY AT RANDOM */
4: $j \leftarrow \emptyset$ /* INITIALIZE EMPTY JOB */
5: **for** $i = 1$ to n **do** /* GENERATE THE TASKS OF THE JOB */
6: $s_t \leftarrow \mathcal{U}[1, |\mathcal{S}|]$ /* UNIFORMLY AT RANDOM */
7: $q_t \leftarrow \mathcal{U}[q_{\min}, q_{\max}]$ /* UNIFORMLY AT RANDOM */
8: $t \leftarrow (s_t, q_t)$ /* INITIALIZE THE TASK */
9: $j \leftarrow j \cup \{t\}$ /* ADD THE TASK TO THE JOB */
10: **end for**
11: select random agent a with uniform distribution
12: **if** JOBPROCESSING(j,a) **then** /* SEE ALGORITHM 3.2 */
13: $\mathsf{profit}(a) \leftarrow \mathsf{profit}(a) + \mathsf{profit}(j)$
14: charge all helpers /* COST-PRODUCING COOPERATION */
15: **end if**
16: **end for**
17: **end procedure**

In our scenario, in each step $k \cdot |\mathcal{A}|$ jobs are generated and assigned to randomly chosen agents with uniform distribution. The parameter k is called job factor. It is an exogenous parameter of the system. This leads to an assignment of on average k jobs per agent, which specifies the number of generated jobs and, thus, determines the number of interactions between agents. The jobs are dynamically generated and separately assigned to the agents and processed by the agents. This leads to one fundamental property of our system. The agents are not able to reason about the whole job set and to select the most beneficial one. We decided to do this because we concentrate on the cooperation aspect and not on the aspect of most efficient task allocations as it is done in similar models (de Weerdt and Zhang, 2008; de Weerdt et al., 2007).

If an agent is not able to fulfill all tasks of a job, it searches for cooperation partners (see Algorithm 3.2). This process is illustrated in Figure 3.1. Here, a job consisting of three tasks is assigned to agent a_1 (see Figure 3.1a). The skills s_1, s_2 and s_3 are required for fulfilling tasks t_1, t_2, and t_3. Besides the shown skills in the figure, the colors also represent the specific skills. As agent a_1 only offers skill s_1, the agent allocates two tasks of the job to its neighbors a_2 and a_3, which both have accepted to help agent a_1 for the fulfillment of jobs (see Figure 3.1b). If a_2 or a_3 has declined the request or if the job contains a task with an unprovided skill in the vicinity of agent a_1, then no task would have been allocated at all and the job would have been



(a) Job is assigned to agent a_1.



(b) a_1 has found cooperative neighbors and has allocated the tasks.

Figure 3.1: Some small example illustrating the concerned scenario.

discarded. The whole process of processing a job is described in Algorithm 3.2.

To model that cooperating agents spend their resources for fulfilling jobs they are not responsible for, we charge them for their cooperation. Therefore, we have the exogenous parameter $c \in \mathbb{R}^-$ which is a cost factor. Whenever agent a cooperates with agent b by performing a task t of a job j assigned to b, a's profit is reduced by adding $c \cdot q_t$ to its profit. q_t is the payoff for fulfilling task t. The profit function is formally defined in the following definition:

Definition 3.6 (Profit Function): *Let $\mathcal{J}_C(a)$ be the set of completed jobs that have been allocated to agent a. Let $\mathcal{T}_\mathcal{H}(a)$ be the set of tasks that agent a has processed which belong to completed jobs that have been allocated to agent a's neighbors. Then, the* profit function $\mathsf{profit} : \mathcal{A} \to \mathbb{R}$ *is defined as:*

$$\mathsf{profit}(a) := \sum_{j \in \mathcal{J}_C(a)} \mathsf{payoff}(j) + \sum_{t \in \mathcal{T}_\mathcal{H}(a)} c \cdot q_t$$

The charging of the helping agents is done in line 14 of Algorithm 3.1. In our approach, it is possible that one agent helps a second agent by processing more than one task of the foreign job. The agent then is charged more than once as it also uses more of its resources. The whole algorithm that also contains the proposed adaptation part is given in Algorithm 3.3.

After all jobs are generated and processed by the agents, the adaptation starts. Each agent a calculates the so-called elite set $\mathcal{E}_a$ of the $\varepsilon \in \mathbb{N}$ best performing agents of its neighborhood. Note, that the parameter ε has to be chosen carefully with respect to the neighborhood size. Suppose we have $\varepsilon \leq |\mathcal{N}_a|$, then agent a will always be part of the elite set and, thus, will not perform the adaptation step. Performance is measured as the profit achieved during one simulation step, which is one execution of the algorithm JOBGENERATIONANDPROCESSING.

If the agent is not in set of the ε best performing agents and has neighbors that are not willing to cooperate with it, the agent is said to be *unsatisfied* and adapts itself. Note that the threshold

Algorithm 3.2 Job Processing

Input: job j
agent a
Output: true if job can be processed, false otherwise

```
1:  procedure JOBPROCESSING(j, a)
2:      for all t = (s_t, p_t) ∈ j do
3:          if s_t ∈ S_a then
4:              processor(t) ← a
5:          else
6:              N_a(s_t) ← {b ∈ N_a | s_t ∈ S_b}
7:              if N_a(s_t) = ∅ then
8:                  mark j as uncompleted
9:                  return false
10:             end if
11:             H ← {b ∈ N_a(s_t) | b is willing to cooperate with a}
12:             if H = ∅ then
13:                 mark j as uncompleted
14:                 return false
15:             end if
16:             h ← randomly select an agent out of H
17:             processor(t) ← h
18:         end if
19:     end for
20:     for all t = (s_t, p_t) ∈ j do
21:         allocate t at agent processor(t)
22:     end for
23:     return true
24: end procedure
```

values of the neighbors are properties that are not observable. Therefore, the agents cannot compute the set of agents that will not cooperate with them. However, an agent can record the agents that have declined its helping requests and it can approximate the set of agents that are unwilling to cooperate with it. The adaptation step consists of three parts: *ideal selection*, *adaptation to the selected ideals* and *social networking*. These three parts are described in more detail in the remainder of this section.

3.2.1 Selection Strategies for the Ideal Set

Each agent a that is unsatisfied selects a local set of ideal agents $\mathcal{I}_a \subset \mathcal{N}_a$ from its neighborhood to adapt to their values. The cardinality $|\mathcal{I}_a|$ is an exogenous parameter which has to be chosen

Algorithm 3.3 Simulation

Input: number of agents $|\mathcal{A}|$
number of skills $s_{\max}$
job factor k
minimum/maximum number of tasks a job consists of $t_{\min}$ and $t_{\max}$
minimum/maximum payoff per task $q_{\min}$ and $q_{\max}$
number of elite agents ε
selection strategy for ideal set
adaptation strategy
probability $\mathsf{Pr}_{\mathcal{N}}$ of executing social networking

1: **procedure** SIMULATION
2: Initialize $|\mathcal{A}|$ agents and neighborhoods randomly
3: **loop**
4: JOBGENERATIONANDPROCESSING($\mathcal{A}$, $\mathcal{S}$, k, $t_{\min}$, $t_{\max}$, $q_{\min}$, $q_{\max}$)
5: **for all** agents $a \in \mathcal{A}$ **do**
6: $\mathcal{E}_a \leftarrow \varepsilon$ best agents of $\mathcal{N}_a \cup \{a\}$
7: **if** $a \notin \mathcal{E}_a \wedge \exists b \in \mathcal{N}_a : (b, a) \notin \mathcal{C}$ **then**
8: Select $\mathcal{I}_a \subseteq \mathcal{N}_a$ /* IDEAL SELECTION */
9: Adapt to $\mathcal{I}_a$ /* ADAPTATION */
10: with probability $\mathsf{Pr}_{\mathcal{N}}$: replace r uncooperative neighbors by r randomly chosen agents /* SOCIAL NETWORKING */
11: **end if**
12: **end for**
13: **end loop**
14: **end procedure**

carefully as the size of the neighborhoods has to be taken into account. However, the selection can be done using different strategies. We will focus on the following strategies:

best-selection Select the best performing agents of the neighborhood, such that every neighbor that does not belong to the ideal set has a profit that is less or equals to the neighbors of the ideal set. Formally:

$$\forall a^* \in \mathcal{I}_a \, \forall b \in \mathcal{N}_a \backslash \mathcal{I}_a : \mathsf{profit}(a^*) \geq \mathsf{profit}(b)$$

The idea behind this selection strategy is, that agents are believed to gain much profit because of very good ratings for the propositions. Agents that have quite high profit are those agents that gain much cooperation as many jobs can be fulfilled that are assigned to those agents. Another reason for the high profit may be, that they probably did not received much negative reward as a punishment for cooperation. Therefore, these are selected as role models as the agents want to imitate them to achieve more profit in future time steps.

worst-selection Select the worst performing neighbors, such that all neighbors that are not in the ideal set have greater or equal profit. Formally:

$$\forall a^* \in \mathcal{I}_a \; \forall b \in \mathcal{N}_a \backslash \mathcal{I}_a : \mathsf{profit}(a^*) \leq \mathsf{profit}(b)$$

This strategy is meant as follows. By imitating the worst performing agents, the agent believes that it raises the probability of receiving help from these agents. The agents with low profit are those that gain less cooperation but are willing to cooperate with others. Therefore, they have been punished by the system and their profit is very low or even negative. If the differences between the agent and the worst performing neighbors decreases, then the probability of receiving their help will probably increase.

random-selection Randomly select some neighbors as ideals. In contrast to the other two strategies, this strategy does not require any knowledge about the performance of the neighboring agents.

The number of ideals $|\mathcal{I}|$ is an exogenous parameter for the simulation and different settings for this parameter will be examined in Chapter 5. Note that this parameter is equal for all agents. As we have proposed the main strategies of selecting the ideal agents, we now want to describe the adaptation step and the adaptation strength strategies.

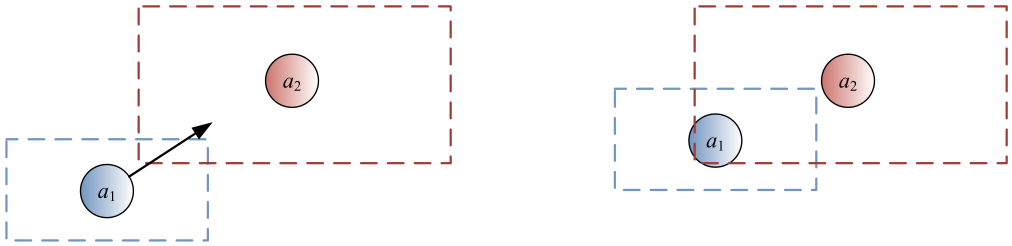
3.2.2 Adaptation Strategies

In the adaptation step the agent a adapts its proposition values $\mathcal{V}_a$ to the ratings of its ideal sets $\mathcal{I}_a$. The rule for this adaptation is the following:

$$\mathcal{V}_a \leftarrow \mathcal{V}_a + \eta \cdot \frac{1}{|\mathcal{I}_a|} \Big(\Sigma_{a^* \in \mathcal{I}_a} (\mathcal{V}_{a^*} - \mathcal{V}_a) \Big) \tag{3.2}$$

This formula contains an important, exogenous parameter $\eta \in [0, 1] \subset \mathbb{R}$, which specifies the adaptation strength. The idea behind this step is, that the ideal agents are believed to be more successful due to better values for the propositions. Therefore, the agent wants to change its values to be more like its ideals. One aspect of this parameter is the value. The value specifies the strength of the adaptation where 0 means no adaptation at all and 1 pure copying. The best setting for this parameter is analyzed in Chapter 5 where experimental results are presented. To illustrate the idea behind moving the value-vector let us consider the following example in Figure 3.2.

In both figures, the agents are represented in the hypercube defined by the dimensions of the value vectors, i.e. the number of propositions. To ease the illustration we only consider two propositions in this example. The center of the circles represents the exact position of the value-vectors and the dashed rectangles show the tolerance area defined by the threshold vectors. Figure 3.2a visualizes the given situation where agent a_1 wants to adapt its values to the values of a_2. Figure 3.2b shows the result of the adaptation. Now, agent a_2 will cooperate with agent



(a) Situation before a_1 adapts its values to agent a_2.

(b) Situation after the adaptation step.

Figure 3.2: Small example illustrating the idea of the adaptation step.

a_1 because the center of a_1 is within the tolerance area of a_2. This is exactly what agent a_1 intended to achieve by moving the vector and, thus, imitating the values of a_2. Note, however, that the tolerance area is invisible to agent a_1.

One can think of several strategies for the adaptation part. The adaptation is highly influenced by the adaptation strength η and, thus, the selection of this parameter can be done in several ways. It can be initially set and changed by simulated annealing (Kirkpatrick, 1984) in every step by the agents. Another possibility would be to mutate the parameter with some probabilities, as it is done in evolutionary computing (Eiben and Smith, 2003). As several strategies for adaptation exist, we will now describe three strategies that will be examined in Chapter 5. These strategies are:

1. Fixed $\eta = 0.5$ for all agents
2. Randomly choose η for every agent in the initialization phase, then fixed
3. Randomly choose η for every agent in every simulation step

The first strategy is very simple as for all agents the adaptation strength is equal. The other two strategies are more interesting. Both deal with randomized setting of the parameter with uniform distribution. The difference is that strategy 2 gives every agent a different value but after the initialization phase, the parameter stays constant. The third strategy also resets the value after one simulation step.

3.2.3 Social Networking

The last part of the considered approach is the *social networking* which is executed with probability $\mathsf{Pr}_{\mathcal{N}}$. The agent chooses r agents from its neighborhood, that have not cooperated with it, and replaces them by r randomly chosen agents from the population. If there are less than r agents, which have been uncooperative, all of them are replaced. Note, that the whole pop-

ulation is invisible for the agent but there are approaches that deal with this problem. In an anonymous Peer-to-Peer System nodes can randomly choose another node out of the whole set without knowing every other node (Vishnumurthy and Paul, 2006, 2007). Therefore, we concentrate on the effects of the process of social networking and ignore techniques how the new agents can be selected if not the whole population is known to the agents. By replacing the agents the situation could not worsen but only improve for the agent. This is due to the fact that all agents, which are uncooperative are replaced by randomly chosen agents, with a non-zero probability of selecting potentially cooperative agents. As stated above, only those agents will help this agent, if their values for the propositions and the thresholds fit together.

If the social networking is executed with a non-zero probability, the neighborhoods change over time. On the one hand, this means that agents lose neighbors because they are exchanged and on the other hand, the neighborhoods of newly selected neighbors can grow. This is due to the fact that agents are not allowed to reject neighborhood requests. Therefore, we introduce a fixed maximum number of neighbors allowed per agent ($\mathcal{N}_{\max}$). If this number is reached, then a link to a randomly chosen neighbor is deleted from the neighborhood. In our experiments, we always set a maximum number of allowed links per agent to prevent agents from being connected to nearly every other agent. This would give the agents the possibility to get knowledge about nearly every agent in the system, which would be some kind of global knowledge.

3.2.4 Overview on System's Parameter

The following table briefly recalls the parameters of the system:

3.3 Conclusion

In this chapter, we formally presented the considered approach as well as the benchmark scenario of job/task allocation. With the help of the neighborhood definition based on a network structure we achieved the goal of having only a local view on the system. Each agent can only sense and interact with agents that are directly connected to it. The property of having cooperation decisions that are based on a similarity measure between agents is modeled with the help of rating vectors for a system-wide set of propositions. Adaptivity is also given as the agents can adapt to a set of neighbors with the intension of improving their performance. The proposed approach is very simple as the mechanism only needs the values of the other agent to decide, if an agent will cooperate with this agent. Also the adaptation part does not need any further information. As the cooperation relation which specifies which agent will cooperate with another agent is in general not symmetric, the proposed multiagent system works with directed cooperation decisions. Through the social networking step—exchange of neighbors with some probability if the agent is unsatisfied with the current situation—the network is changed and, thus, the presented system has a dynamic network. The last property that was identified in Sec-

Table 3.1: Parameters of the System.

Parameter	Meaning
$\lvert\mathcal{A}\rvert$	population size
$\lvert\mathcal{S}\rvert$	number of skills in the system
$\lvert\mathcal{S}_a\rvert$	number of skills per agent
$\mathcal{N}_{\max}$	maximum number of neighbors allowed per agent
m	number of propositions
$v_{\max}$	maximum rating for a proposition
$\Theta_{\max}$	maximum tolerance of an agent
$q_{\min}, q_{\max}$	minimum / maximum payoff for task fulfillment
$t_{\min}, t_{\max}$	minimum/maximum number of tasks a job consists of
k	job factor, that determines the number of interactions
c	cost-factor for charging helpers
η	adaptation strength
ε	number of elite agents
$\lvert\mathcal{I}\rvert$	number of ideal agents
$\mathsf{Pr}_{\mathcal{N}}$	probability of executing social networking
r	number of replaced agents in social networking

tion 1.1 is emergent cooperation. Considering only the formal model, one cannot decide if the proposed approach has this property. However, there are no hard-coded cooperation rules. If high level of cooperation are reached, then the system shows emergent cooperation. This is one aspect that is analyzed in the next two chapters.

4 Formal Analysis

In this chapter, we formally analyze the considered approach. First, we discuss the probability of having all skills in the vicinity of an agent with the restriction that an agent only has a single skill. Second, we analyze the convergence behavior of the value propagation in small networks. And third, an analysis of the computational complexity is presented.

The main results of this chapter are as follows:

- We show that the probability of having all skills in the vicinity of an agent can be computed with the help of Stirling numbers of the second kind if each agent is equipped with a single skill only.
- It is shown show that the probability is strongly connected to the number of skills in the system and to the neighborhood size.
- We show that for a large number of skills in the system an unrealistic large average number of neighbors is needed to get a high probability of having all skills in the agents' vicinity.
- We prove the convergence to mutual cooperative behavior for the two-agent case.
- An example is presented where mutual cooperation will not be able to emerge for the three-agent case and which strategies may prevent this behavior.
- We show that the proposed approach has quadratic complexity in the number of agents in the worst case and linear complexity in the average case from the simulative point of view (global view).
- It is shown that the proposed approach has linear complexity in the number of neighbors from the agents' viewpoint (local view).

4.1 Skills in the Agents' Vicinity

In this section, we analyze the probability of having all skills in the vicinity of an agent given the number of neighbors $|\mathcal{N}_a|$ and the number of possible skills in the system $|\mathcal{S}|$. First, we give some notations and then we will start the analysis.

For the analysis, we need to define the set of agents under consideration and the number of skills in the system. Therefore, we use the following notation in this section:

- the set of agents in agent a's vicinity is denoted by $\mathcal{A}_a$, where

$$\mathcal{A}_a := \mathcal{N}_a \cup \{a\}, \text{ with } n := |\mathcal{A}_a| \tag{4.1}$$

- the set of skills in the system

$$\mathcal{S}, \text{ with } s_{\max} := |\mathcal{S}| \tag{4.2}$$

Let us assume that every agent only has a single skill, i.e. $|\mathcal{S}_a| = 1$ for every agent $a \in \mathcal{A}$, and that the number of possible skills $s_{\max}$ is not greater than the number of agents in a's vicinity, i.e. $|\mathcal{S}| = s_{\max} \leq |\mathcal{A}_a|$. Then, we can define a function skills that maps the set of agents under consideration to the set of all possible skills:

$$\mathsf{skills} : \mathcal{A}_a \to \mathcal{S} \tag{4.3}$$

It is easy to see that this function by definition is total as no agent may have none of the skills. There exist $|\mathcal{S}|^{|\mathcal{A}_a|}$ many (total) functions with the given signature, if $|\mathcal{S}| \leq |\mathcal{A}_a|$. Nevertheless, we want to know the number of surjections as the following property should hold:

$$\bigcup_{b \in \mathcal{A}_a} \mathsf{skills}(b) = \mathcal{S} \tag{4.4}$$

So, we are searching for the number of surjective mappings from a set with n elements to a set with $s_{\max}$ elements with $s_{\max} \leq n$. This question is closely related to the question of how many ways exist to partition a set of n elements into m non-empty subsets with $m \leq n$. The answer is provided by the *Stirling numbers of the second kind*. The connection between the Stirling numbers of the second kind and surjections has been investigated in literature (see, e.g. (Brualdi and Bogart, 1977; Lloyd and Ledermann, 1985; Tucker, 2007)). We will use these results in this section.

Lemma 4.1 (Stirling Number of the Second Kind (Abramowitz and Stegun, 1972)): *The number of ways of partitioning a set of n elements into m non-empty subsets with $m \leq n$ is given by*

$$S(n,m) := \frac{1}{m!} \cdot \sum_{i=0}^{m} (-1)^i \cdot \binom{m}{i} \cdot (m-i)^n \tag{4.5}$$

The sequence of numbers $S(n,m)$ is known as the Stirling numbers of the second kind.

The problem of partitioning a set of n elements into m non-empty subsets does not consider the different permutations of the subsets. However, in the context of functions we should consider the permutations. This leads us to the following lemma.

Lemma 4.2: *The number of surjective, total functions mapping from a set of n elements to a set of m elements is*

$$\mathrm{Sur}(n,m) := \sum_{i=0}^{m} (-1)^i \cdot \binom{m}{i} \cdot (m-i)^n. \tag{4.6}$$

Proof: Note, that the following holds:

$$\mathrm{Sur}(n,m) = m! \cdot S(n,m) \tag{4.7}$$

The number of surjections can be computed with the help of the Stirling numbers of the second kind. However, if we count the ways of partitioning a set of n elements into m non-empty subsets we can call these subsets equivalence classes. For partitioning, the order of equivalence classes does not count. In the context of surjections the order is relevant which leads to the factor $m!$ as there are $m!$ permutations of equivalence classes. □

Thus, the probability of having all skills in the vicinity of agent a having $n-1$ neighbors is

$$\mathsf{Pr}\Big(\bigcup_{x\in\mathcal{A}_a} \text{skills}(x) = \mathcal{S}\Big) = \frac{\text{Sur}(n, s_{\max})}{{s_{\max}}^n} \tag{4.8}$$

as $\text{Sur}(n, s_{\max})$ many surjections exist under all ${s_{\max}}^n$ functions mapping from the agents under consideration to the skill sets with $s_{\max} \leq n$.

For specific numbers of skills $|\mathcal{S}| \in \{3, 5, 7, 10, 12, 15\}$ and specific neighborhood sizes $|\mathcal{N}_a| \in \{0, 1, 2, \ldots, 35\}$ we have calculated the probability of having all skills in the vicinity of a specific agent a. Note that a neighborhood size of 5 results in $|\mathcal{A}_a| = 6$. Figure 4.1 illustrates the results.

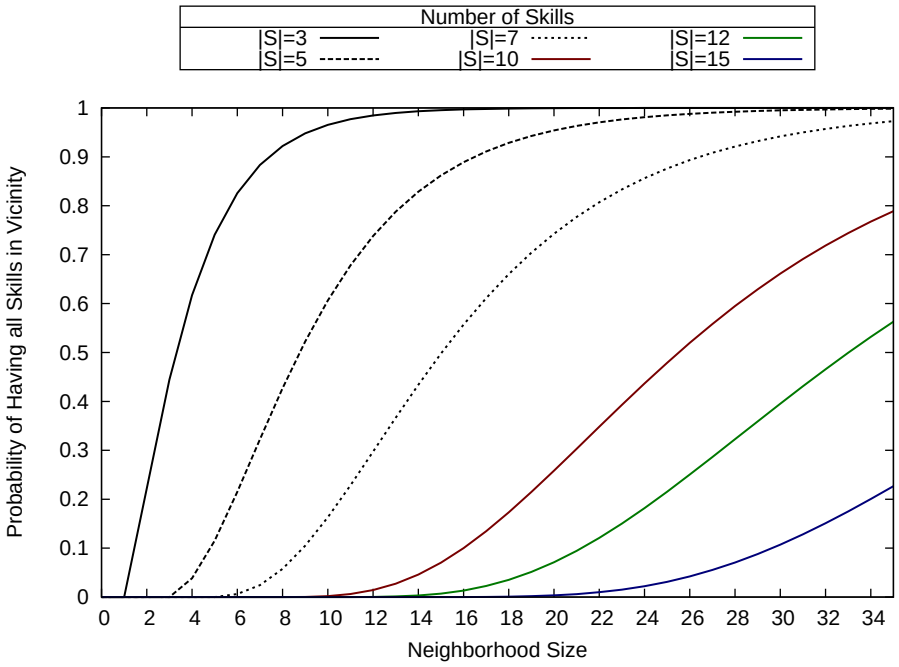


Figure 4.1: Comparison of probabilities of having all skills in the vicinity of agent a given its neighborhood size and the number of skills in the system $|\mathcal{N}_a|$.

As expected, the probability of having all skills in agent a's vicinity gets lower if the number of skills in the system gets greater with constant neighborhood size. With 20 neighbors—leading

Table 4.1: Number of required neighbors for specific skill set sizes to get a probability of 99% of having all skills in an agent's vicinity (left-hand side). The right-hand side table shows the required number of neighbors for a relaxed probability of 95%.

$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{N}_a\rvert_{\text{needed}}$
3	14
5	27
7	42
10	65
12	81
15	105
20	148
30	236

$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{N}_a\rvert_{\text{needed}}$
3	11
5	20
7	32
10	50
12	62
15	82
20	116
30	187

to 21 agents in vicinity—and five skills in the system the probability is about 95% but only about 25% if there are ten skills in the system. The question arises, how many neighbors are needed such that (nearly) all skills are in the vicinity of a specific agent. That is, when does the probability gets close to 100%? This question is hard to answer if the number of skills in the system is arbitrary. Nevertheless, it can be answered if we know the number of skills that are possible. Let us assume that *close to one* means greater than 99%. Then we can compute the required number of neighbors if every agent is endowed with a single skill and $|\mathcal{S}|$ skills are in the system. The left-hand side of Table 4.1 gives the number of neighbors $|\mathcal{N}_a|_{\text{needed}}$ for skill set sizes $|\mathcal{S}| \in \{3, 5, 7, 10, 12, 15, 20, 30\}$. As can be seen, the number of required neighbors is quite high if we want to have a probability of 99%. If we relax the requirement to have a probability of 95% the number of neighbors decreases significantly, which is presented on the right-hand side of Table 4.1.

In the experimental analysis provided in Chapter 5, we will see that this high numbers of neighbors are not needed due to the network adaptation process, i.e. the *social networking*.

4.2 Value Propagation in Small Networks

In this section, we analyze the convergence behavior of our adaptation mechanism. We want to get insights into the mechanism of adaptation and want to take a closer look on very special scenarios. To ease the analysis only static interaction networks are considered here. Therefore, the influences of the dynamics of the network can be neglected. We only deal with very small agent sets in order to be able to analyze the propagation of the values through the network. We claim that these small scenarios can be found in greater networks in the form of sub-networks. Therefore, the results from this section can also be transferred to large networks composed of thousands of agents.

It is shown that for systems composed of only two agents a convergence to mutually cooperative behavior can be expected. However, we present a system of three agents where convergence to cooperative behavior cannot be observed. We also present strategies to avoid this behavior. Parts of this section have been previously published in (Eberling and Kleine Büning, 2010a, 2011).

4.2.1 The Simplest Scenario

Let us start with a very simple scenario consisting of two agents. The system is illustrated in Figure 4.2. Here the agents have different skills as the colors suggest. We assume that the combination of both skills is sufficient for all jobs, i.e. there is no other skill required to fulfill the job. If there would be jobs, which require a skill that these two agents cannot provide, then these jobs will not have any influence on this small system's behavior concerning the adaptation, as these jobs will never be fulfilled. Thus, the payoff of these jobs will not influence the agents' profit and that is why they do not influence the adaptation step. Therefore, we neglect such jobs.

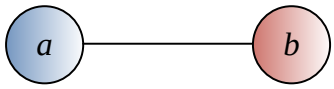


Figure 4.2: Simple MAS composed of two agents.

We denote with $\mathsf{profit}(a)$ the profit that an agent a earned in one simulation step as it is defined in Definition 3.6. In the scenario with two agents, the job phase can produce the following three different profit scenarios:

1. $\mathsf{profit}(a) = \mathsf{profit}(b)$
2. $\mathsf{profit}(a) > \mathsf{profit}(b)$
3. $\mathsf{profit}(a) < \mathsf{profit}(b)$

Case 1 is very simple since no adaptation takes place, if both agents have the same profit. As

case 2 and 3 are symmetric, we concentrate on case 2 in the remainder of this section. In case 2 agent b always adapts its values to agent a. For the value adaptation we take a slightly different version of the adaptation rule provided in Equation 3.2 as we only have one ideal agent and not a whole ideal set. Therefore, Equation 4.9 gives the new value-vector in step $t+1$ of agent b after it adapts its vector to agent a in simulation step t.

$$\mathcal{V}_b^{t+1} = \mathcal{V}_b^t \cdot (1-\eta) + \eta \cdot \mathcal{V}_a^t \tag{4.9}$$

Since $\mathsf{profit}(a) > \mathsf{profit}(b)$, agent a never adapts and, thus, its value vector never changes. Therefore, it always has the vector it received in the initialization. This is reflected in the following equation, which holds for the considered case:

$$\forall t : \mathcal{V}_a^t = \mathcal{V}_a^0 \tag{4.10}$$

As we are interested in the value propagation, we have to take a look at the value-vectors in every simulation step. For agent a this is trivial as it does not change its value vector at all. For agent b we now want to compute the changes in every simulation step and provide a formula to calculate the value vector for agent b for an arbitrary simulation step. This leads to the following development:

$$\begin{aligned}
\mathcal{V}_b^1 &= \mathcal{V}_b^0 \cdot (1-\eta) + \eta \cdot \mathcal{V}_a^0 \\
\mathcal{V}_b^2 &= \mathcal{V}_b^1 \cdot (1-\eta) + \eta \cdot \mathcal{V}_a^0 \\
&= \mathcal{V}_b^0 \cdot (1-\eta)^2 + \eta(1-\eta) \cdot \mathcal{V}_a^0 + \eta \cdot \mathcal{V}_a^0 \\
\mathcal{V}_b^3 &= \mathcal{V}_b^2 \cdot (1-\eta) + \eta \cdot \mathcal{V}_a^0 \\
&= \mathcal{V}_b^0 \cdot (1-\eta)^3 + \eta(1-\eta)^2 \cdot \mathcal{V}_a^0 + \eta(1-\eta) \cdot \mathcal{V}_a^0 + \eta \cdot \mathcal{V}_a^0 \\
&\vdots \\
\mathcal{V}_b^{t+1} &= \mathcal{V}_b^0 \cdot (1-\eta)^{t+1} + \eta \cdot \mathcal{V}_a^0 \cdot \sum_{i=0}^{t} (1-\eta)^i
\end{aligned}$$

Using the last equation, we now can compute the values of b for every simulation step. To show that the system composed of two agents converges to mutual cooperation we need to show that the distance between the value vectors never increases as this would result in the contrary case. This is stated in the following lemma:

Lemma 4.3: *Let* $\mathsf{dist}(a, b, t)$ *be the distance of the value vectors of two agents* a *and* b *in step* t *with:*

$$\mathsf{dist}(a, b, t) = \left|\mathcal{V}_a^t - \mathcal{V}_b^t\right|$$

In a scenario with just two agents, the distance never increases, i.e.

$$\forall t : \mathsf{dist}(a, b, t+1) \leq \mathsf{dist}(a, b, t)$$

Proof: To ease the proof let us consider a single proposition, only. The proof can easily be extended to m propositions in general. For better readability $\mathcal{V}$ denotes the value for this single proposition instead of a one-dimensional vector. By definition the distance between the values for the proposition is

$$\mathsf{dist}(a, b, t+1) = \left|\mathcal{V}_a^{t+1} - \mathcal{V}_b^{t+1}\right|$$

As stated above we have to consider the three cases for the profit distribution.

Case 1: In the current simulation step the agents gained the same profit, i.e.: $\mathsf{profit}(a) = \mathsf{profit}(b)$. As no agent has a higher profit, both are satisfied and no adaptation takes place. Therefore, it holds that

$$\mathsf{dist}(a, b, t+1) \leq \mathsf{dist}(a, b, t).$$

Case 2: Agent a gained more profit in the current simulation step, i.e.: $\mathsf{profit}(a) > \mathsf{profit}(b)$. Then agent a is satisfied and agent b is unsatisfied. Therefore, we have $\mathcal{V}_a^{t+1} = \mathcal{V}_a^t$ as agent a does not adapt its values and $\mathcal{V}_b^{t+1} = \mathcal{V}_b^t + \eta(\mathcal{V}_a^t - \mathcal{V}_b^t)$ as agent b adapts to agent a. Therefore, we have

$$\mathsf{dist}(a, b, t+1) = \left|\mathcal{V}_a^t - \mathcal{V}_b^t - \eta(\mathcal{V}_a^t - \mathcal{V}_b^t)\right|.$$

We have to distinguish the cases for the current proposition values for both agents. Therefore, we have the following two cases:

Case 2.a: $\mathcal{V}_a^t \geq \mathcal{V}_b^t$:

$$\begin{aligned}
\mathsf{dist}(a, b, t+1) &= \left|\mathcal{V}_a^t - \mathcal{V}_b^t - \eta(\mathcal{V}_a^t - \mathcal{V}_b^t)\right| \\
&= \mathcal{V}_a^t - \mathcal{V}_b^t - \eta(\mathcal{V}_a^t - \mathcal{V}_b^t) \\
&= \left|\mathcal{V}_a^t - \mathcal{V}_b^t\right| - \eta \cdot \left|\mathcal{V}_a^t - \mathcal{V}_b^t\right| \\
&= \mathsf{dist}(a, b, t) - \eta \cdot \mathsf{dist}(a, b, t)
\end{aligned}$$

Case 2.b: $\mathcal{V}_a^t < \mathcal{V}_b^t$:

$$\begin{aligned}
\mathsf{dist}(a, b, t+1) &= \left|\mathcal{V}_a^t - \mathcal{V}_b^t - \eta(\mathcal{V}_a^t - \mathcal{V}_b^t)\right| \\
&= -\mathcal{V}_a^t + \mathcal{V}_b^t + \eta(\mathcal{V}_a^t - \mathcal{V}_b^t) \\
&= \left|\mathcal{V}_a^t - \mathcal{V}_b^t\right| - \eta \cdot \left|\mathcal{V}_a^t - \mathcal{V}_b^t\right| \\
&= \mathsf{dist}(a, b, t) - \eta \cdot \mathsf{dist}(a, b, t)
\end{aligned}$$

The values for the propositions are always greater or equal to zero. The same holds for the adaptation strength η. Therefore, from both cases it follows that $\mathsf{dist}(a, b, t+1) \leq \mathsf{dist}(a, b, t)$.

Case 3: $\mathsf{profit}(a) < \mathsf{profit}(b)$. This case is symmetric to case 2 which means that it also leads to the result that $\mathsf{dist}(a, b, t+1) \leq \mathsf{dist}(a, b, t)$.

From all three cases it follows that $\mathsf{dist}(a,b,t+1) \leq \mathsf{dist}(a,b,t)$ which proves the lemma. $\square$

Up to now we know that the distances between both value vectors can never increase in this scenario. In this system, we have three different profit scenarios in each simulation step. As the payoff for the jobs is uniformly distributed, we claim that all profit sscenarios are also uniformly distributed and, therefore, the probability for each situation is exactly $\frac{1}{3}$. Therefore, in every third simulation step on average we have no adaptation, as case 1 will take place.

We now want to know how many steps are needed until both agents are willing to cooperate with each other, i.e. $a \in \mathcal{C}_b$ and $b \in \mathcal{C}_a$. Therefore, we need a sufficient number of adaptation steps. Let us assume that case 3 never occurs which means that we have two situations where agent b adapts to agent a and one simulation step without adaptation because of case 1. We can make this assumption, as case 2 and 3 are symmetric and do not influence the speed of convergence. To realize such an adaptation setting, we can assume that agent a is less tolerant, i.e. $\Theta_a < \Theta_b$. This means that agent a determines the number of steps needed, since $a \in \mathcal{C}_b$ will follow first. From the proof of Lemma 4.3 we know:

$$\begin{aligned} \mathsf{dist}(a,b,t) &= \mathsf{dist}(a,b,t-1) - \eta \cdot \mathsf{dist}(a,b,t-1) \\ &= (1-\eta) \cdot \mathsf{dist}(a,b,t-1) \end{aligned}$$

We can now conclude that:

$$\begin{aligned} \mathsf{dist}(a,b,t) &= (1-\eta) \cdot \mathsf{dist}(a,b,t-1) \\ &= (1-\eta)^2 \cdot \mathsf{dist}(a,b,t-2) \\ &\vdots \\ &= (1-\eta)^t \cdot \mathsf{dist}(a,b,0) \end{aligned}$$

Thus, we are searching for the earliest simulation step t that satisfies $(1-\eta)^t \cdot \mathsf{dist}(a,b,0) \leq \Theta_a$. Under the assumption that $\mathsf{dist}(a,b,0) \neq 0$, we obtain:

$$\begin{aligned} & (1-\eta)^t \cdot \mathsf{dist}(a,b,0) && \leq && \Theta_a \\ \Leftrightarrow \quad & (1-\eta)^t && \leq && \tfrac{\Theta_a}{\mathsf{dist}(a,b,0)} \\ \Leftrightarrow \quad & t \cdot \ln(1-\eta) && \leq && \ln\left(\tfrac{\Theta_a}{\mathsf{dist}(a,b,0)}\right) \\ \Leftrightarrow \quad & t && \overset{\eta\in(0,1)}{\geq} && \tfrac{\ln\left(\frac{\Theta_a}{\mathsf{dist}(a,b,0)}\right)}{\ln(1-\eta)} \end{aligned}$$

This means that in step

$$t' = \left\lceil \frac{\ln(\Theta_a) - \ln(\mathsf{dist}(a,b,0))}{\ln(1-\eta)} \right\rceil \tag{4.11}$$

it will hold that $a \in \mathcal{C}_b$ and $b \in \mathcal{C}_a$. Note that we have assumed, that the adaptation strength η is taken from the open interval $(0,1)$. $\eta = 0$ would result in no adaptation, but this is what we want to examine. Thus, this value is omitted. Additionally, $\eta = 1.0$ is omitted, as this would mean that after a single step both agents would cooperate in the following as both will have the same value-vectors after the adaptation. This easy case is neglected here.

Equation 4.11 only considers that in every simulation step we have case 2. As case 3 is symmetric to case 2 it would lead to similar time demands but from the other agent's perspective. Again, case 1 can also occur and it will be the case in every third simulation step on average. Therefore, a more precise formulation for the expected time demands in this setting would be

$$\frac{3}{2} \cdot \max \left\{ \left\lceil \frac{\ln(\Theta_a) - \ln(\mathsf{dist}(a,b,0))}{\ln(1-\eta)} \right\rceil, \left\lceil \frac{\ln(\Theta_b) - \ln(\mathsf{dist}(a,b,0))}{\ln(1-\eta)} \right\rceil \right\} \tag{4.12}$$

Equation 4.12 considers all three situations that can occur for the profit distribution and takes into account the assumed uniform distribution of the jobs' payoff and jobs' requirements. Note that case 1 slows down the development as no adaptation takes place, if both agents reach the same profit. However, eventually it will hold that case 2 or 3 will again occur and the process is continued.

Although we have seen good results in previous work (cp. (Eberling, 2009; Eberling and Kleine Büning, 2010b)), we cannot ensure convergence in every setting as stated in Lemma 4.4:

Lemma 4.4: *The adaptation cannot ensure convergence. There are settings in which the system will fail.*

Proof: Section 4.2.2 gives an example where the adaptation does not converge. □

4.2.2 A Simple Scenario Without Convergence

In this section, we present a system where convergence cannot take place. In this system we have three agents. For the agents and the interaction network we consider the following formal specification, which is explained in detail in the following.

- set of agents $\mathcal{A} = \{a, b, c\}$
- $\mathsf{IN} = (\{a, b, c\}, \{\{a, b\}, \{b, c\}\})$
- $\mathcal{S} = \{1, 2, 3, 4, 5\}$
- $t_{\min} = t_{\max} = 3$ and $q_t = 1$ for all tasks t
- $\mathcal{S}_a = \{1\}$, $\mathcal{S}_b = \{3\}$, $\mathcal{S}_c = \{5\}$
- $\mathcal{P} = \{p_1\}$

- $\mathcal{V}_a = 0, \mathcal{V}_b = 50, \mathcal{V}_c = 100$
- $\Theta_a = 2, \Theta_b = 100, \Theta_c = 2$

Again, we only consider a single proposition and interpret the values $\mathcal{V}$ as rational numbers instead of one-dimensional vectors. Figure 4.3 illustrates the structure of the interaction network.

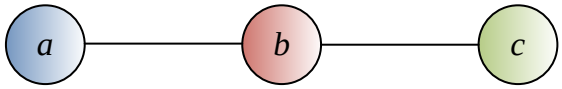


Figure 4.3: Simple MAS composed of three agents.

The system contains a skill set with five elements where the agents provide three of them. Each job contains exactly tree skills and every task leads to a payoff of 1 utility unit. The cooperation costs are therefore -0.25 utility units. Agent b is connected to the agents a and c where those agents are not linked together. Therefore, cooperation can only take place between the pairs a, b and b, c. When we look at the proposition values and thresholds one should notice that agent b is very tolerant whereas the other two agents are very intolerant. Equation 3.2 gives the adaptation rule:

$$\mathcal{V}_a^{t+1} = \mathcal{V}_a^t + \eta \cdot (\mathcal{V}_{a^*}^t - \mathcal{V}_a^t) \tag{4.13}$$

Equation 4.13 means, that agent a adapts its values by adding the weighted distance between the value of a's best performing neighbor a^* and its own value. This moves the value into the direction of the best performing agent. Clearly, $\mathcal{V}_a^0$ is the initial value given in our specification. Now, consider the following profit scenario:

$$\mathsf{profit}(a) > \mathsf{profit}(c) > \mathsf{profit}(b), \text{for odd } t \tag{4.14}$$

$$\mathsf{profit}(c) > \mathsf{profit}(a) > \mathsf{profit}(b), \text{for even } t \tag{4.15}$$

This profit scenario can be the result of the relative intolerant agents a and c (i.e. $\Theta_a = \Theta_c = 2$) and the very tolerant agent b. This can lead to an alternating adaptation of agent b to agent a in odd simulation steps and to agent c in even simulation steps. As the agents a and c only have a single neighbor, agent b, and this agent is always the worst performing one, they never adapt. Therefore, the length of the value-interval remains constant.

If we set $\eta = 0.5$ and let agent b adapt in the alternating way as described above, we obtain the values for the proposition as shown in Table 4.2. The value of agent b changes in every step and it can be observed that it does not converge to a single value but it oscillates between the values $33\frac{1}{3}$ and $66\frac{2}{3}$. For both directions it holds that in every simulation step the minimal distance is one third of the interval length, i.e. $33\frac{1}{3}$. Therefore, agent b never receives help from the other two agents. The only possibility for agent b to gain profit is the fulfillment of a job containing three times skill 3. However, this situation is very rare, if we consider the job generation mechanism as it is described in Algorithm 3.1. In our setting we have $|\mathcal{S}| = 5$ and

Table 4.2: Change of values during adaptation for 14 simulation steps.

t	$\mathcal{V}_a^t$	$\mathcal{V}_b^t$	$\mathcal{V}_c^t$
0	0	50	100
1	0	25	100
2	0	65.5	100
3	0	31.25	100
4	0	65.625	100
5	0	32.8125	100
6	0	66.4063	100
7	0	33.2031	100
8	0	66.6016	100
9	0	33.3009	100
10	0	66.6504	100
11	0	33.3252	100
12	0	66.6626	100
13	0	33.3313	100
14	0	66.6656	100

each agent has a single skill. Each job consists of three tasks. Therefore, we have a probability of 0.8% to generate a job that requires three times the same skill. However, as agent b is very tolerant it always helps the other two agents if they ask for help. That is why agent b is punished very often in contrast to the other two agents which never receive negative payoff.

Although this scenario leads to no convergence, the sequence of the $\mathcal{V}_b^t$ of agent b consists of two subsequences, each converging to a fixed value. We can identify these values for every η. Table 4.3 gives the approximated bounds for some η.

However, this construction is very unrealistic. In scenarios that have been considered in previous work (Eberling, 2009; Eberling and Kleine Büning, 2010b) this problem does not occur or at least it does not lead to significant performance losses. There, we dealt with 1000 agents and neighborhood sizes of 15 to 20 agents in a random network. Because of the results in (Eberling, 2009; Eberling and Kleine Büning, 2010b), we assume that in random networks the probability of having situations without convergence gets very close to zero.

Table 4.3: Convergence points for the subsequences for $\mathcal{V}_b^t$.

η	odd t	even t
0.0	50	50
0.1	47.303	52.572
0.2	44.442	55.554
0.3	41.176	58.824
0.4	37.5	62.5
0.5	33.333	66.666
0.6	28.571	71.429
0.7	23.077	76.923
0.8	16.666	83.333
0.9	9.091	90.909
1.0	0	100

A very strong assumption we made in this subsection is, that agent b adapts to its neighbors in an alternating way. If the agent adapts to one neighbor only, we get a similar convergence behavior as in the scenario considered in Section 4.2.1. Assume, that only the case occurs in which agent a is the overall best agent. Then, we can apply Equation 4.11 to calculate the time steps t' needed until agent a and b will mutually cooperate, if the adaptation strength is set to $\eta = 0.5$:

$$t' = \left\lceil \frac{\ln(2) - \ln(50)}{\ln(0.5)} \right\rceil = 5$$

This means that after five adaptation steps if holds that $(a, b) \in \mathcal{C}$ and $(b, a) \in \mathcal{C}$. Especially it holds that $\mathsf{dist}(a, b, t) \leq \Theta_a$ for all $t \geq 5$ which means that b will receive cooperation from agent a and this lets agent b perform better than agent c, after the fifth simulation step. Consequently, c will adapt to b and the interval between the values of agent a and agent c will diminish. If this happens then we eventually have a situation in which the agents will mutually cooperate as the distances between their values fall under their threshold values.

The question remains how this non-converging behavior may be detected and avoided. The core problem is that agent b in our example is oscillating between two ratings for the proposition. For each end-point of the oscillation the agent adapts its value to a specific other agent. Therefore, it is possible to learn from the history of adaptation steps. The agent needs to record which ideal agent was selected, what the value has been before and after the adaptation step.

If this information is available to the agent it may reason about the history of adaptation steps and if the presented situation appears, it can change its strategy and select one of the previously selected ideal agents and mark the agent as non-ideal. Thus, the agent will not adapt to that agent any longer and will get close enough with its rating to the only agent it adapts to. This will eventually result in a situation where mutual cooperation will emerge between these two agents and the ratings interval will eventually diminish.

4.3 Computational Complexity

In this section, we analyze the computational complexity of the proposed approach. As we have presented in Section 3.2, the approach basically consists of three algorithms. The core builds SIMULATION which calls JOBGENERATIONANDPROCESSING. JOBGENERATIONANDPROCESSING itself calls JOBPROCESSING. Thus, we will start with the analysis of JOBPROCESSING followed by JOBGENERATIONANDPROCESSING until we give the complete computational complexity of SIMULATION.

Most steps of Algorithm 3.2 have complexity of $\mathcal{O}(1)$ and, thus, we will only concentrate on those parts that have a different complexity. Line 6 has a complexity of $\mathcal{O}(|\mathcal{N}_a|)$ as each neighbor has to be considered in order to calculate the set of agents that offer the requested skill. The same holds for the set of potential helpers that is constructed in line 11. Thus, the overall complexity of Algorithm 3.2 JOBPROCESSING is

$$\mathcal{O}(|j| \cdot |\mathcal{N}_a|) \tag{4.16}$$

where $|j|$ is the size of the job, i.e. the number of tasks.

Algorithm 3.1 (JOBGENERATIONANDPROCESSING) contains one outer for-loop which is executed exactly $k \cdot |\mathcal{A}|$ times as this is the number of jobs generated in each simulation round. The inner for-loop is executed n times where n is the number of tasks the constructed job should consist of. Since, we always have $t_{\min} \leq n \leq t_{\max}$, we can neglect the execution of the inner for-loop in our considerations. All other steps in the algorithm have constant complexity besides the call of JOBPROCESSING. Thus, the complexity of Algorithm 3.1 is:

$$\mathcal{O}(k \cdot |\mathcal{A}| \cdot |j| \cdot |\mathcal{N}_a|) \tag{4.17}$$

To complete the analysis we now have to consider Algorithm 3.3 (SIMULATION). We neglect the initialization phase as this strongly depends on the network type. However, we can say that the initialization is at most polynomial in the number of agents. The call of algorithm JOBGENERATIONANDPROCESSING has complexity $\mathcal{O}(k \cdot |\mathcal{A}| \cdot |j| \cdot |\mathcal{N}_a|)$ as presented before. The determination of the set of elite agents and the test if the agent itself is part of the set can be done in $\mathcal{O}(|\mathcal{N}_a|)$ as only the agent itself has to be compared to the neighbors and we have to count the number of neighbors which have a greater profit than the agent itself. The

determination of having an agent which does not cooperate with the agent can obviously also be done in $\mathcal{O}(|\mathcal{N}_a|)$. The same holds for building the set of ideal agents. The change of the proposition value-vectors has complexity of $\mathcal{O}(|\mathcal{P}|)$ as each component has to be moved. The social networking step has complexity of $\mathcal{O}(r)$ as r agents have to be changed plus costs for finding new agents from the global set $|\mathcal{A}|$ which is neglected, here. Thus, the for-loop has complexity of $\mathcal{O}((3 \cdot |\mathcal{N}_a| + |\mathcal{P}| + r) \cdot |\mathcal{A}|)$. Finally, we have for the complexity of Algorithm SIMULATION:

$$\mathcal{O}((3 \cdot |\mathcal{N}_a| + |\mathcal{P}| + r) \cdot |\mathcal{A}| + k \cdot |\mathcal{A}| \cdot |j| \cdot |\mathcal{N}_a|) \tag{4.18}$$

As the neighborhoods cannot be greater than the whole population and under the assumptions that $|\mathcal{P}| \ll |\mathcal{A}|$, $|j| \ll |\mathcal{A}|$, and $k \ll |\mathcal{A}|$, we have a worst case complexity of

$$\mathcal{O}(|\mathcal{A}|^2) \tag{4.19}$$

For most realistic scenarios, the neighborhoods are much more smaller. Thus, in the normal case of $|\mathcal{N}_a| \ll |\mathcal{A}|$ we have a complexity of approximately $\Theta(|\mathcal{A}|)$ as the average case.

This analysis is from the simulative viewpoint, which is a global view. As the approach is designed to run distributed on the agents, we will now consider the computational complexity for a single agent in a single run of the simulation loop. Therefore, we only consider the job processing and the adaptation step. For the adaptation step we again get a complexity of $\mathcal{O}(3 \cdot |\mathcal{N}_a| + |\mathcal{P}| + r)$ as we had from the global viewpoint. For the analysis of the job processing we have $\mathcal{O}(|j| \cdot |\mathcal{N}_a|)$ for each job that is allocated to the agent. As on average k jobs are allocated to an agent we obtain for the computational complexity of a single agent in one simulation step:

$$\mathcal{O}(k \cdot |j| \cdot |\mathcal{N}_a| + 3 \cdot |\mathcal{N}_a| + |\mathcal{P}| + r) \tag{4.20}$$

which, again, can be simplified to

$$\mathcal{O}(|\mathcal{N}_a|) \tag{4.21}$$

under the same assumptions as before. Therefore, the proposed approach is linear in the number of neighbors when considering the computational complexity for a single agent in a single simulation step.

4.4 Conclusion

In this chapter, we gave a formal analysis of two important parts of our system. A formula was presented to predict the probability of having all possible skills in the vicinity of the agent. We proved that this formula corresponds to the number of surjections between two finite sets. We claimed that the number of required neighbors for greater skill set sizes is unrealistically large if the probability of having all skills in the vicinity of the agents should be very high, i.e. 99%. Therefore, the agents have to deal with the problem that not every possible skill is in their vicinity.

In the second part of the chapter, we presented convergence analyses for proposition value propagation. We proved for the two-agent case that the adaptation eventually results in mutual cooperation. For the three-agent case, we presented an example where no mutual cooperation can emerge. We also presented an approach to detect and prevent such settings. Nevertheless, for this the agent would have to have much knowledge about earlier adaptation steps. We claim that the property of having agents with low knowledge capabilities is violated if the agents would be able to record all important information about previous adaptation steps. We will see in the experimental analysis that in most settings these information are practically not needed, as high level of completed jobs can be achieved without this knowledge.

Finally, in the third part, we have shown that the proposed approach has quadratic complexity in the number of agents in the worst case but linear complexity in the number of agents in the average case. As this measure needs a global view, we have additionally shown that the distributed approach has linear complexity in the number of neighbors from the local agents' view.

5 Experimental Results

In this chapter, we provide experimental results concerning the influence of the most important simulation parameters. Before this, we will give an overview on the simulation setup and the base setting for the parameters. In each section, we then describe the variation of the examined parameter and analyze the influence.

In this chapter we will present that

- cooperation can emerge even if cooperation is not for free.
- the system can deal with many propositions although the number of propositions has great influence on the cooperation willingness.
- even for maximum tolerance values of half the rating space (i.e. $\Theta_a \in (0, 50]$) high levels of cooperation can be reached.
- the fixed adaptation strategy (i.e. $\eta = 0.5$ for all agents) strategy works best.
- adapting to the best neighbor works best.
- if the profit of other agents is unobservable then adapting to two randomly selected neighbors is an alternative.
- the average neighborhood size in combination with the total number of skills in the system has great influence on the system's performance and, thus, both should be selected carefully.
- a minimum number of interactions per simulation step is required in order to achieve a development to a high job completion rate.
- the algorithm is scalable and robust against different population sizes.

Parts of this chapter have been previously published in (Eberling, 2009) and (Eberling and Kleine Büning, 2010b).

5.1 Basic Experimental Setup

In the following, we present the basic parameter settings for all experiments. The experiments ran for 200 simulation steps. We have chosen this number of simulation steps as we want the system to develop quite fast and, thus, do not grand the system more time to develop. Each parameter setting was repeated 30 times and we will present the average values over these repetitions. In previous experiments (Eberling, 2009; Eberling and Kleine Büning, 2010b) we have seen that the outcomes do not significantly change with additional numbers of repetitions and,

thus, we identified that 30 repetitions are sufficient to obtain reliable outcomes of the system. The population size is set to $|\mathcal{A}| = 1000$ since we want to consider large agent sets. The interaction network IN is an Erdõs-Rényi random network following the definition provided in (Erdõs and Rényi, 1959). Erdõs-Rényi random networks are constructed as follows. For each pair of nodes add the edge between the nodes with some probability. Here, we use a connection probability of $\mathsf{Pr}_{\mathsf{con}} = 0.015$. This leads to a medium dense network with an average neighborhood size of 15 for every agent, which—from our point of view—seems to be a good size for the experimental analysis. Due to the social networking phase the neighborhoods could grow but we only allow a maximum neighborhood size of $\mathcal{N}_{\max} = 30$. If this limit is reached the agent has to cut a link to a randomly chosen agent each time a new link should be established. The growth of neighborhoods is due to the fact that a chosen agent is not allowed to reject new interaction links. This is the result of one of our requirements of having one-sided cooperation decisions (cf. Chapter 3).

In the system, there are five skills available and agents are equipped with one skill, only. The agents rate $m = 5$ propositions with values from the interval $[0, 100]$ and tolerance values from $(0, 100]$. Each job consists of exactly three tasks (i.e. we set $t_{\min} = t_{\max} = 3$) and the responsible agent is rewarded for the job fulfillment with a payoff of 3 (i.e. we set $q_{\min} = q_{\max} = 1$). Consider an agent having a single skill out of a set of five skills. In addition, consider a job which is generated by Algorithm 3.1 on page 29 and contains exactly three skills. Then, it is easy to see that the probability of completing a job without the help of other agents is only 0.8%. For the cooperation cost we set the cost factor to $c = -0.25$.

Concerning the analysis of having all skills in the vicinity of an agent we can identify the following. Each agent is equipped with one skill out of a set of five skills and each agent has on average 15 neighbors. This leads to a probability of 86% that every agent has access to each possible skill. However, this will not result in high cooperation rates in the very beginning as the value-vectors have to be taken into account. We just give the system the possibility of reaching high levels based on these parameter settings. If more than 86% for the job completion rate is reached, we can assume that the social networking caused the increase.

During the adaptation phase the agents adapt to the single best performing neighbor (i.e. $|\mathcal{I}| = 1$). We also fixed the adaptation strength strategy to the first strategy, i.e. one fixed value for all agents ($\eta = 0.5$). To decide, if an agent is satisfied, an agent compares itself to the four best performing agents ($\varepsilon = 4$). Last but not least the social networking is performed with probability $\mathsf{Pr}_{\mathcal{N}} = 0.01$ and one agent is replaced in this step ($r = 1$). Table 5.1 summarizes this basic configuration. If not stated otherwise, all parameters are set as described above. We will only concentrate on different parameter values that are used for the examination of the parameter's influence.

We will analyze three outcomes of the system. The first measurement is the percentage of completed jobs over simulation time. The second is the so-called local cooperation willingness, which gives a percentage of how many neighbors will receive cooperate from a specific agent. The last measurement is the global cooperation willingness, which is calculated by comparing

all of the agents. Formally, we have for the local cooperation willingness LCW:

$$\mathsf{LCW} := \frac{\sum_{a \in \mathcal{A}} \frac{|\{b \in \mathcal{N}_a \mid \forall p \in \mathcal{P} : |\mathcal{V}_a(p) - \mathcal{V}_b(p)| \leq \Theta_a(p)\}|}{|\mathcal{N}_a|}}{|\mathcal{A}|} \tag{5.1}$$

and for the global cooperation willingness GCW:

$$\mathsf{GCW} := \frac{\sum_{a \in \mathcal{A}} \frac{|\{b \in \mathcal{A} \setminus \{a\} \mid \forall p \in \mathcal{P} : |\mathcal{V}_a(p) - \mathcal{V}_b(p)| \leq \Theta_a(p)\}|}{|\mathcal{A}| - 1}}{|\mathcal{A}|} \tag{5.2}$$

Table 5.1: Base Configuration.

Parameter	Meaning	Value
$\lvert\mathcal{A}\rvert$	population size	1000
$\lvert\mathcal{S}\rvert$	number of skills in the system	5
$\lvert\mathcal{S}_a\rvert$	number of skills per agent	1
$\mathcal{N}_{\max}$	maximum number of neighbors allowed per agent	30
m	number of propositions	5
$v_{\max}$	maximum rating for a proposition	100
$\Theta_{\max}$	maximum tolerance of an agent	100
$q_{\min}$	minimum payoff for task fulfillment	1
$q_{\max}$	maximum payoff for task fulfillment	1
$t_{\min}$	minimum number of tasks a job consists of	3
$t_{\max}$	maximum number of tasks a job consists of	3
k	job factor	10
c	cost-factor for charging helpers	-0.25
η	adaptation strength	0.5
ε	number of elite agents	4
$\lvert\mathcal{I}\rvert$	number of ideal agents	1
$\mathsf{Pr}_{\mathcal{N}}$	probability of executing social networking	0.01
r	number of replaced agents in social networking	1

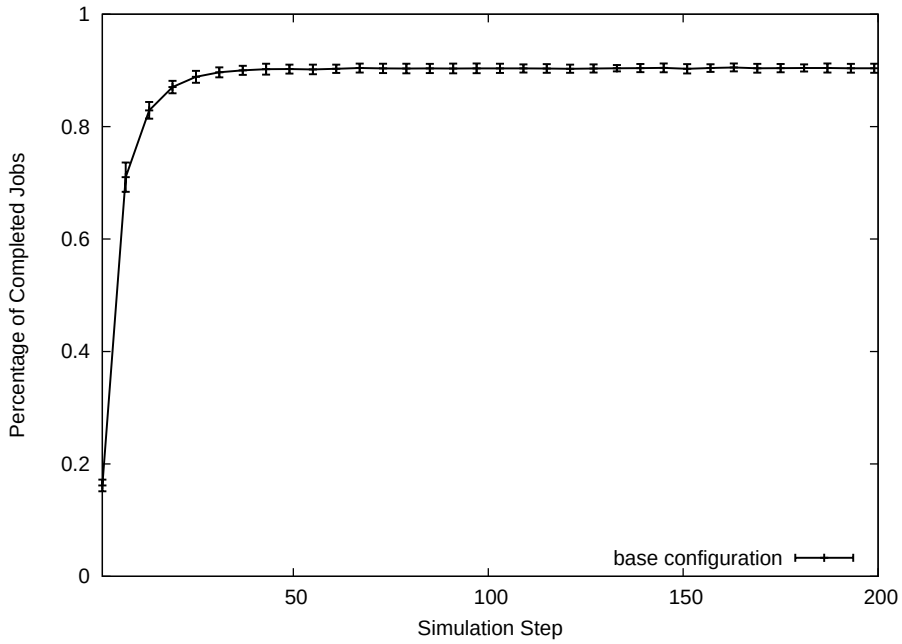


Figure 5.1: Percentages of completed jobs for the base configuration.

Figures 5.1–5.3 show the results for the base configuration. We present the averaged value over 30 individual repetitions. The bars present the standard deviations. As can be observed, the proposed mechanism leads to good results for the base configuration. Through the adaptation mechanism a high level of completed jobs is reached quickly.

For the local cooperation willingness given in Figure 5.2, we observe that nearly every agent will cooperate with its neighbors in the later simulation steps. Different from this behaves the global cooperation willingness as presented in Figure 5.3. The agents do not reach a state, where each agent would cooperate with all others from a global view. Nevertheless, the reached level of global cooperation willingness is high.

In the following sections, we will present the influence of different parameters on the system's behavior. Many details will be discussed in this experimental analysis. By the modular structure of this chapter, the reader can decide itself which parts to read in detail based on the results presented in the beginning and in the conclusion provided at the end of this chapter.

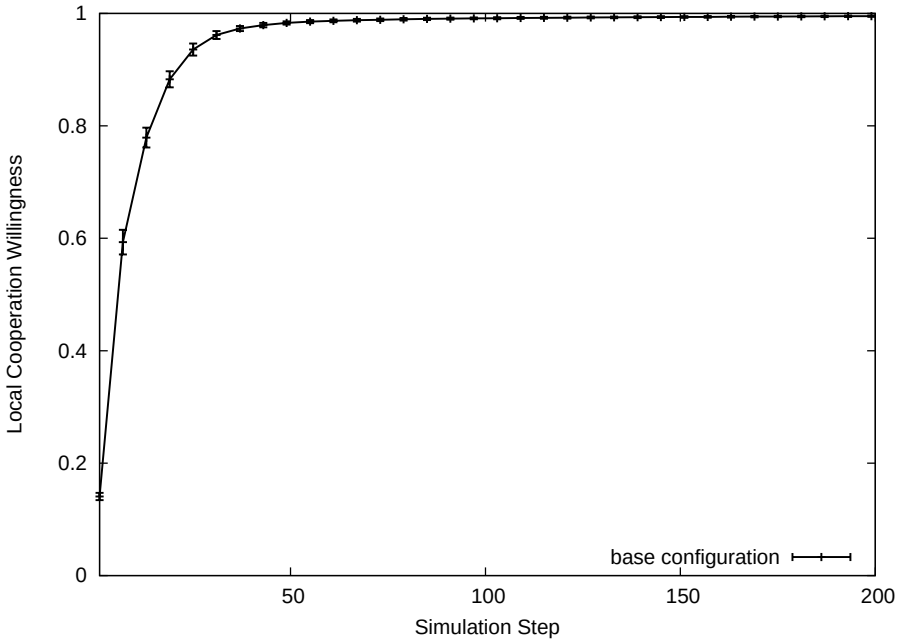


Figure 5.2: Local cooperation willingness for the base configuration.

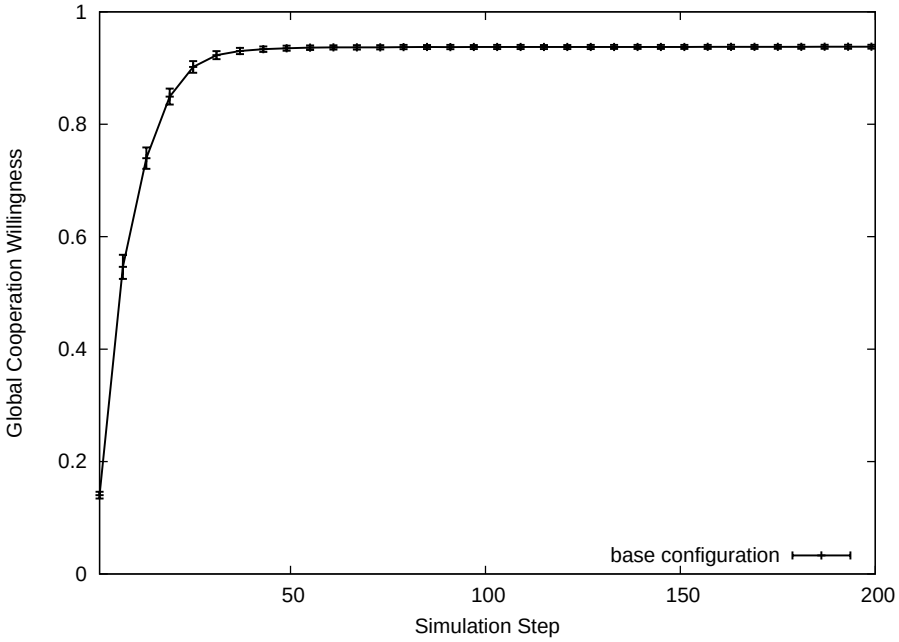


Figure 5.3: Global cooperation willingness for the base configuration.

5.2 Cooperation Cost-Factor

In this section, we examine the influence of cooperation cost factor c chosen from

$$\{0.0, \underline{-0.25}, -0.5, -0.75, -1.0\}$$

for the cooperation, where the underlined value is the given by the basic setup. In our model, agents have to pay if they help another agent to fulfill a job. Therefore, we use the cost factor, which determines how much an agent has to pay. The amount of negative payoff is calculated as the product of the task's payoff and the cost factor. Figures 5.4–5.6 give the results for these experiments.

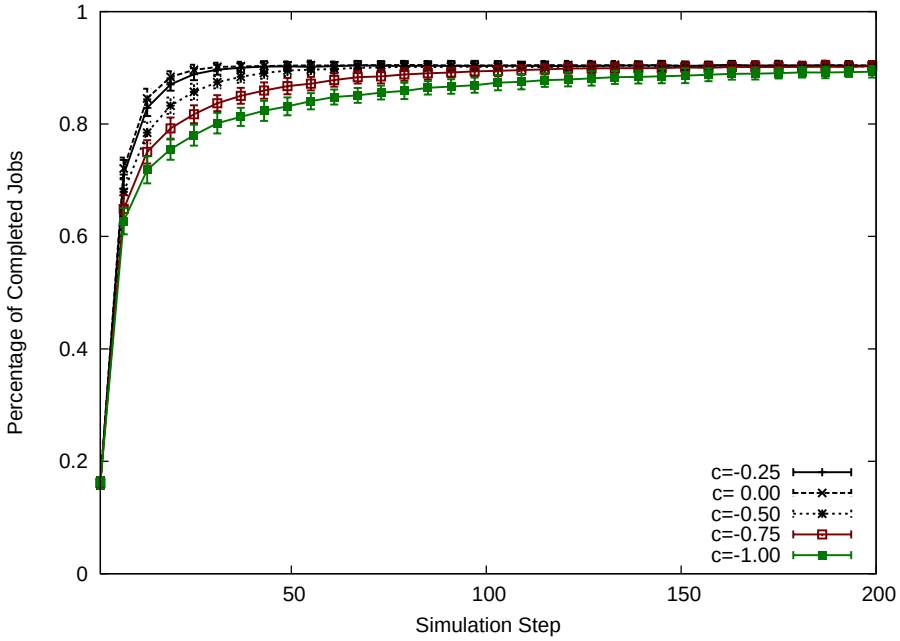


Figure 5.4: Percentages of completed jobs for variations of c.

As we can see in Figure 5.4, for all parameter values the same percentage of completed jobs can be reached although the simulation for $c = -1.0$ converges slightly earlier. There are only small differences in the early steps of the simulation, i.e. in the first 100 simulation steps. Here, we can see that the simulation without any cooperation costs reaches slightly better values. This is not surprising, as cooperating agents are not punished for their behavior. The standard value with $c = -0.25$ only performs marginally worse. The other simulations perform worse proportionally with the increase of cooperation costs and are more distinguishable.

With increasing cooperation costs the motivation for cooperation decreases. Agents, who did not help other agents but receive much help, have very high utility values and become very attractive as role models for adaptation. Therefore, the proposition value-value vectors of agents with lower utility values are moved into their direction, which leads to the observed development of lower cooperation, and, thus, in lower percentage of completed jobs. This can also be

observed in Figure 5.5, where the local cooperation willingness is presented. The local cooperation willingness is the percentage of neighbors with whom an agent would cooperate. For larger cooperation costs the number of interactions where a job cannot be fulfilled gets greater, too. This is because at least one task cannot be processed due to a missing cooperative neighbor in the local neighborhood.

In contrast to the local cooperation willingness, the global cooperation willingness is the percentage of all agents in the agent set $\mathcal{A}$ that an agent is willing to cooperate with. We took this as a global measurement for showing the global influence of the adaptation mechanism on the values in the whole population. Figure 5.6 presents the global cooperation willingness. If both the local and global measurements are compared, it can be seen that the global cooperation willingness converges at about 95%, although the local cooperation willingness reaches 100% for values $c \geq -0.5$ within the 200 simulation steps. This shows that the ratings locally converged to similar values but they differ slightly in the global perspective.

We have seen that the influence of the cooperation costs is not that strong on the system's performance. Although cost-free cooperation is slightly better we claim that cost-free cooperation is not a necessity for achieving high levels of cooperation. Therefore, we identify our standard value of $c = -0.25$ as the best choice for this parameter since it results in the best system's performance if cost-free cooperation is prohibited. Only the cost-free cooperation leads to better results, but this is just a comparison to show how much is lost if cooperation produces costs.

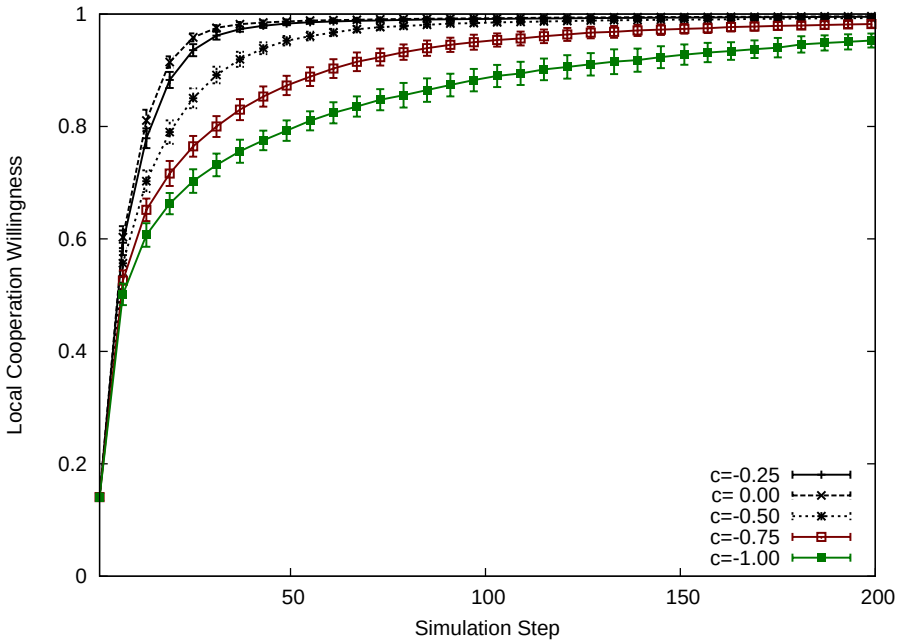


Figure 5.5: Local cooperation willingness for variations of c.

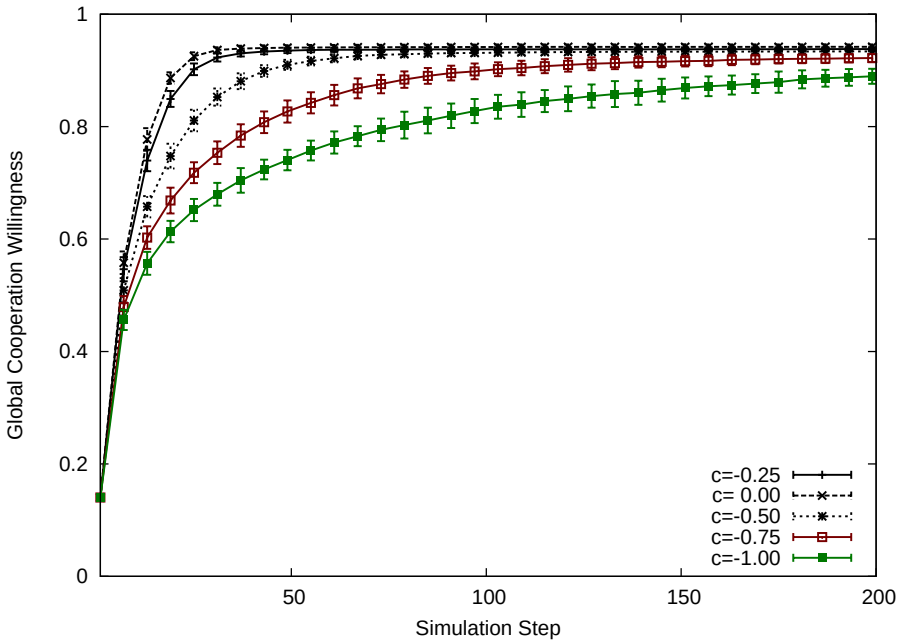


Figure 5.6: Global cooperation willingness for variations of c.

5.3 Strength of Adaptation

In this section, we examine the influence of the adaptation factor

$$\eta \in \{0.0, 0.1, 0.3, \underline{0.5}, 0.7, 0.9, 1.0\}$$

where the underlined value comes from the basic setup. A main idea of our model is the adaptation step described in Section 3.2.2. The strength of this adaptation depends on the adaptation strength η. As standard value for the adaptation strength we have chosen $\eta = 0.5$ since this is not purely copying and yield to good results in early experiments. The results for different values of this parameter are shown in Figure 5.7. As can be seen, all experiments with $\eta \neq 0.0$ reach the same level of completed jobs. Only the speed of achieving this goal slightly differs. But we can see almost the same behavior for totally different values and, therefore, we can build small sets of parameter values that lead to very close results. The best results can be achieved for $\eta \in \{0.3, 0.5, 0.7\}$. The second best results can be achieved for $\eta \in \{0.1, 0.9, 1.0\}$ which leads to a slightly slower development in reaching the high level of completed jobs of about 90%. The worst results are achieved for $\eta = 0.0$ where less than 20% of the jobs can be completed due to no adaptation and the resulting differences between the proposition ratings.

The local and global cooperation willingness are shown in Figure 5.8 and Figure 5.9, respectively. Here, we can see that for all settings with $\eta \neq 1.0$ the global cooperation willingness does not reach 100%. This is due to the fact that no copying of values takes place but only adaptation.

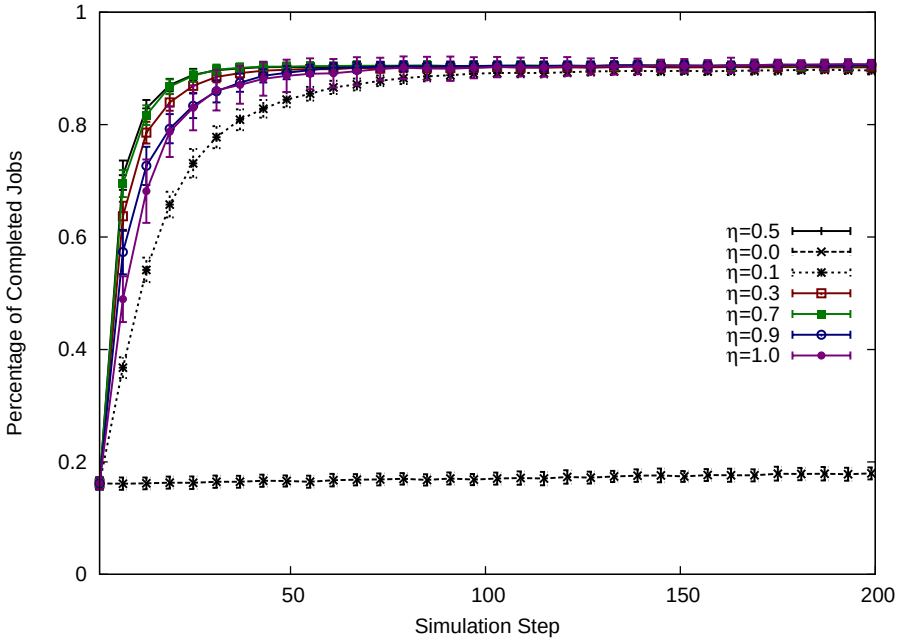


Figure 5.7: Percentages of completed jobs for variations of η.

On the other hand we reach 100% of local cooperation willingness for all simulations in which adaptation takes place. Like for the percentage of completed jobs, we only see a difference in the speed of reaching this high level. As can be seen, the global cooperation willingness is slightly higher for $\eta \in \{0.7, 0.9\}$ than for $\eta = 0.5$. However, $\eta = 0.5$ leads to the desired high levels of local cooperation willingness requiring the lowest number of iterations. It is also observable that the convergence is slower for higher or lower values of the adaptation strength. This is obvious, as for low values the value-vectors are moved very slowly. Therefore, the tolerance threshold for the components' differences is not reached in such a fast way. Interestingly, we also get a slower development if the adaptation strength is too high. We belief that this is due to the fact that the agents in a neighborhood do not all have the same ideal they adapt to. Therefore, almost copying the values of their ideal leads perhaps to cooperation with this agent but on the other hand the distance to other agents gets greater.

To sum up, we identified that half-step adaptation ($\eta = 0.5$) leads to the best results. We claim that this is due to the fact that we have the best balance between pure copying and no adaptation at all. The agents change their values but not too rigorously. Therefore, they do not depart too much from other agents as pure copying does, which leads to the observed behavior.

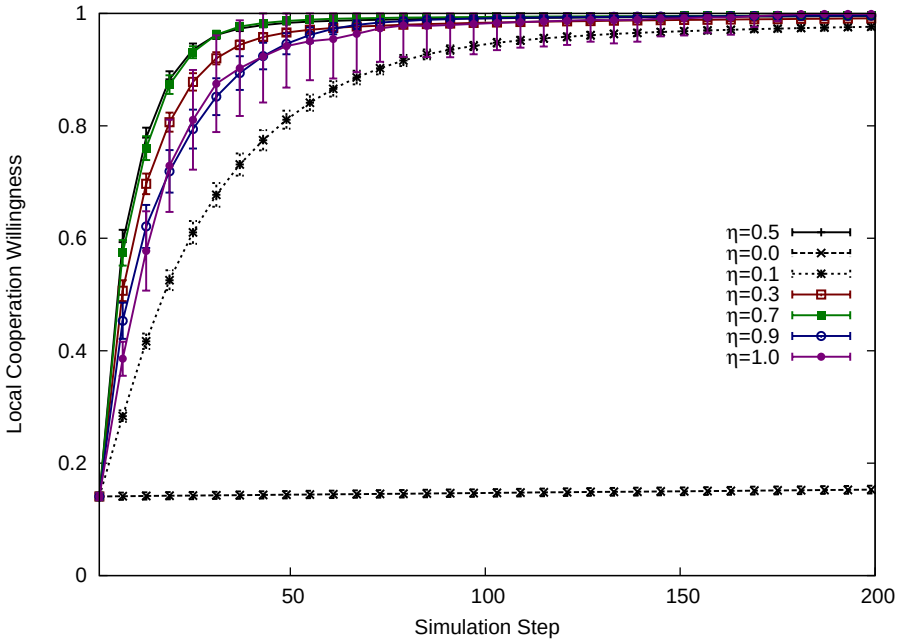


Figure 5.8: Local cooperation willingness for variations of η.

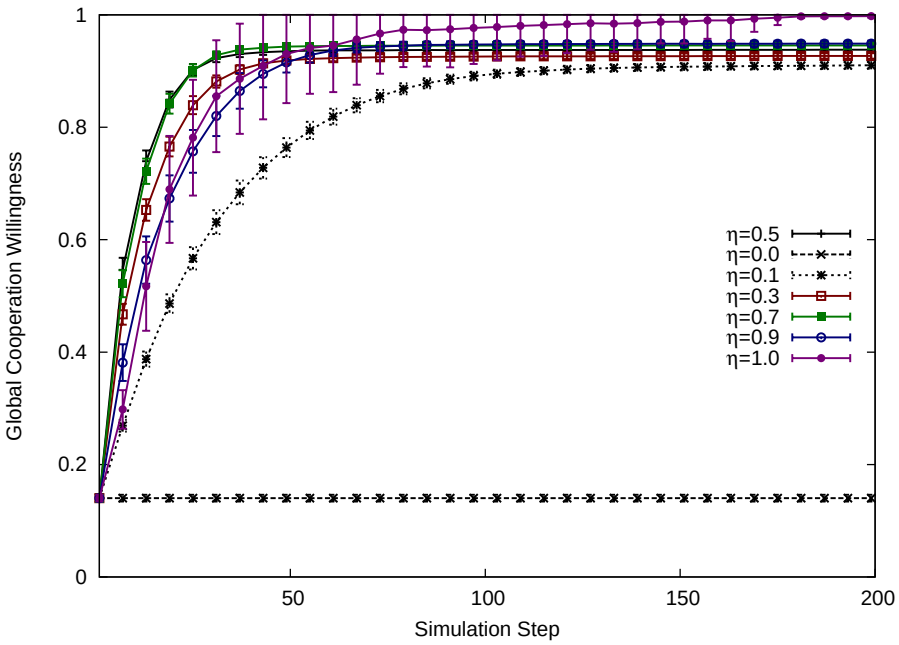


Figure 5.9: Global cooperation willingness for variations of η.

5.4 Size of Elite Set

The size of the elite set has large influence on the overall system's behavior. The size determines how many agents will be satisfied by definition and how many would possibly tend to adapt. If the size of the elite set is near the number of neighbors, nearly all agents will be satisfied as the probability of belonging to the elite set is very large. Since the size of the neighborhoods is 15 on average, an agent can measure the performance of 16 agents. The ε best agents are called the elite set. If the agent is not in this elite set and has at least one agent in its neighborhood which did not cooperate in the current simulation step the agent is called *unsatisfied* and will adapt its proposition-values and perform a social networking step with probability $\mathsf{Pr}_{\mathcal{N}}$. The larger the elite set, the lower is the probability for performing adaptations. Therefore, we have chosen the values for this examination as follows: $\varepsilon = 2$ is a very small set leading to many unsatisfied agents. $\varepsilon = 8$ is a very big set containing approximately half of all agents that a single agent knows about. $\varepsilon = 4$ and $\varepsilon = 6$ lead to sets somewhere in between these extremes, where $\varepsilon = 4$ is our base value for this parameter.

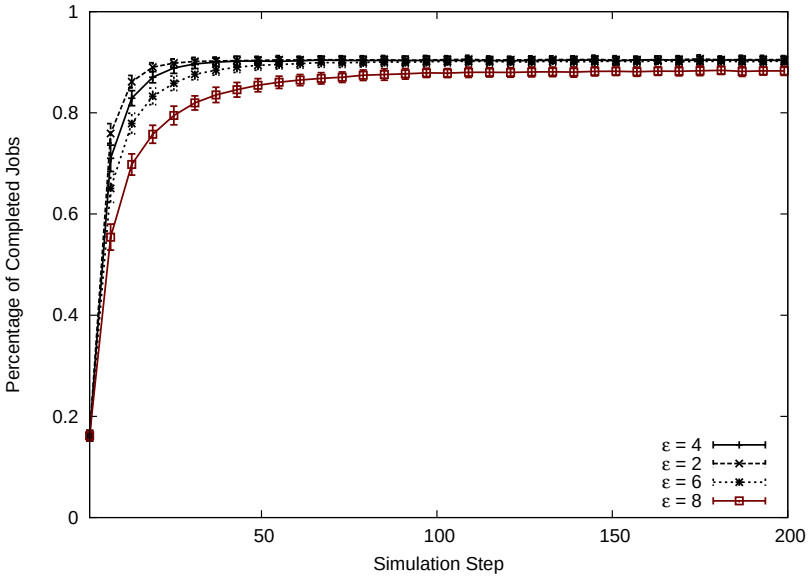


Figure 5.10: Percentages of completed jobs for variations of ε.

The results of the variation of the elite set size ε are shown in Figures 5.10–5.12. As can be seen for $\varepsilon \neq 8$ we get very similar results with only small differences. All simulations reach the same level of completed jobs but only differ in the speed of development. We observe the fastest convergence for $\varepsilon = 2$ and the slowest convergence for $\varepsilon = 6$. As stated above the pressure on the agents is very large for small elite sets, which leads to the observed behavior. For all $2 \leq \varepsilon \leq 6$ we also observe that the local cooperation willingness is very close to 100%. As expected the performance is not satisfying for a large elite set containing eight agents. Here, the pressure on the agents is very small which leads to less adaptation and less cooperation willingness locally and globally. Therefore, fewer jobs can be processed. As a conclusion,

we identify $\varepsilon = 4$ as the best value for further experiments. It is small enough to have high adaptation pressure but does not introduce too much pressure, as e.g. $\varepsilon = 2$. However, for comparison reasons we presented the results for $\varepsilon = 2$.

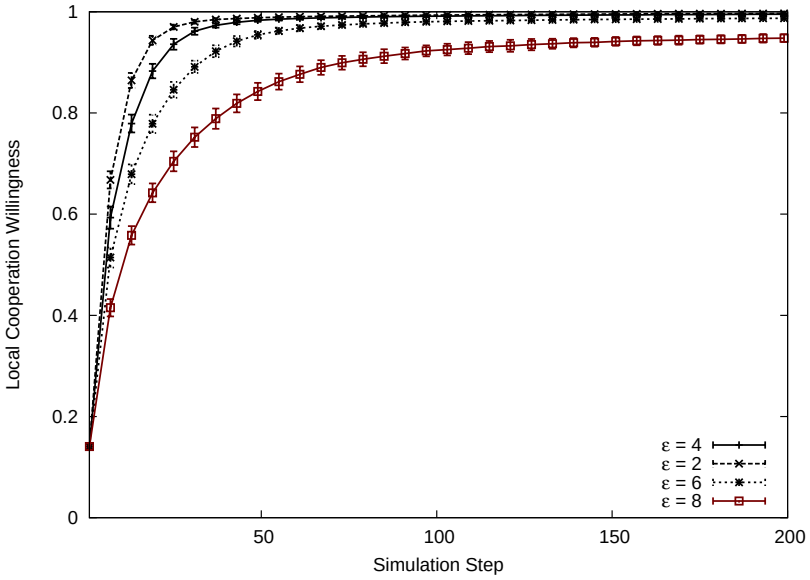


Figure 5.11: Local cooperation willingness for variations of ε.

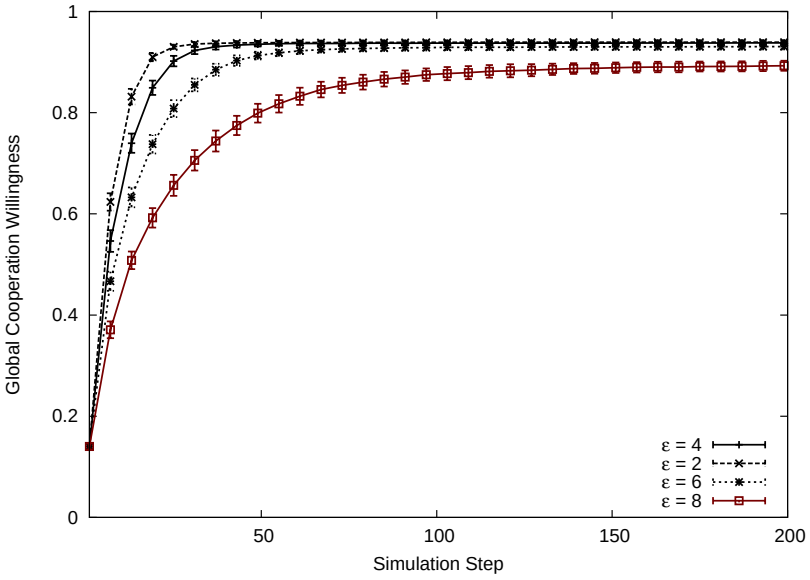


Figure 5.12: Global cooperation willingness for variations of ε.

5.5 Number of Propositions

In this section, we examine the influence of the number of propositions. Clearly, the larger the number of propositions the lower the probability of fulfilling all criteria as their number grows by each proposition.

Figure 5.13 presents the percentage of completed jobs for different numbers of propositions over the simulation steps. The number of propositions for the different runs was set to $m \in \{\underline{5}, 10, 15, 20\}$, where $m = 5$ is the base value. As we can see, the number of propositions has strong influence on the performance. For small and medium numbers of propositions ($m = 5$ and $m = 10$) we have very similar developments for the percentage of completed jobs. In all four scenarios we reach the same high level of cooperation. The only difference is the convergence speed. Here, the development speed decreases with an increasing number of propositions.

As the values for the propositions could be used to influence the behavior of the agents, a larger number of propositions leads to the possibility of having more detailed descriptions of the agents' behavior mechanism. Since all parameter values lead to high cooperation rates, we select $m = 10$ for the next experiments. On the one hand we have a reasonable high number of constraints and on the other hand we have a quite fast development to mutually cooperative agents. This is also reflected when we look at the local and global cooperation willingness (cf. Figure 5.14 and Figure 5.15). All in all the results show that the approach is able to produce good results even if high numbers of propositions are considered.

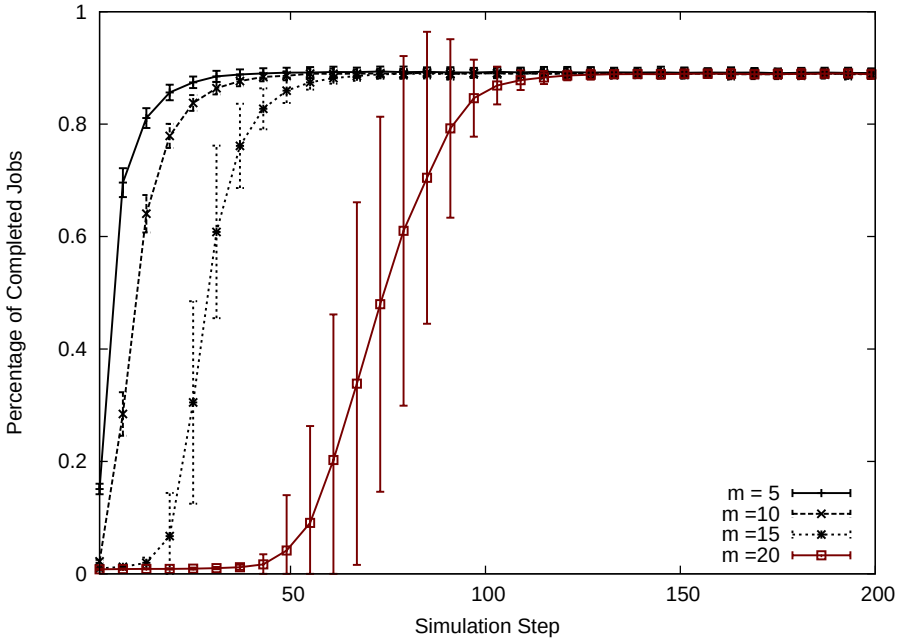


Figure 5.13: Percentages of completed jobs for different numbers of propositions.

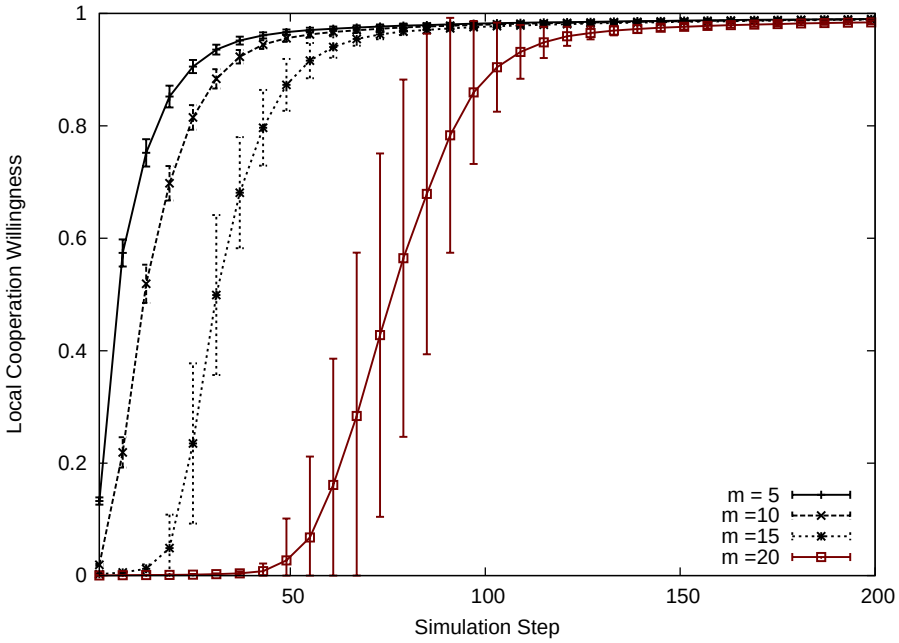


Figure 5.14: Local cooperation willingness for different numbers of propositions.

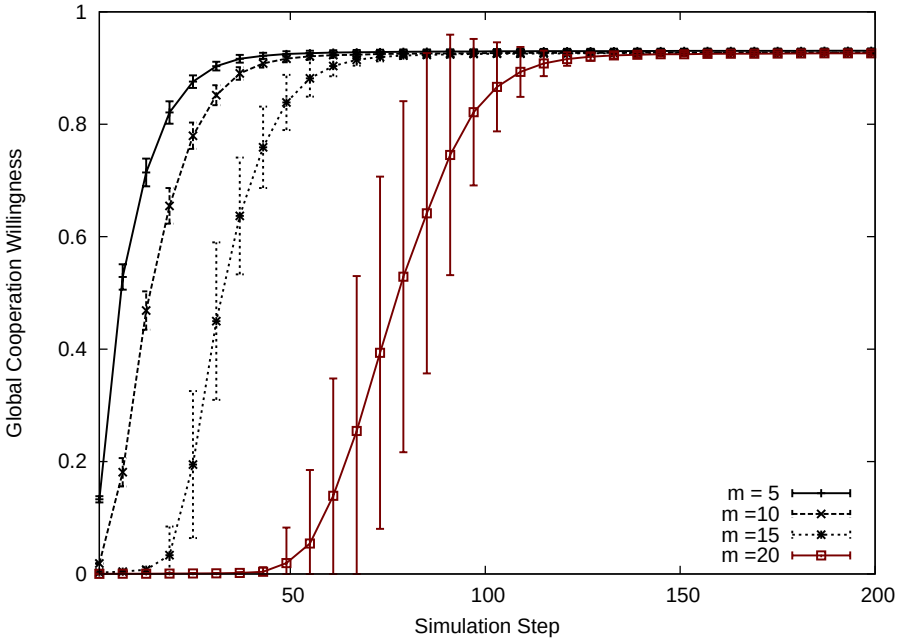


Figure 5.15: Global cooperation willingness for different numbers of propositions.

5.6 Influence of the Tolerance

Using the following experiments, we analyze the influence of the tolerance space on the percentage of completed jobs. We selected the maximum threshold value $\Theta_{\max}$ from $\{\underline{100}, 75, 50, 25\}$ where $\Theta_{\max} = 100$ was the base value for this parameter. The threshold is initialized within the interval $(0, \Theta_{\max}]$ using a uniform distribution. This leads to an expected value of $\frac{\Theta_{\max}}{2}$ on average for the threshold values.

It is obvious, that agents which have a value of 100 for a specific proposition are extremely tolerant as the maximum difference of two values for a specific proposition is 100, which is the maximum rating. Lower tolerance values lead to more intolerant agents in the population. Figures 5.16–5.18 present the results of these experiments. As can be observed, the parameter $\Theta_{\max}$ has large influence on the performance of the system. We cannot identify any development of the percentage of completed jobs, when $\Theta_{\max}$ is set to 25. This is due to the fact that nearly no cooperation can take place in such a system. Therefore, agents solely gain profit from jobs that they can process on their own. This leads to the situation that all agents nearly have the same profit in each simulation step and, therefore, each agent is within its own elite set. This results in agents that are always satisfied and which do not adapt at all. There is no way to escape from this situation and that is why we get the observed behavior. For higher tolerance values very high cooperation rates can be reached. The only difference can be found in the development speed, similar to the results for the number of propositions.

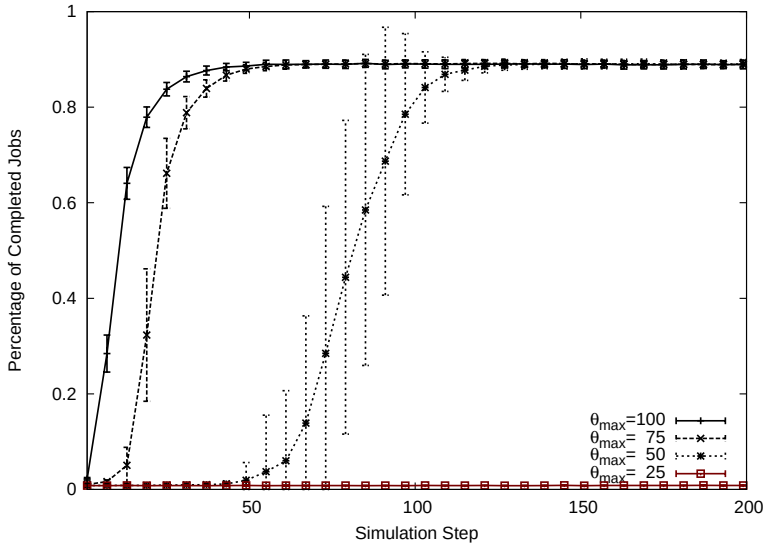


Figure 5.16: Percentages of completed jobs for different tolerance values.

As a conclusion, we assume that $\Theta_{\max} = 50$ is a reasonable medium tolerance value, since it leads to high levels of cooperation and is low enough to model different degrees of tolerance if this is intended in a specific application. Surely, in such scenarios the pressure on the agents is very high but we showed with these experiments that cooperation could emerge even if the

agents are relatively intolerant.

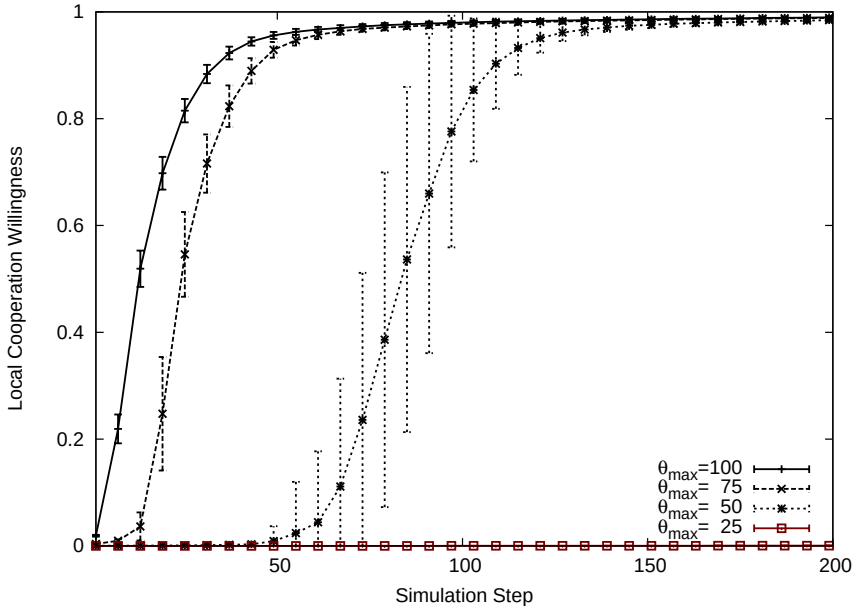


Figure 5.17: Local cooperation willingness for different tolerance values.

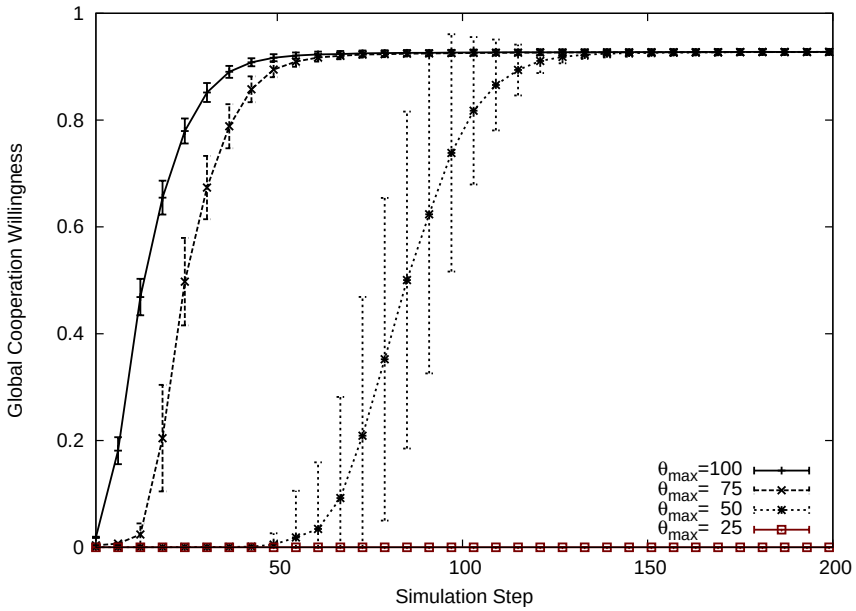


Figure 5.18: Global cooperation willingness for different tolerance values.

5.7 Adaptation Strength Strategies

In Section 5.3, we showed that the adaptation strength has large influence on the cooperation rate. We now investigate the influence of different strategies for the adaptation strength. As mentioned in Section 3.2.2, we investigated three different strategies. Figures 5.19–5.21 present the results for these experiments.

As can be observed, the random selection strategies show different system's performance. If each agent selects a randomly chosen adaptation strength $\eta \in [0, 1]$ in the initialization phase, the percentage of completed jobs increases much faster in the beginning but does not reach the same level as the fixed-η strategy. After simulation step 125, the job completion rate of the fixed-η strategy performs best. The random initialization of η only outperforms the fixed $\eta = 0.5$ strategy at the beginning of the simulation. The "random selection in every step" strategy is worse than the two other strategies. Only at the very end of the simulations this strategy results in more completed jobs than the second strategy and gets very close to the performance of the fixed-η strategy.

From these experiments we can conclude that the fixed $\eta = 0.5$ strategy value for all agents in all simulation steps is the best strategy as it is simple to understand for a deep analysis of the system and leads to sufficiently high levels of cooperation in a fast way.

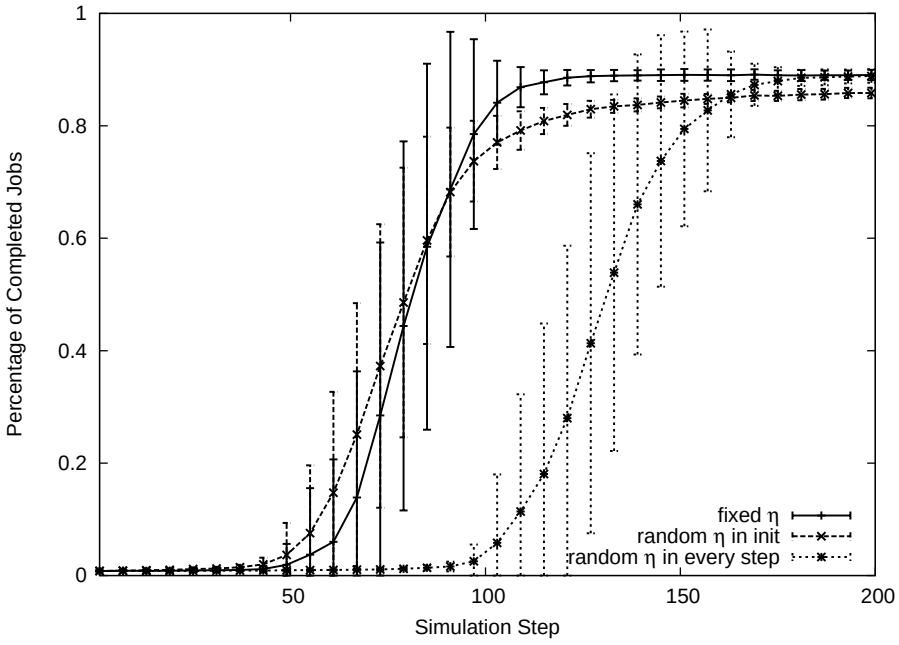


Figure 5.19: Percentages of completed jobs for different types of η.

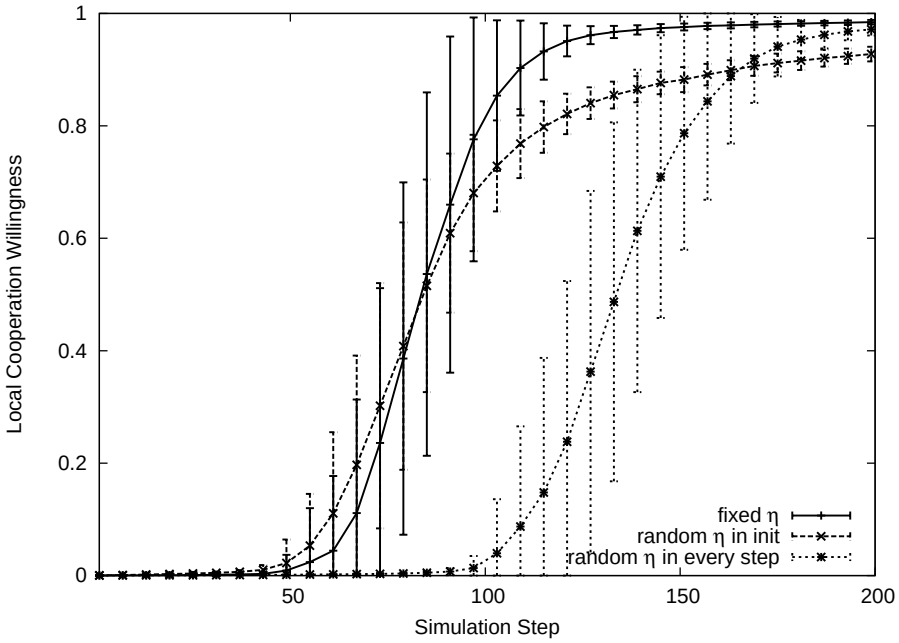


Figure 5.20: Local cooperation willingness for different types of η.

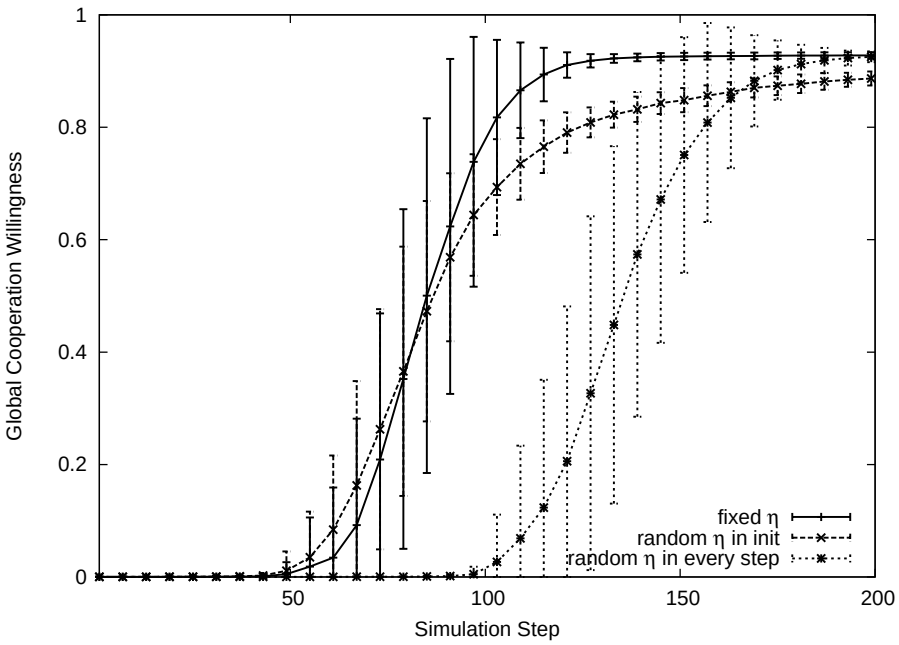


Figure 5.21: Global cooperation for different types of η.

5.8 Ideal Selection Strategies

The adaptation of the agents is strongly influenced by the selected ideals. We analyze the influence of different strategies as described in Section 3.2.1. As stated in that section, there are mainly three different strategies to select the set of ideals. The first is to select the best performing agents from the neighborhood, the second is a random selection of neighbors without respect to their performance and the third is to select the worst performing neighbors. Another aspect that is analyzed in this section is the size of the ideal set. In all earlier experiments, the agents adapt to the single best performing neighbor. Now, we will also analyze ideal sets with two and three agents.

We start with the examination of the three different strategies if the ideal set consists of a single agent. Figures 5.22–5.24 present the results for these experiments. As Figure 5.22 shows, there is a significant difference in the system's performance in the first 120 simulation steps. There, the best results can be observed, if the best neighbor is selected as the ideal agent. The other two strategies can also be distinguished, as the selection of the worst agent in the neighborhood leads to better results than selecting a random neighbor. This is also reflected by the local and global cooperation willingness (see Figure 5.23 and Figure 5.24). Although selecting the worst performing agents leads to better results in this settings compared to the random selection, one can notice that the mean value for the local and global cooperation willingness is higher for the random selection strategy in the last 30 simulation steps.

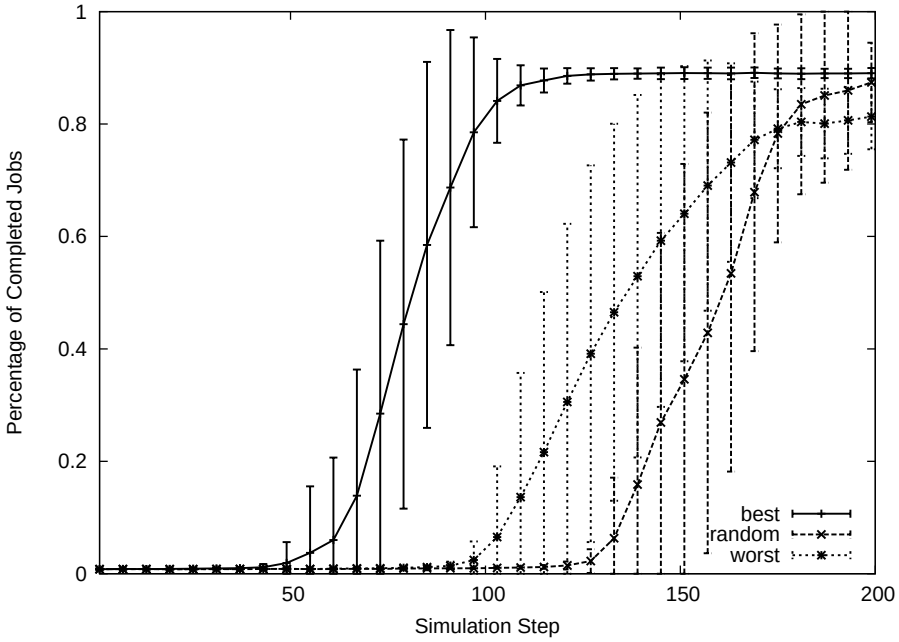


Figure 5.22: Percentages of completed jobs for selecting one ideal.

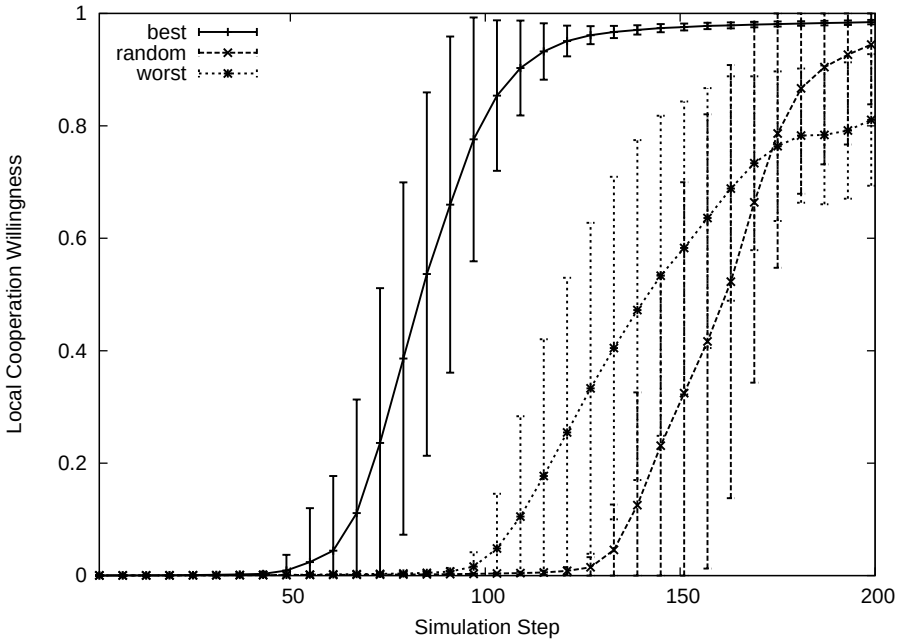


Figure 5.23: Local cooperation willingness for selecting one ideal.

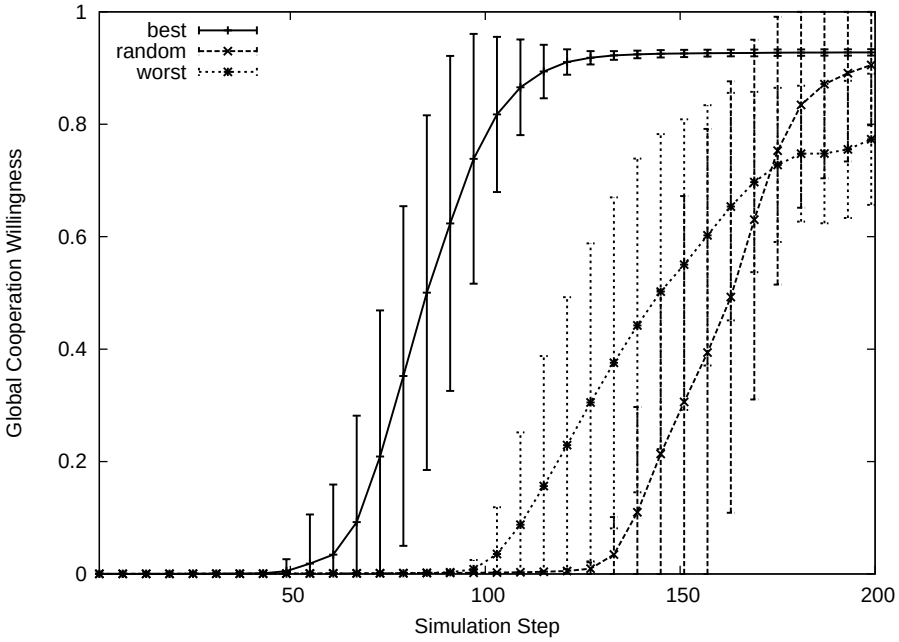


Figure 5.24: Global cooperation willingness for selecting one ideal.

Figures 5.25–5.27 present the results if the set of ideal agents contains two agents. Here we see different results. However, the strategy of selecting the best neighbors as ideals again leads to the best results. Noticeably, the strategy of selecting random ideals performs better now. The mean performance over all runs is better than the worst selection strategy, which is different to the simulations with only one ideal agent. It also reaches the same level as the best selection strategy. However, the worst selection strategy is not able to reach this high level within the given time window of 200 simulation steps. The local and global cooperation willingness (cf. Figure 5.26 and Figure 5.27) show the same behavior. This leads to the conclusion that if the ideal set consists of two agents and we have an application where the profit of other agents belongs to private knowledge, the best and worst selection strategy are not applicable as both cannot be computed. If this is the case the random selection strategy seems to be a good alternative. It is able to reach the same level of cooperation as these strategies, although the convergence speed is slower.

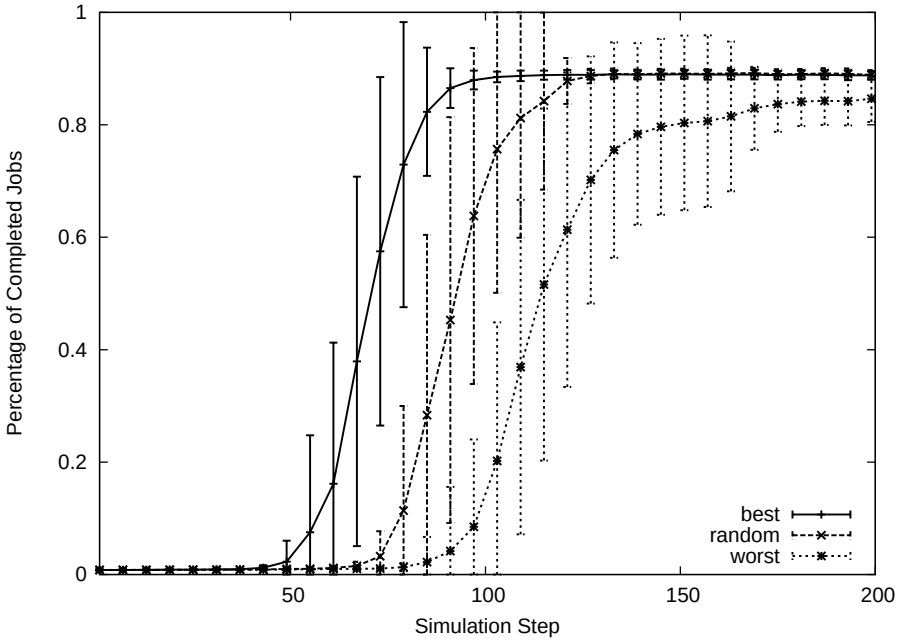


Figure 5.25: Percentages of completed jobs for selecting two ideals.

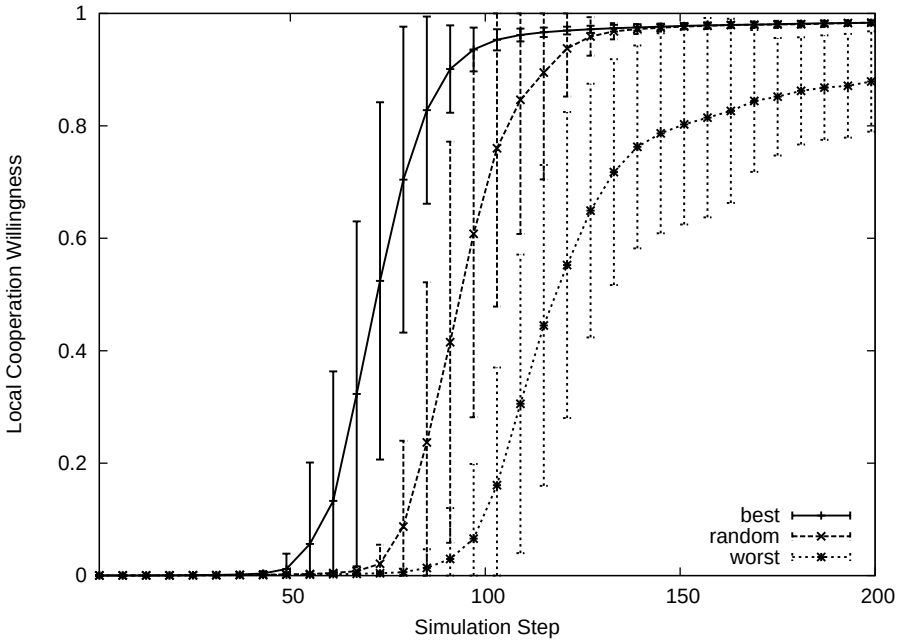


Figure 5.26: Local cooperation willingness for selecting two ideal.

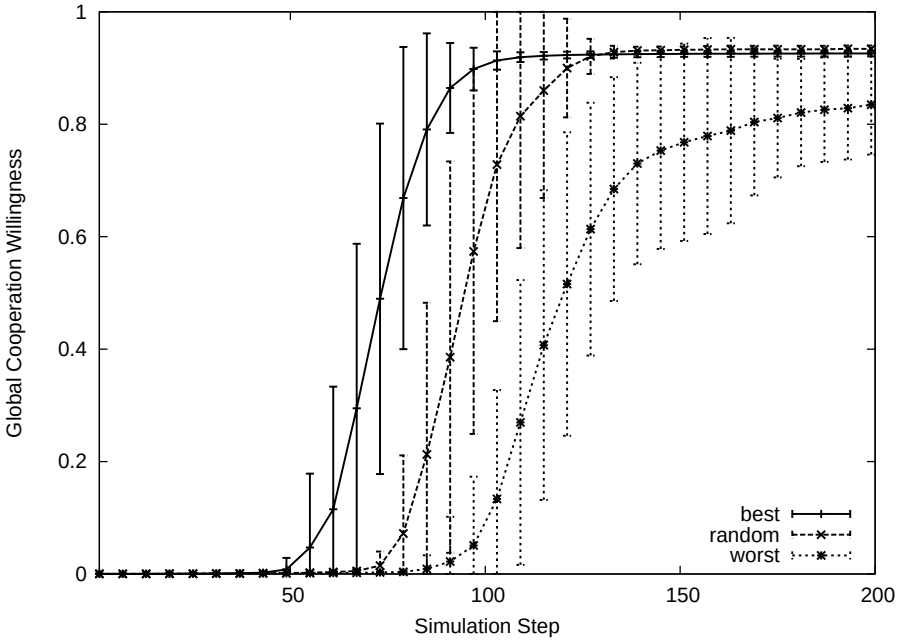


Figure 5.27: Global cooperation willingness for selecting two ideal.

Figures 5.28–5.30 show the results for ideal sets with cardinality three. Compared to the behavior seen before, the convergence gets faster. Here, we can see again that "the best selection strategy" performs best and that the random selection strategy performs slightly worse. Again, both reach the same level of cooperation but with slightly different time requirements. All in all, the same behavior as for ideal sets containing two agents can be seen.

To sum up the influence of the ideal selection strategies, we claim that the strategy "select the single best neighbor" leads to the best results. As the agents only need to adapt to a single agent the amount of computational power is low in this strategy. This strategy can only be improved if more than one agent constitutes the ideal set. If the application at hand does not allow sensing the profit of the neighbors then the best choice is to adapt to a randomly selected set of neighbors. Here, it is important to have at least two agents in the ideal set and, therefore, to adapt to two agents in one step. This leads to the same high levels of completed jobs as the best selection strategy does. This strategy has the disadvantage of requiring more computation as the adaptation is based on more than one other value vector. Nevertheless, it has the great advantage that it does not need any knowledge about the performance of the neighboring agents.

As our model allows the agents to measure their neighbors' performances, we will select the "single best selection strategy" in the remaining experiments.

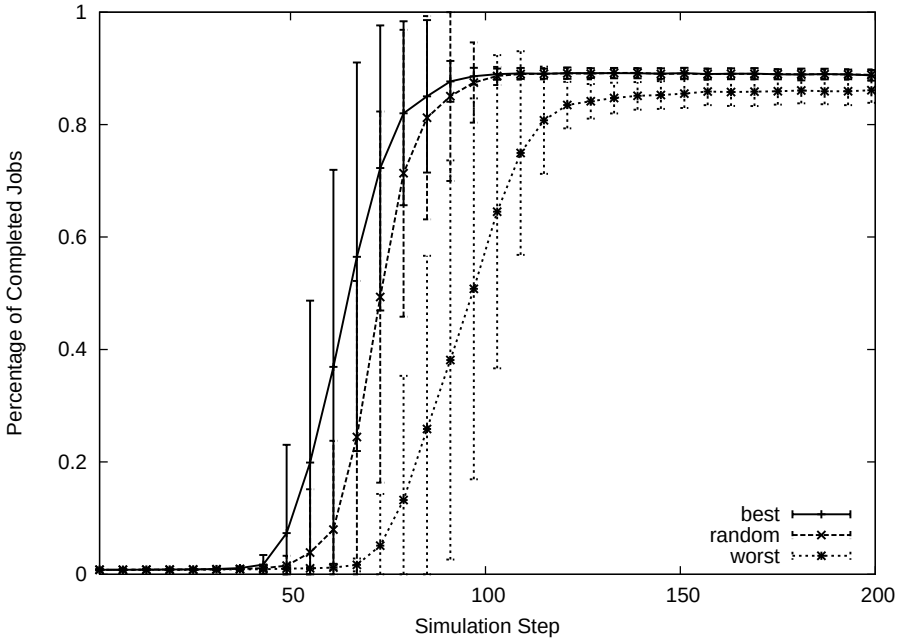


Figure 5.28: Percentages of completed jobs for selecting three ideals.

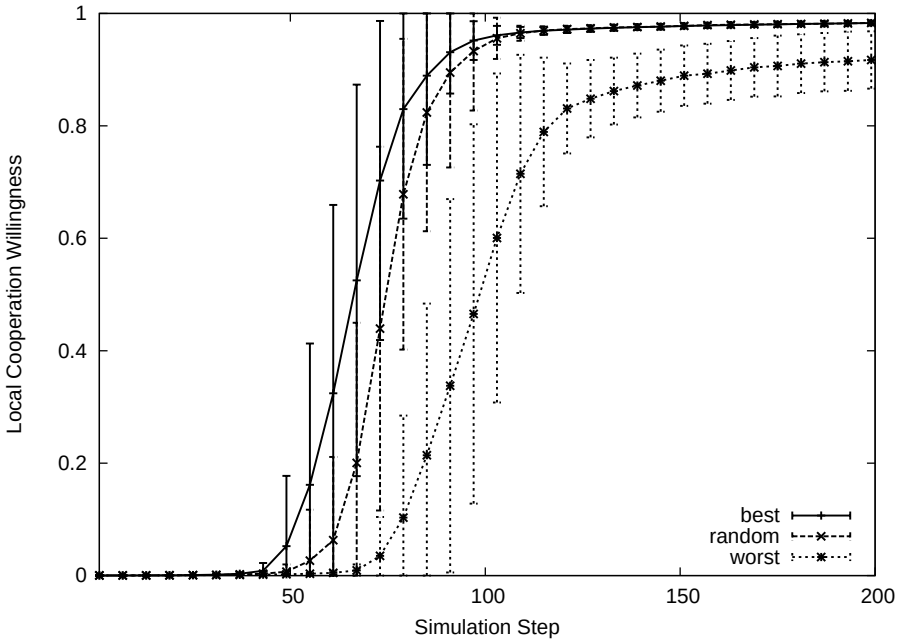


Figure 5.29: Local cooperation willingness for selecting three ideal.

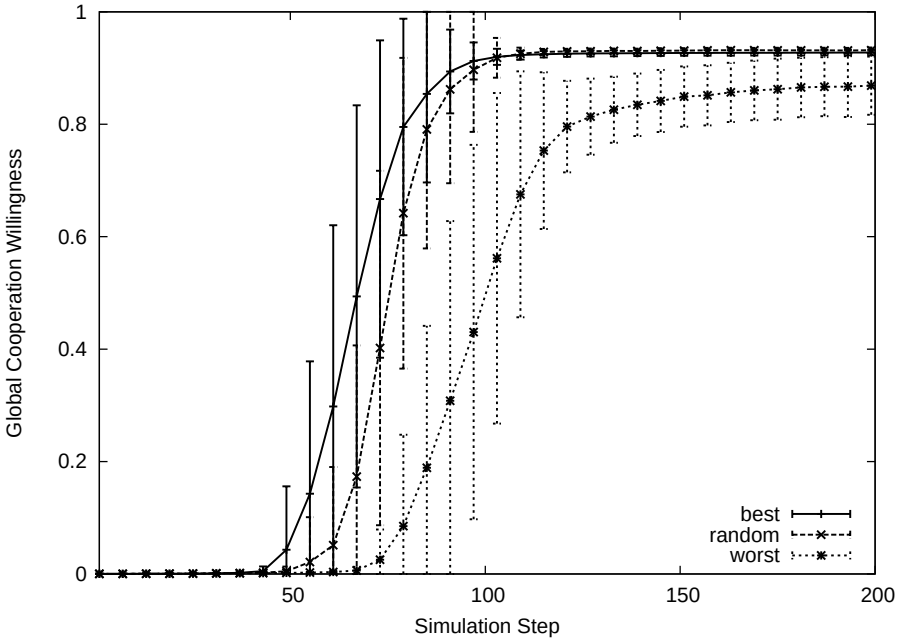


Figure 5.30: Global cooperation willingness for selecting three ideal.

5.9 Size of the Neighborhoods

In this section, we examine the influence of different neighborhood sizes. Obviously, the neighborhood size has big impact on the system's performance as the probability for finding cooperative neighbors, which provide a required skill, increases with the number of possible interaction partners. We set the initial number of neighbors to $|\mathcal{N}_a|_{\text{init}} \in \{5, 10, \underline{15}, 20, 25, 30\}$ where 15 is the value of our basic configuration. Note that due to the social networking part the neighborhood of an agent may grow. In these experiments we fixed the number of allowed neighbors to the number of initially assigned neighbors. Again, we used an Erdõs-Rényi random network with connection probabilities that lead to the intended mean degree of the initial neighborhood sizes.

Figures 5.31–5.33 present the results of these experiments. With increasing number of neighbors the percentage of completed jobs grows (cf. Figure 5.31). This is caused by the larger possibility of having a neighbor that provides a specific skill and that is willing to cooperate. For $|\mathcal{N}_a|_{\text{init}} \geq 15$ it can be observed that high percentages of completed jobs are achieved. For $|\mathcal{N}_a|_{\text{init}} = 5$ or $|\mathcal{N}_a|_{\text{init}} = 10$ no job that needs cooperation is completed. Only a small fraction of the jobs that can be completed without the help of others is processed. We expected this behavior, as for $|\mathcal{N}_a|_{\text{init}} = 5$ the possibility of having all required skills in the neighborhood is very low. In these experiments the total number of skills in the system was set to 5 as it is provided by the basic configuration. Although the possibility is larger if each agent has 10 neighbors the percentage of completed jobs is nearly zero as no small groups mutually cooperate in the system. This is also reflected by Figure 5.32 and Figure 5.33 which give the local and global cooperation willingness respectively. As no mutual cooperation takes place, no small group can locally perform good enough to make others adapt to them. Because the size of the elite set is 4 in the basic configuration, it is obvious that in very small neighborhoods the number of satisfied agents is very high which again leads to very low percentages of adapting agents.

Interestingly, it can be observed that increasing the neighborhood size to 15 on average leads to a good performing system. We reach high levels of completed jobs and the cooperation willingness increases. In an additional experiment we analyzed another set of initial neighborhood sizes, i.e. $|\mathcal{N}_a|_{\text{init}} \in \{11, 12, 13, 14\}$ and compared them to the results gained from $|\mathcal{N}_a|_{\text{init}} = 10$ and $|\mathcal{N}_a|_{\text{init}} = 15$ (cf. Figure 5.34). We wanted to get insights in what the minimum number of neighbors is to reach high levels of completed jobs. To favor readability of the figure we did not plot the standard deviation for these experiments. It can be observed that 12 neighbors are needed to reach higher percentages of completed jobs within 200 simulation steps. There is a small improvement also for 11 neighbors but only slightly better than for smaller neighborhoods. To reach more than 80% of completed jobs, 13 neighbors are needed and larger neighborhoods just improve the reachable amount of completed jobs and speed up the development as it can be seen in Figure 5.34. This underlines the theoretical results of our formal analysis on neighborhood sizes (cf. Section 4.1).

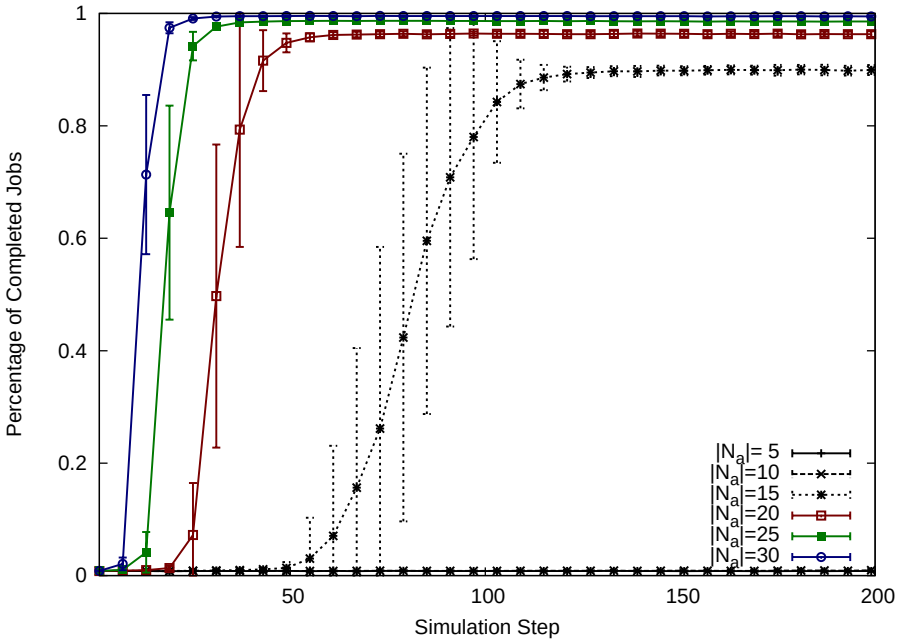


Figure 5.31: Percentages of completed jobs for different neighborhood sizes.

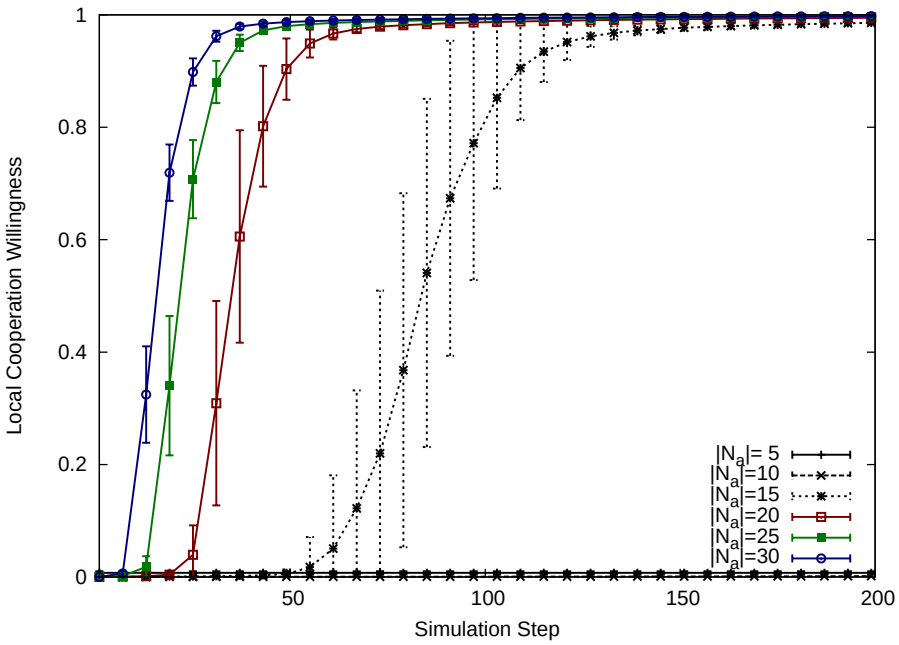


Figure 5.32: Local cooperation willingness for different neighborhood sizes.

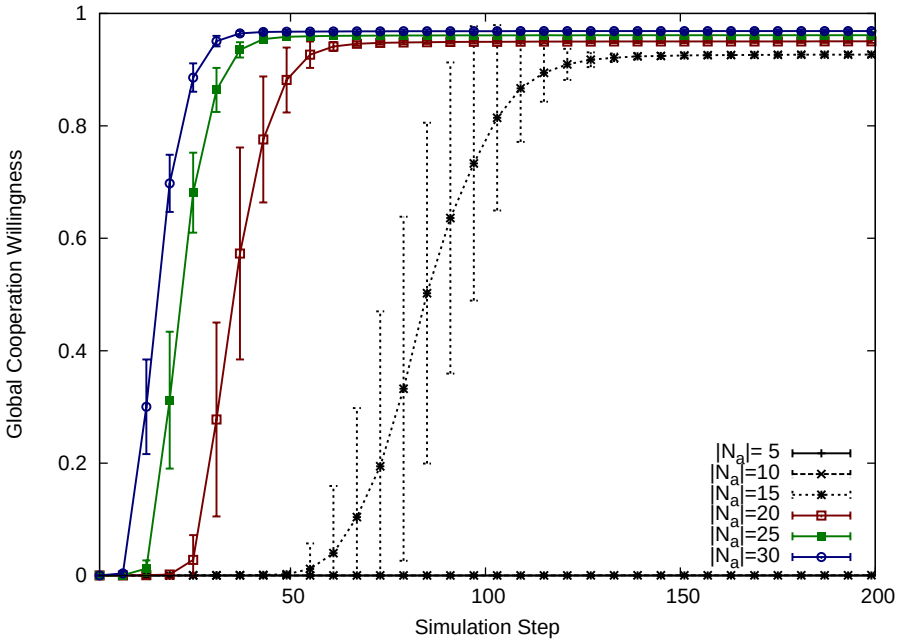


Figure 5.33: Global cooperation willingness for different neighborhood sizes.

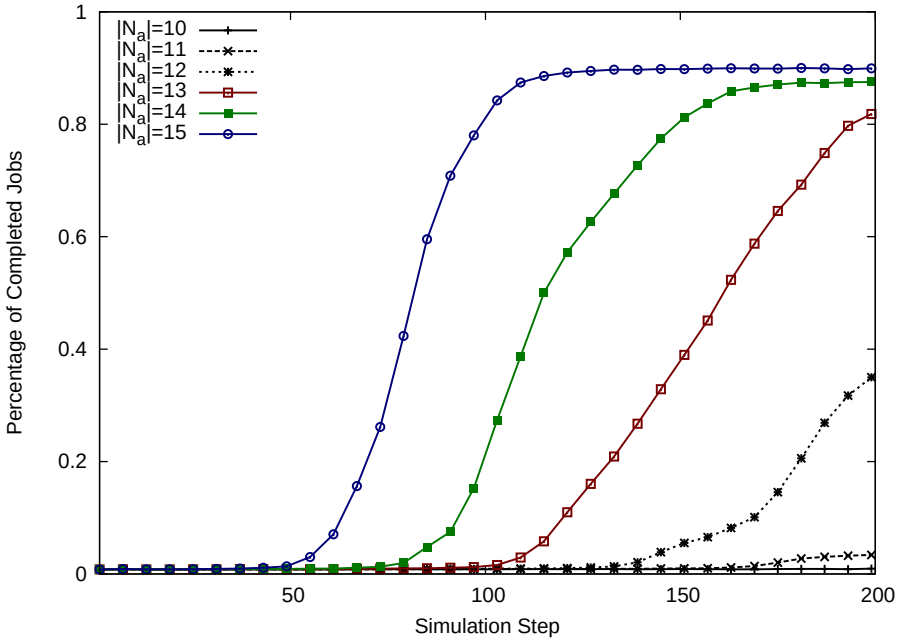


Figure 5.34: Percentages of completed jobs for small neighborhood sizes.

5.10 Complexity of Jobs

In this section, we analyze the influence of the jobs' complexity, i.e. the influence of $t_{\min}$ and $t_{\max}$. We define the complexity of a job as the number of tasks that have to be fulfilled. The complexity of each single job j is $t_{\min} \leq |j| \leq t_{\max}$. To analyze the influence we decided to experiment with $|j| \in \{1, 2, \underline{3}, 4, 5\}$, first. $|j| = 3$ is the parameter value of the base configuration. We assume that if the complexity of a single job increases the job processing gets harder for an agent because the agent needs more cooperative neighbors than it would need if the job would be smaller. As each agent has a single skill out of a set of five possible skills the probability of processing a job without help is $\left(\frac{1}{5}\right)^{|j|}$. Additionally, large numbers of propositions make it hard to complete the jobs. To lower the pressure from the number of propositions we decided to have $|\mathcal{P}| = 5$ propositions in these experiments.

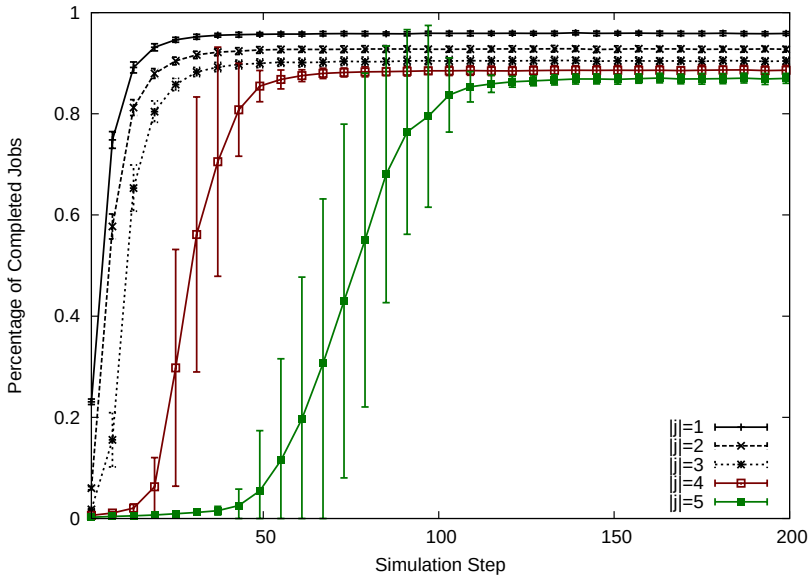


Figure 5.35: Percentages of completed jobs for different job complexities.

Figures 5.35–5.37 show the results for the five different job complexities. As the job complexity increases, the percentage of completed jobs decreases (cf. Figure 5.35). In addition, the speed of convergence gets lower. As can be seen, the percentage of completed jobs differs significantly for all five settings. Different to this is the local cooperation willingness as shown in Figure 5.36. In all settings the same high level of local cooperation willingness is reached. Only the speed of development gets lower for greater job sizes. The same holds for the global cooperation willingness. Although nearly 100% of the agents are willing to cooperate with their neighbors, some agents are not willing to cooperate with arbitrary agents as shown in Figure 5.37.

We also simulated random job sizes uniformly distributed between $t_{\min} = 1$ and $t_{\max} = 5$. As expected, the development is nearly the same as if each job contains exactly three tasks (cf.

Figure 5.38). To illustrate the influence of the number of propositions we have also simulated the same settings with $|P| = 10$ propositions. As Figure 5.39 shows, only settings with job complexities of $|j| \leq 3$ reach high job completion rates. This is due to the fact that the pressure from the propositions is too high and nearly no cooperation takes place leading to agents having all very low utility values and, therefore, no agent adapts at all.

As a conclusion, we have seen that the agents are able to process jobs with high complexity and reach high job completion rates. However, the job's complexity has to be chosen carefully, as the number of propositions also has large influence on this part. For large sets of propositions only up to three tasks per job can be processed by the agents.

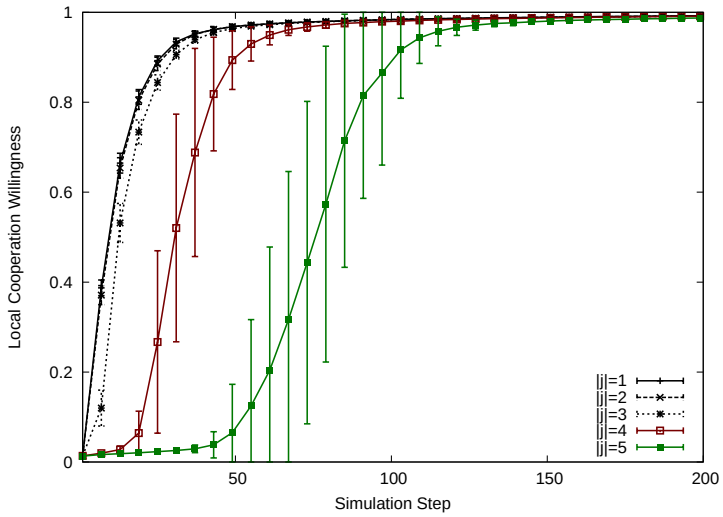


Figure 5.36: Local cooperation willingness for different job complexities.

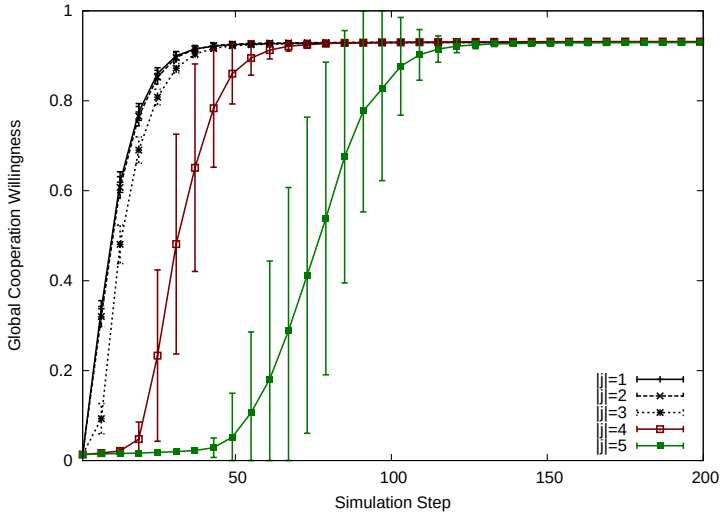


Figure 5.37: Global cooperation willingness for different job complexities.

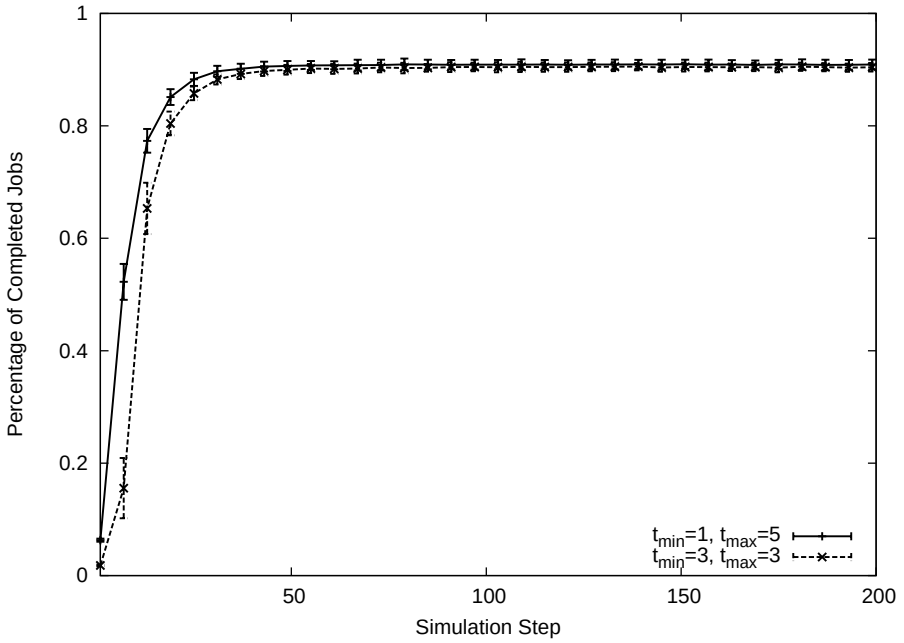


Figure 5.38: Percentages of completed jobs with random number of tasks per job.

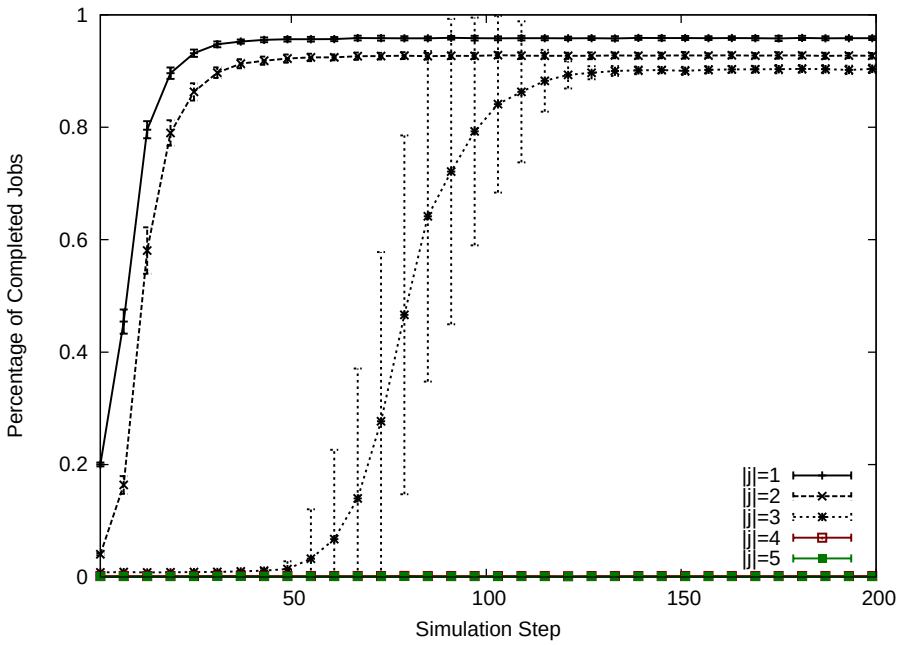


Figure 5.39: Percentages of completed jobs for different job complexities and 10 propositions.

5.11 Size of Skill Sets

The influence of different skill set sizes is examined in this section. In the system there are two different skill sets. The first is part of the agents and models the agents abilities, i.e. $\mathcal{S}_a$. In previous experiments each agent was only able to perform tasks that require a single specific skill. Now we select the number of skills per agent from $|\mathcal{S}_a| \in \{\underline{1}, 2, 3\}$. Obviously, this number depends on the number of possible skills in the system. Because of this, we take the number of possible skills—which is the second skill set in the system—from $|\mathcal{S}| \in \{3, \underline{5}, 7, 10, 12, 15\}$. Both, $|\mathcal{S}_a| = 1$ and $|\mathcal{S}| = 5$ are part of the base configuration.

To analyze the influence of the skill set sizes we concentrate on the percentage of completed jobs. The results are presented in Figures 5.40–5.42. Figure 5.40 shows the results for agents containing a single skill and different numbers of possible skills in the system. As can be observed, the percentage of completed jobs is inversely proportional to the possible number of skills. The best results are obtained if only a few skills are possible. For $|\mathcal{S}| = 3$ the system develops very fast and reaches around 98% of completed jobs. For moderate sized skill sets ($|\mathcal{S}| = 5$) the development is significantly slower and only about 90% of all jobs can be completed in late simulation steps. For larger skill sets nearly no job can be completed. Larger system skill sets are only possible if the number of skills per agent is increased. For two skills per agent it can be observed that only for $|\mathcal{S}| = 12$ and $|\mathcal{S}| = 15$ the system does not complete any multi-skill job. Only if three skills per agent are allowed, we can obtain high job completion rates for all simulated numbers of possible skills. This leads to the hypothesis that for high job completion rates it has to hold that:

$$\frac{|\mathcal{S}_a|}{|\mathcal{S}|} \geq 0.2 \tag{5.3}$$

which means that the number of possible skills must not be larger than five times the number of skills per agent.

To underline this hypothesis, we experimented with ten possible skills in the system and growing numbers of skills per agent. The results are given in Figure 5.43. It can be observed that for simulations where the number of possible skills is ten times the number of skills per agent, i.e. $|\mathcal{S}_a| = 1$ and $|\mathcal{S}| = 10$, no job can be completed. If the agents have 20% or 33% of the possible skills they are able to fulfill high numbers of jobs at the end of the simulation.

We conclude that the number of skills has significant influence on the system's performance. We claim that each agent should be endowed with 20% of the possible skills to reach high levels of completed jobs. It remains open how the number of neighbors and the jobs sizes influence this hypothesis. This will be analyzed in later sections.

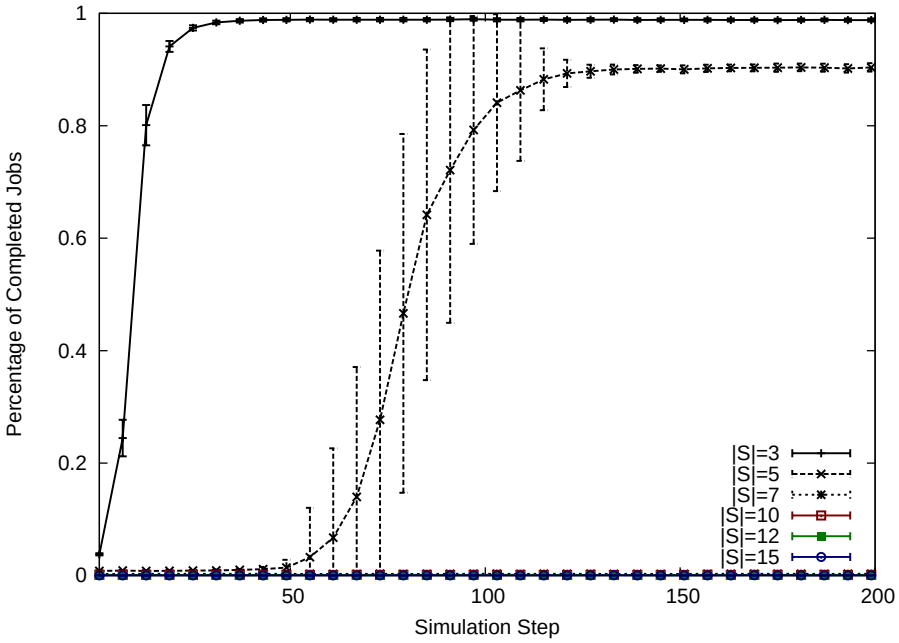


Figure 5.40: Percentages of completed jobs for different system skill set sizes and one skill per agent.

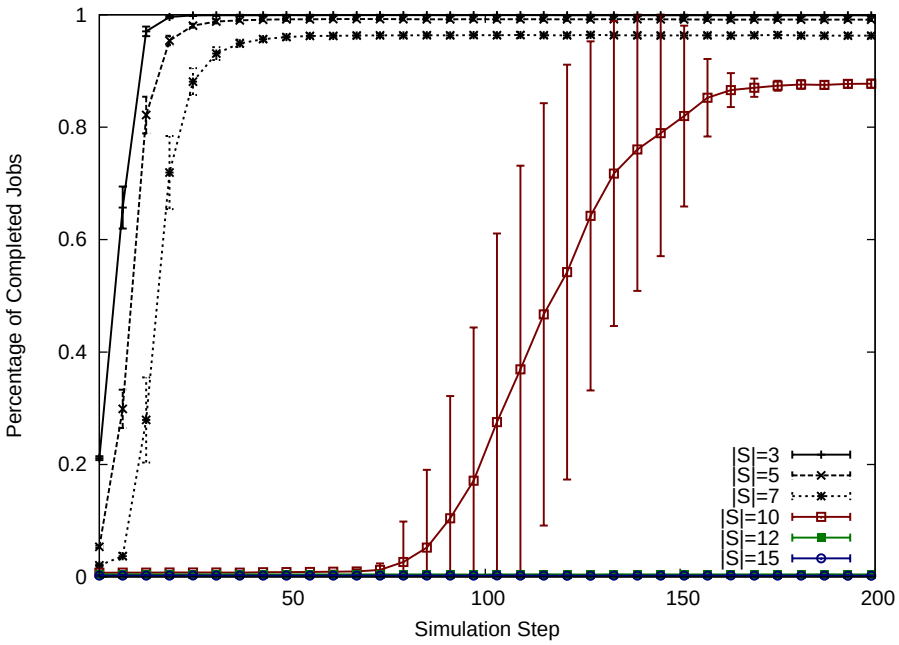


Figure 5.41: Percentages of completed jobs for different system skill set sizes and two skills per agent.

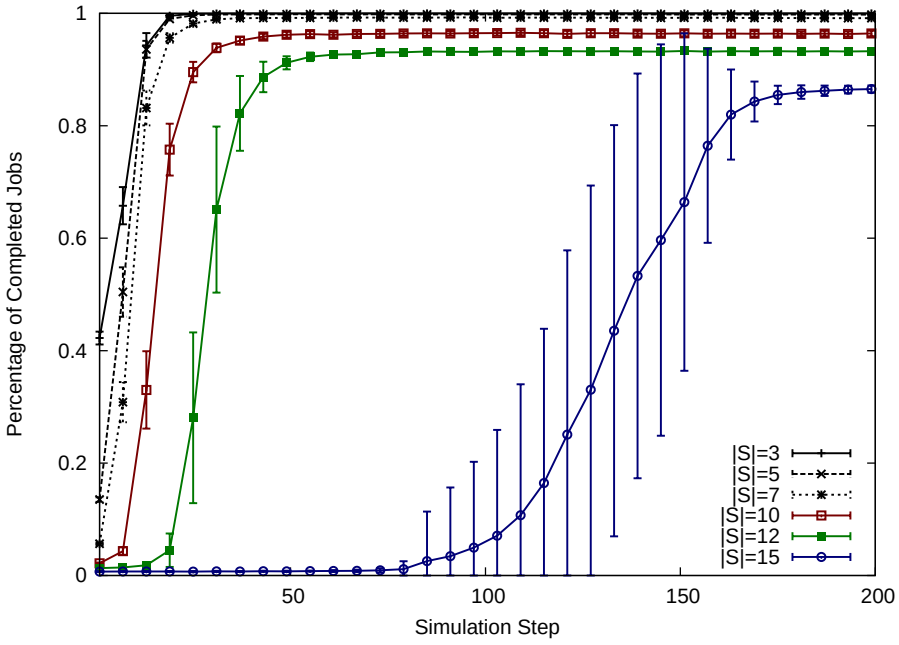


Figure 5.42: Percentages of completed jobs for different system skill set sizes and three skills per agent.

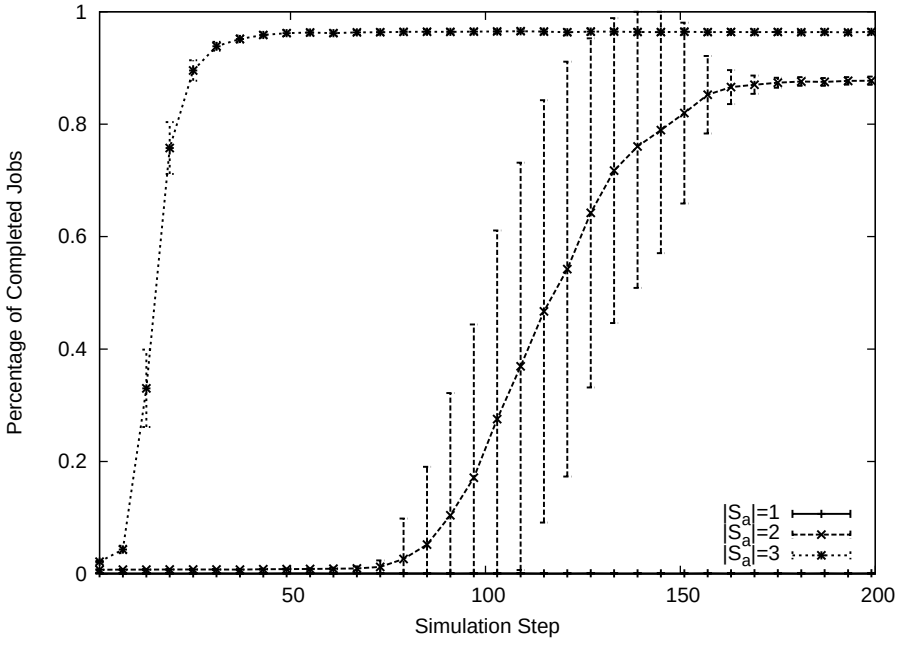


Figure 5.43: Percentages of completed jobs for different number of skills per agent and ten possible skills.

5.12 Influence of the Job Factor

The influence of the job factor is analyzed in this section. Therefore, we have chosen the job factor k from

$$k \in \{1, 3, 7, \underline{10}, 13, 15, 17, 20\}$$

where $k = 10$ is the value of the base configuration.

The results are given in Figures 5.44–5.46. As expected, the job factor has large influence on the system's performance. The job factor determines the number of generated jobs and, therefore, determines the number of interactions between the agents. At the beginning the proposition values and the threshold vectors are initialized uniformly at random. Cooperation in early simulation steps is very rare, as the values for the propositions do not match. Since the number of interactions increases the agents may process jobs only by coincidence. That is why a number of interactions is needed to have at least some cooperative and successful interactions. Only if agents are able to process some of their jobs, they may be taken as an ideal agent. If none of the jobs is processed, every agent is satisfied, as agents will have the same payoff as their neighbors.

Figure 5.44 shows the percentage of completed jobs. For all $k \geq 10$ the system reaches the same percentage of completed jobs within the considered 200 simulation steps. Only the speed of development differs. As the number of jobs increases, high levels of completed jobs are reached earlier. For all $k < 10$ there is no development at all. The same behavior can be seen if we consider the local and global cooperation willingness which are given in Figure 5.45 and Figure 5.46, respectively. This leads to the conclusion that for smaller k an insufficient number of successful interactions take place. Therefore, all agents are satisfied and no agent adapts.

As there is large difference between $k = 7$, which leads to no development, and $k = 10$, which results in high levels of cooperation, we performed additional experiments with $k \in \{7, 8, 9, 10\}$ to determine the minimal job factor for which the system shows cooperative behavior. The result is shown in Figure 5.47. For $k > 7$ we see that the system reaches cooperative behavior. For $k = 9$ the same high level of completed jobs is reached as in the case of $k = 10$ but, again, the convergence speed is much slower. The speed gets even slower for $k = 8$ but we can see that it will probably reach the same level as the curve is still growing.

We can conclude that the job factor has great influence on the system's performance as it determines the number of interactions between the agents. At least an average workload of 8 jobs per agent is needed to achieve cooperative behavior and even a workload of 9 jobs per agent is needed to achieve high cooperative behavior within the time window of 200 simulation steps. Therefore, we will always choose $k = 10$ in the other experiments as it was assumed in the base configuration.

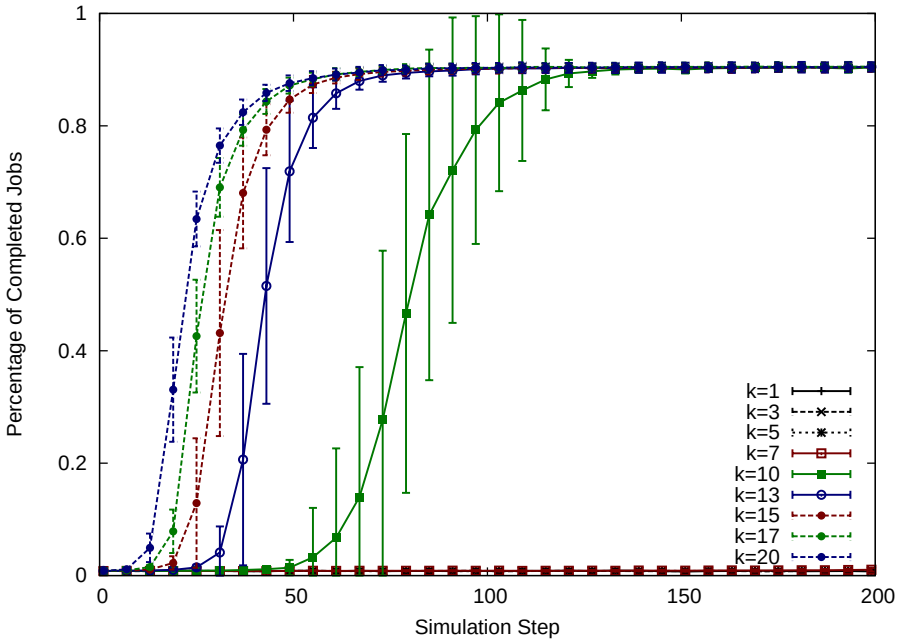


Figure 5.44: Percentages of completed jobs for variations of k.

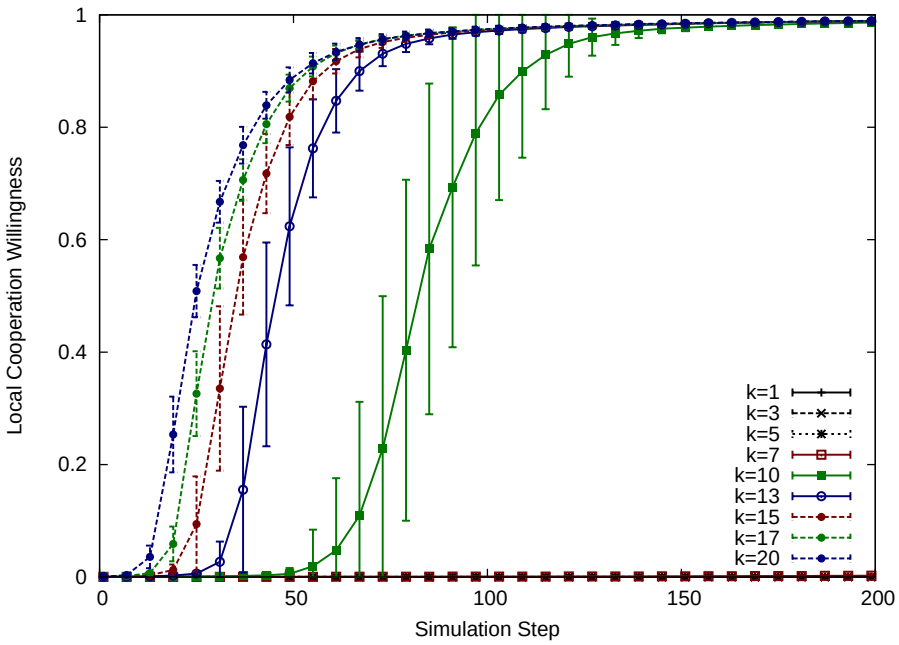


Figure 5.45: Local cooperation willingness for variations of k.

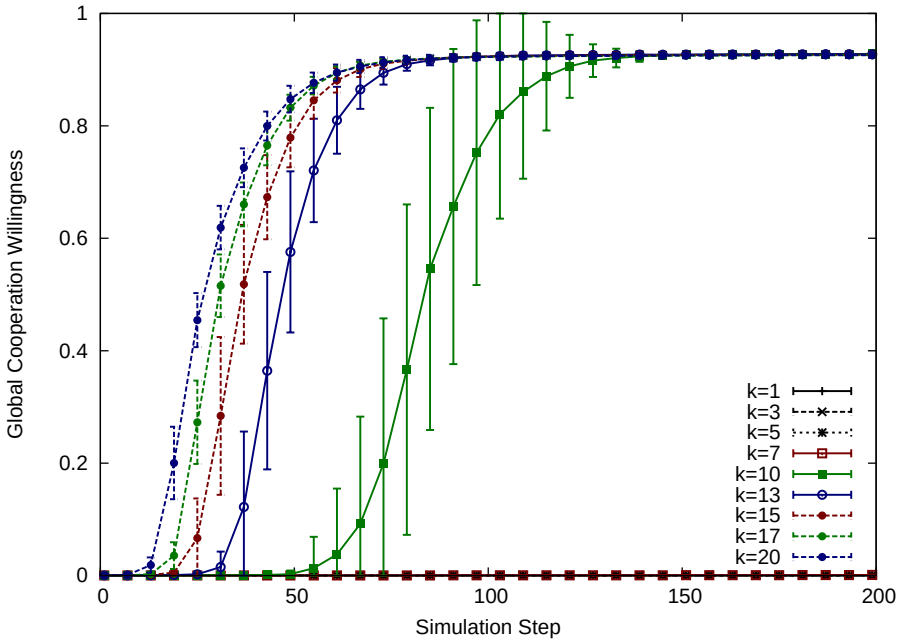


Figure 5.46: Global cooperation willingness for variations of k.

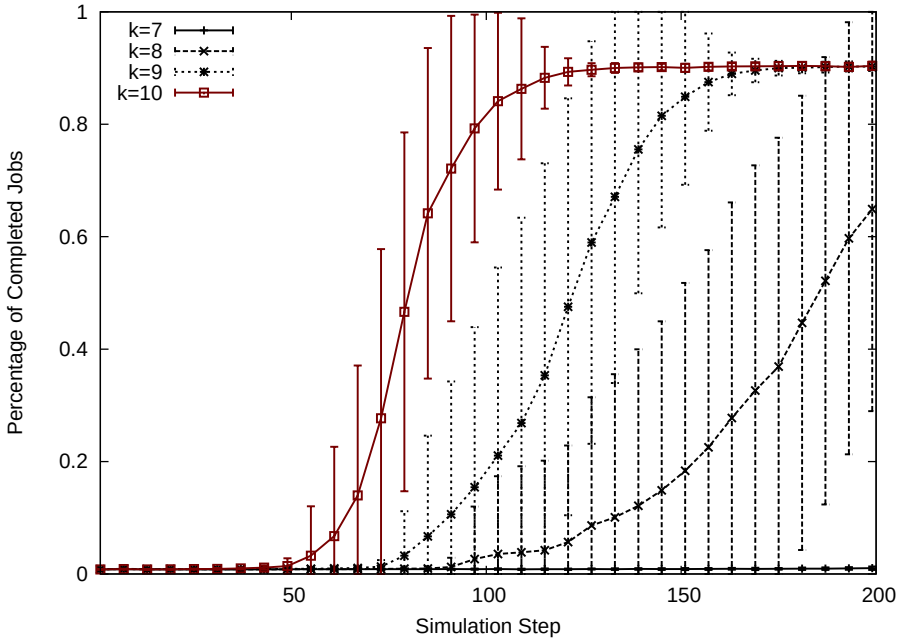


Figure 5.47: Percentages of completed jobs for variations of k between 7 and 10.

5.13 Interdependencies between Selected Parameters

To analyze the interdependencies between some special parameters, we used a 2^k factorial design experiment (Jain, 1991). In such an experiment one has to specify a *low* and a *high* value for each parameter and then all combinations are considered in simulations. We investigated the influence of the neighborhood sizes $|\mathcal{N}_a|$, the ratio between the number of skills per agent to the number of skills in the system $\frac{|\mathcal{S}_a|}{|\mathcal{S}|}$, the job complexity $|j|$ and the number of different propositions $|\mathcal{P}|$. With the factorial design method we are able to identify on the one hand the single influence of a factor and on the other hand we are able to quantify the influence of combinations of these factors.

Table 5.2: Setting for the factorial design analysis.

	***low* value**	***high* value**
$\|\mathcal{N}_a\|$	5	35
r	1	7
$\|\mathcal{S}\|$	2	20
$\|j\|$	1	5
$\|\mathcal{P}\|$	1	20

Table 5.2 gives the setting for the 2^4-factorial design analysis for the four parameters under consideration. We simulated the system with 1000 agents over 200 simulation steps with 30 independent runs and analyzed the average system's performance in the last time step. For the number of neighbors $|\mathcal{N}_a|$ of each agent, we selected a low value of 5 and a high value of 35. The interaction network was an Erdõs-Rényi random network with a connection probability of 0.005 and 0.035 respectively. As we have dynamics in the model, each agent may change r neighbors. To get rid of this influence without losing the dynamics, we fixed the value of changeable neighbors to 20% of the neighbors. Therefore, we have different low and high values for this parameter although we do not consider this parameter in the factorial design analysis. Each agent in the system is endowed with a single skill. That is why, we only have to vary the number of skills in the system to analyze different ratios between the number of skills per agent and the possible number of skills in the system. For this experiment we have chosen 2 as the low value and 20 as the high value for the number of skills $|\mathcal{S}|$. Note, that a value of 2 leads to a high value for the ratio whereas a value of 20 leads to a low value for the fraction. For the job complexity we have chosen 1 as the low value and 5 as the high value. Finally, we have used 1 for the low value of the number of propositions and 20 as the high value. The result of this experiment is shown in Table 5.3.

As can be seen in Table 5.3, the most influential factor is the ratio of the skill sets. It has an effect of 40.82% on the system's performance. The second most influential factor is the size of

Table 5.3: Results of the factorial design. Each value is approximated with an accuracy of 10^{-2}.

		$\lvert\mathcal{N}_a\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert j\rvert$	$\lvert\mathcal{P}\rvert$											
Exp	I	A	B	C	D	AB	AC	AD	BC	BD	CD	ABC	ABD	ACD	BCD	$ABCD$	Result
1	+1	−1	−1	−1	−1	+1	+1	+1	+1	+1	+1	−1	−1	−1	−1	+1	0.89
2	+1	−1	−1	−1	+1	+1	+1	−1	+1	−1	−1	−1	+1	+1	+1	−1	0.72
3	+1	−1	−1	+1	−1	+1	−1	+1	−1	+1	−1	+1	−1	+1	+1	−1	0.77
4	+1	−1	−1	+1	+1	+1	−1	−1	−1	−1	+1	+1	+1	−1	−1	+1	0.13
5	+1	−1	+1	−1	−1	−1	+1	1	−1	−1	+1	+1	+1	−1	+1	−1	0.21
6	+1	−1	+1	−1	+1	−1	+1	−1	−1	+1	−1	+1	−1	+1	−1	+1	0.05
7	+1	−1	+1	+1	−1	−1	−1	+1	+1	−1	−1	−1	+1	+1	−1	+1	0
8	+1	−1	+1	+1	+1	−1	−1	−1	+1	+1	+1	−1	−1	−1	+1	−1	0
9	+1	+1	−1	−1	−1	−1	−1	−1	+1	+1	+1	+1	+1	+1	−1	−1	1
10	+1	+1	−1	−1	+1	−1	−1	+1	+1	−1	−1	+1	−1	−1	+1	+1	1
11	+1	+1	−1	+1	−1	−1	+1	−1	−1	+1	−1	−1	+1	−1	+1	+1	1
12	+1	+1	−1	+1	+1	−1	+1	+1	−1	−1	+1	−1	−1	+1	−1	−1	1
13	+1	+1	+1	−1	−1	+1	−1	−1	−1	−1	+1	−1	−1	+1	+1	+1	0.84
14	+1	+1	+1	−1	+1	+1	−1	+1	−1	+1	−1	−1	+1	−1	−1	−1	0.83
15	+1	+1	+1	+1	−1	+1	+1	−1	+1	−1	−1	+1	−1	−1	−1	−1	0.44
16	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	0
	8.88	3.34	−4.13	−2.2	−1.4	0.35	−0.25	0.52	−0.77	0.2	−0.76	−1.67	−1.09	−0.12	0.2	−1.08	Total
	0.56	0.21	−0.26	−0.14	−0.09	0.02	−0.02	0.03	−0.05	0.01	−0.05	−0.1	−0.07	−0.01	0.01	−0.07	Total/16
Sq.		0.05	0.07	0.02	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.01	0.0	0.0	0.0	
$16 \cdot$ Sq.		0.7	1.07	0.30	0.12	0.01	0.0	0.02	0.04	0.0	0.04	0.17	0.07	0.0	0.0	0.07	$\sum = \mathbf{2.62}$
effect		**26.6%**	**40.82%**	**11.52%**	**4.72%**	**0.28%**	**0.14%**	**0.64%**	**1.43%**	**0.09%**	**1.39%**	**6.65%**	**2.83%**	**0.03%**	**0.09%**	**2.78%**	

the neighborhoods. It has an effect of 26.6% on the system's performance. If the number of skills increases the system's performance has to get lower under the assumption that the neighborhood sizes are constant. Assume each agent having a neighborhood size of 15 and each agent is equipped with a single skill. Then, at most 16 skills can be in the system because each agent has at most 16 skills in its vicinity as the agent has one skill and 15 neighbors may help, which each may have different skills. Of course the probability of having all 16 skills in the vicinity is very low. It is obvious that if the neighborhoods grow, more skills may be in the system or in other words the ratio between the skill set sizes may be lower. Interestingly, this is not reflected in the interaction between these two factors as it can be seen in the AB column in Table 5.3. Both factors—the neighborhood size $|\mathcal{N}_a|$ and the number of skills in the system $|\mathcal{S}|$—have an influence of 0.28% on the systems performance, although the interaction between both factors should be very high. To analyze this special interaction we did experiments with growing skill sets and growing neighborhood sizes, which is presented in the next section.

5.14 Combined Influence of Neighborhoods and Skills

In this section, the combined influence of the neighborhood size and the skill set sizes is presented. The average number of neighbors per agent $|\mathcal{N}_a|$ is selected from the set $\{15, 20, 25, 30, 35, 40, 45, 50\}$ and for the number of skills in the system $|\mathcal{S}|$ a value from the set $\{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ is selected. We present the percentage of completed jobs in the 200th simulation step to show which level of completed jobs is achievable with the given settings. Again, the presented values are average values over 30 independent simulations. As the influence of the number of propositions in the system $|\mathcal{P}|$ cannot be neglected we will present the results for a small set of propositions with $|\mathcal{P}| = 5$ and for a medium size set with $|\mathcal{P}| = 10$. Figure 5.48 and Figure 5.49 illustrates the results for five and ten propositions, respectively.

It can be seen for small sets of propositions that for most settings quite high values of completed jobs can be achieved (cf. Figure 5.48). However, with increasing skill set sizes we can observe that larger neighborhoods are required to compensate the pressure of more ambivalent tasks. Note that we only need 50 neighbors if 15 skills are possible in the system in order to achieve a completion rate larger than 90%. The theoretical probability of having all skills in the vicinity of the agent if there are 15 skills in the system and the agent has 50 neighbors is approximately 62% (cf. Section 4.1). Thus, the mechanisms *social networking* and the adaptation phase lead to a system that reaches higher percentages of completed jobs than in the theoretical case. In systems with larger sets of propositions, we see a drastic decrease in the percentage of completed jobs (cf. Figure 5.48). With $|\mathcal{P}| = 10$ a much larger number of neighbors is needed. This is due to the fact that the probability of having cooperative neighbors gets lower with a larger number of constraints that have to be fulfilled. If the pressure on the agents is too high because of missing cooperative partners nearly none of the agents get unsatisfied and, therefore, no adaptation takes place. This results in the observed behavior.

We conclude that the number of neighbors per agent and the number of skills in the system

depend on each other and need to be chosen carefully.

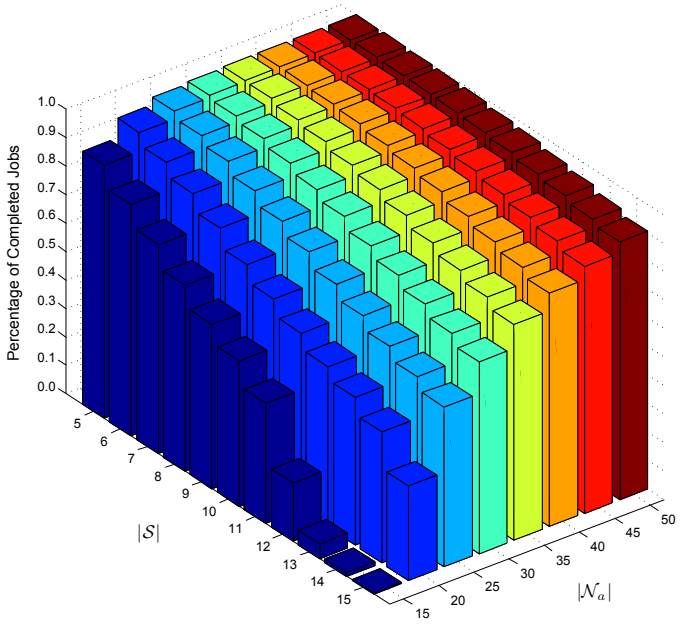


Figure 5.48: Percentage of completed jobs for combined variations of the number of skills in the system and the average number of neighbors per agent with 5 propositions.

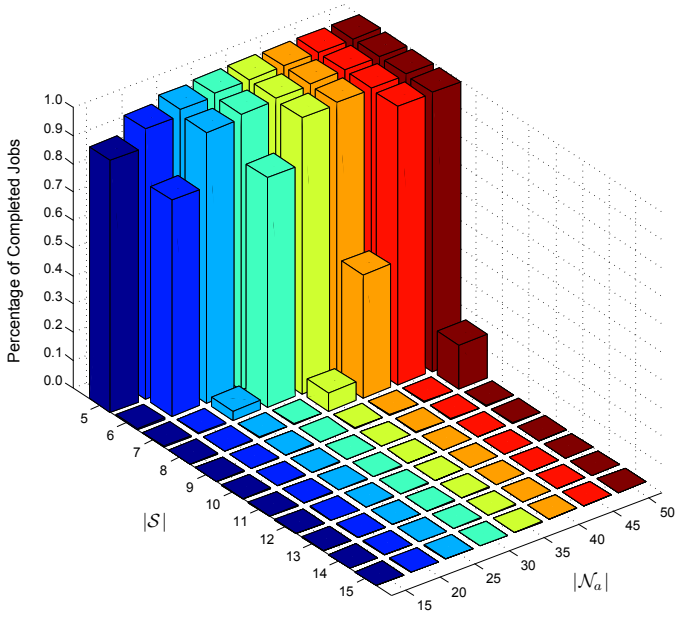
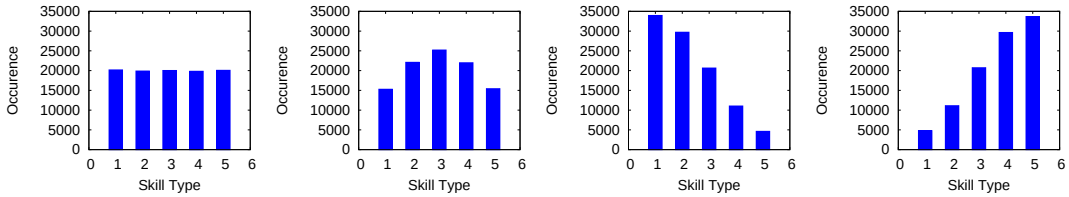


Figure 5.49: Percentage of completed jobs for combined variations of the number of skills in the system and the average number of neighbors per agent with 10 propositions.

5.15 Different Random Distributions

In this section, we analyze the influence of different random distributions on the system's performance. We will only concentrate on random distributions for agents' skills and tasks' skills. For the agents' skills we will assume that they are static and do not change over time. This is a reasonable assumption in our scenario, as the agents do not learn the skills of their neighbors.



(a) $\mathcal{U}(1,5)$ (b) $\mathcal{N}(3;2)$ (c) $\mathcal{N}(1;2)$ (d) $\mathcal{N}(5;2)$

Figure 5.50: Resulting occurrence of the skill types with the given random distributions of 100000 random values.

In previous experiments, we only considered a uniform distribution $\mathcal{U}(1,5)$ for the agent skills and the tasks skills. Now we want to investigate the impact of different distributions. First, we will consider normal distributions with different expected values. Note that normal distributions $\mathcal{N}(\mu,\sigma)$ are continuous by nature but we need a discrete distribution for the skills. Therefore, we decided to round the produced random value to the nearest natural number that represents a skill. If the result is not within the boundaries 1 and 5—like in previous experiments we consider a set of possible skills with cardinality five—we just take another random value until the rounded value is within the boundaries. We decided to take for both random variables the uniform distribution $\mathcal{U}(1,5)$ and the normal distributions $\mathcal{N}(3,2)$, $\mathcal{N}(1,2)$ and $\mathcal{N}(5,2)$ and to consider all combinations in the experiments in this section. To visualize the random distributions we have generated 100000 random values for the skill value between 1 and 5. The resulting distributions can be seen in Figure 5.50. Table 5.4 gives the overview on the experimental setup. The results of the experiments are given in Figures 5.51–5.54.

Figure 5.51 gives the percentage of completed jobs where each agent has a randomly selected skill with uniform distribution $\mathcal{U}(1,5)$ and each task of a job gets a randomly selected skill with different random distributions. The standard case of both distributions being uniformly distributed is given by the solid black line. As can be seen we obtain the same results we have observed in earlier experiments. If the skill distribution of the tasks is normally distributed following $\mathcal{N}(3;2)$ the results become better as the high level of completed jobs is reached earlier. The experiments with two extreme forms of the normal distribution, i.e. $\mathcal{N}(1;2)$ and $\mathcal{N}(5;2)$, lead to a drastic speedup for the development. This can be interpreted as the result of easier jobs. In the case of $\mathcal{N}(1;2)$ roughly 85% of all skills are of the first three types and in the case of $\mathcal{N}(5;2)$ the same number of generated skills is of the last three types. This results in jobs that are easier to process in comparison to the uniformly distributed case.

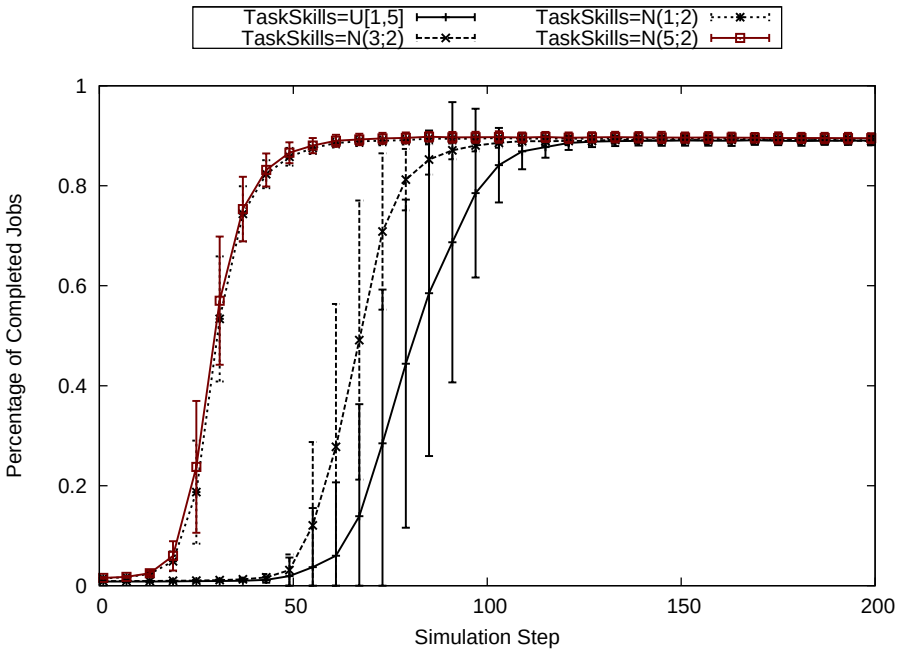


Figure 5.51: Percentage of completed jobs with uniformly distributed agent skills and varying distributions of the tasks.

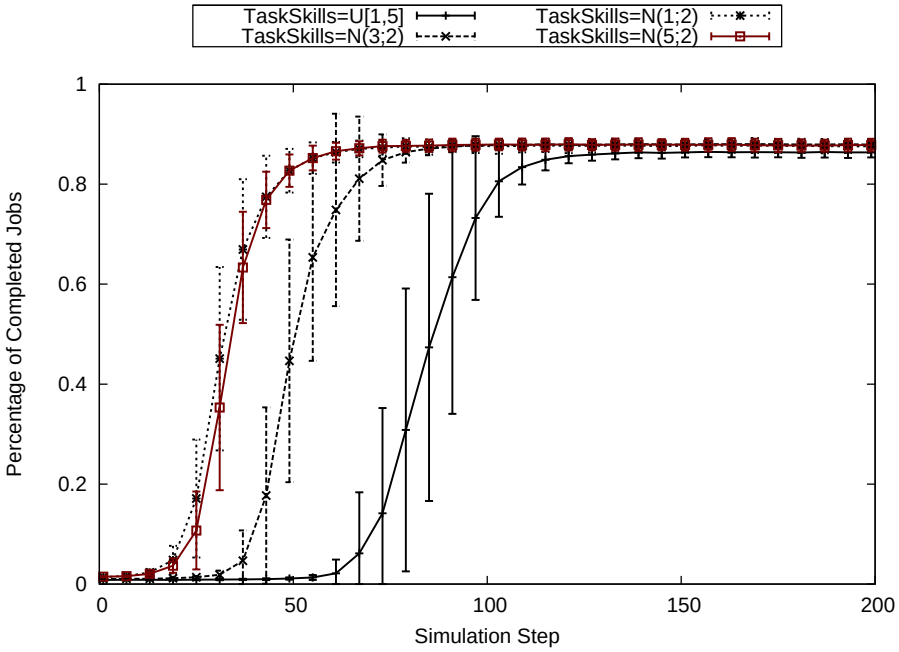


Figure 5.52: Percentage of completed jobs with normal distributed agent skills following $\mathcal{N}(3;2)$ and varying distributions of the tasks.

Table 5.4: Experimental setup for random distributions.

Distribution of agents' skills	Distribution of tasks' skills
$\mathcal{U}(1,5)$	$\mathcal{U}(1,5)$
$\mathcal{U}(1,5)$	$\mathcal{N}(3;2)$
$\mathcal{U}(1,5)$	$\mathcal{N}(1;2)$
$\mathcal{U}(1,5)$	$\mathcal{N}(5;2)$
$\mathcal{N}(3;2)$	$\mathcal{U}(1,5)$
$\mathcal{N}(3;2)$	$\mathcal{N}(3;2)$
$\mathcal{N}(3;2)$	$\mathcal{N}(1;2)$
$\mathcal{N}(3;2)$	$\mathcal{N}(5;2)$
$\mathcal{N}(1;2)$	$\mathcal{U}(1,5)$
$\mathcal{N}(1;2)$	$\mathcal{N}(3;2)$
$\mathcal{N}(1;2)$	$\mathcal{N}(1;2)$
$\mathcal{N}(1;2)$	$\mathcal{N}(5;2)$
$\mathcal{N}(5;2)$	$\mathcal{U}(1,5)$
$\mathcal{N}(5;2)$	$\mathcal{N}(3;2)$
$\mathcal{N}(5;2)$	$\mathcal{N}(1;2)$
$\mathcal{N}(5;2)$	$\mathcal{N}(5;2)$

Figure 5.52 gives the percentage of completed jobs where each agent has a randomly selected skill with normal distribution $\mathcal{N}(3;2)$ and each task of a job gets a randomly selected skill with different random distributions. As can be seen the results do not differ significantly from the previously presented results where the agent skills were drawn uniformly at random. This is due to the fact that the occurrences of skills following $\mathcal{N}(3;2)$ and $\mathcal{U}(1,5)$ are quite similar (cf. Figure 5.50a and Figure 5.50b). Although the three skills $2, 3$ and 4 are the mostly generated skills the skills 1 and 5 each are still generated with probability 15%. Nevertheless, we can identify an advantage if the task skills are also normally distributed with $\mathcal{N}(3;2)$. The reason is that the distributions match better than the normal distribution $\mathcal{N}(3;2)$ and the uniform distribution $\mathcal{U}(1,5)$. Then the skills generated for the task fit better to the skills of the agents.

If the agent skills are randomly drawn with either $\mathcal{N}(1,2)$ or $\mathcal{N}(5,2)$ the results look very similar. Only the roles of the extreme distributions for the tasks' skills change (cf. Figure 5.53 and Figure 5.54). In both scenarios, it can be observed that the maximum job completion rate is lower than in the experiments before. Only in the case where the two distributions are identical

we can observe the high completion rate of earlier experiments.

As a conclusion we can say that if the overall occurrence of each skill—either for skills or for tasks—is not lower than in the uniform case we do not observe a difference in the reachable level of completed jobs. Only the convergence speed differs. However, if the normal distributions are somehow converse to each other the system does not reach a high level of completed jobs.

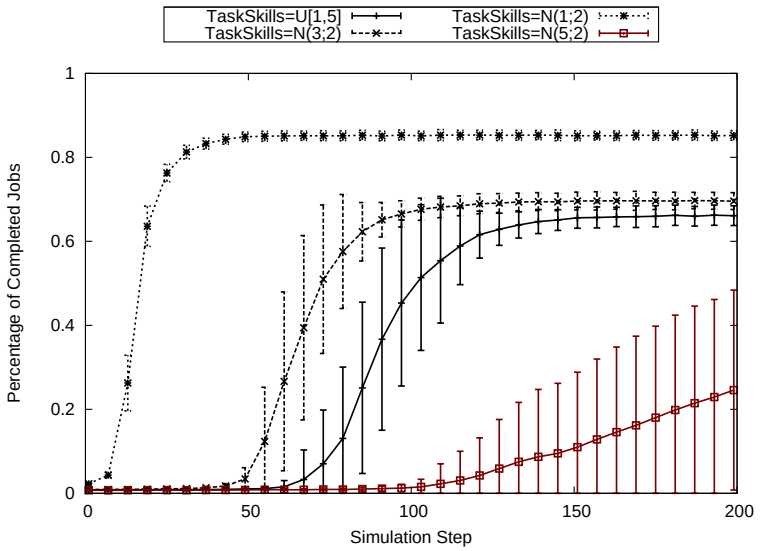


Figure 5.53: Percentage of completed jobs with normal distributed agent skills following $\mathcal{N}(1; 2)$ and varying distributions of the tasks.

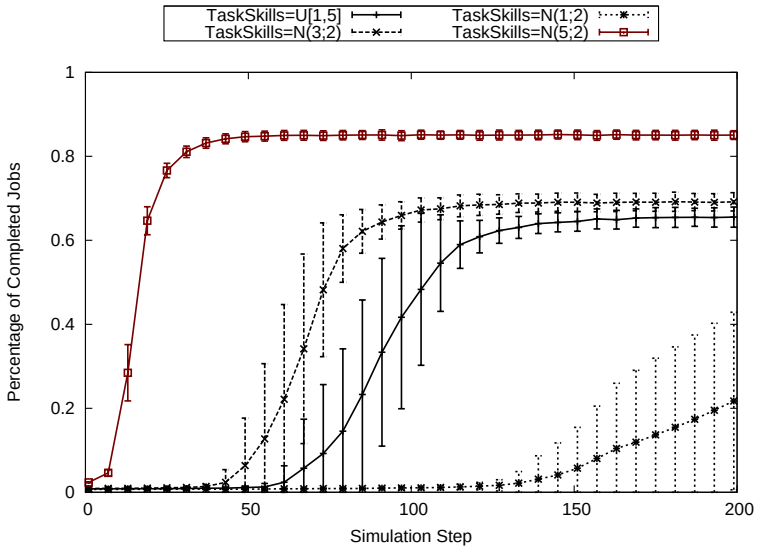


Figure 5.54: Percentage of completed jobs with normal distributed agent skills following $\mathcal{N}(5; 2)$ and varying distributions of the tasks.

5.16 Number of Agents

The scalability of the presented approach is experimentally analyzed in this section. As the average neighborhood size is determined through the number of agents and the connection probability, we have to chose the connection probability $\mathsf{Pr}_{\mathsf{con}}$ for the Erdõs-Rényi random network with respect to the number of agents in the system $|\mathcal{A}|$. Thus, we will experiment with the experimental setup given in Table 5.5.

Table 5.5: Experimental setup for the scalability analysis.

Exp.	$\lvert\mathcal{A}\rvert$	$\mathsf{Pr}_{\mathsf{con}}$
1	100	0.15
2	500	0.03
3	1000	0.015
4	2000	0.0075
5	5000	0.003
6	10000	0.0015

Obviously, the average number of neighbors is 15 in all experiments and, thus, the comparability is ensured. The results are presented in Figures 5.55–5.58.

As can be seen in Figure 5.55, for all numbers of agents the same high level of completed jobs is reached. As the standard deviations are quite large—especially for the case with $|\mathcal{A}| = 100$ agents—we also present the same results without the plotted standard deviations in Figure 5.56. The only difference between the experiments is the development of the completion rate over time. In the beginning the smaller populations start to complete more jobs faster but then the experiments with greater populations outperform the smaller population experiments. Most significantly, this is the case for the smallest population with $|\mathcal{A}| = 100$. If we compare the experiments with $|\mathcal{A}| = 500$ and $|\mathcal{A}| = 1000$ we can see that they start with completing more jobs nearly in identical way but the development for $|\mathcal{A}| = 500$ gets slower at the end. These developments are also reflected in the local and global cooperation willingness (cf. Figure 5.57 and Figure 5.58 resp.).

To conclude, we claim that the proposed algorithm is scalable and robust against different population sizes. Although the population size has slightly influence on the shape of the development we cannot identify significantly differences in the reachable percentage of completed jobs.

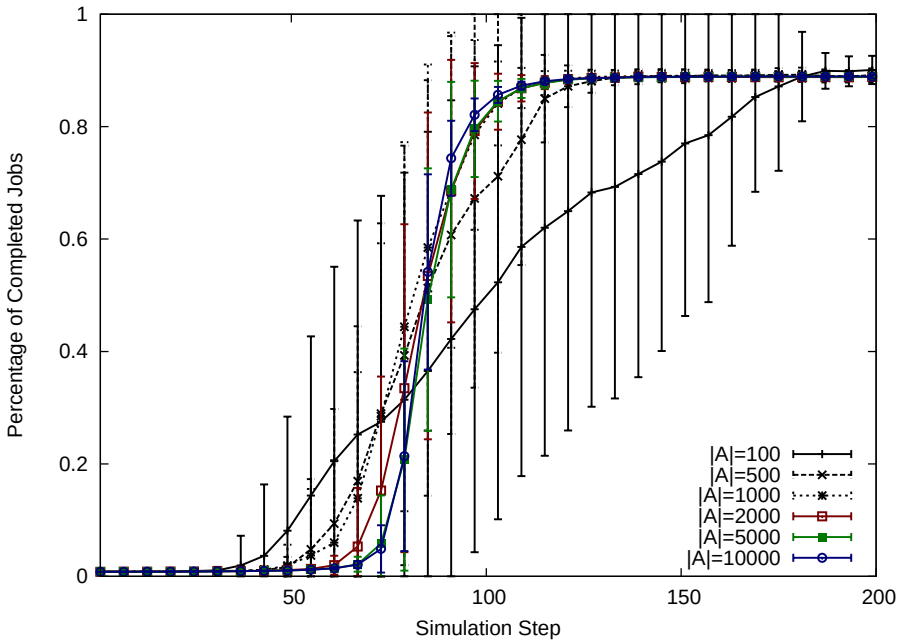


Figure 5.55: Percentages of completed jobs for different numbers of agents.

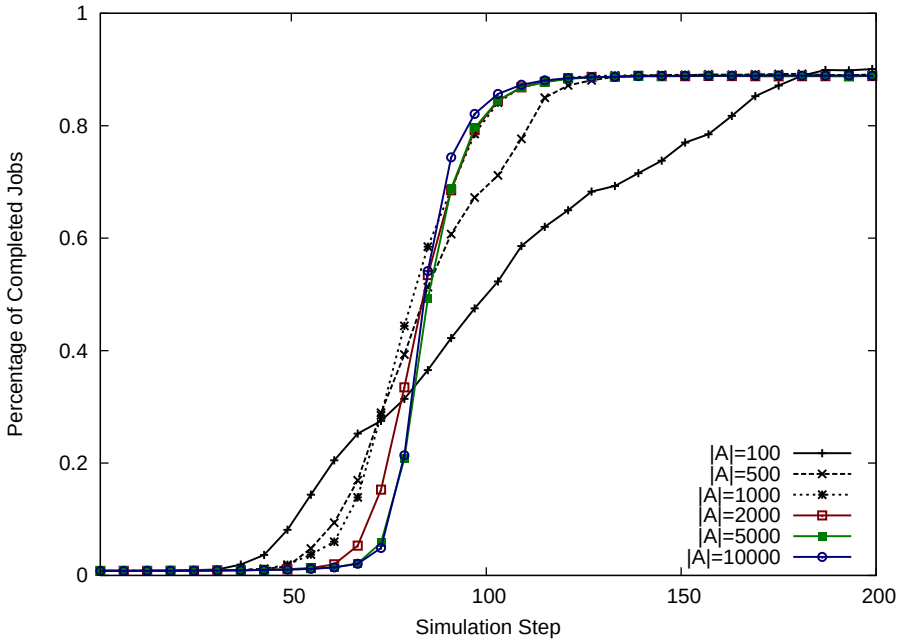


Figure 5.56: Percentages of completed jobs for different numbers of agents without standard deviations.

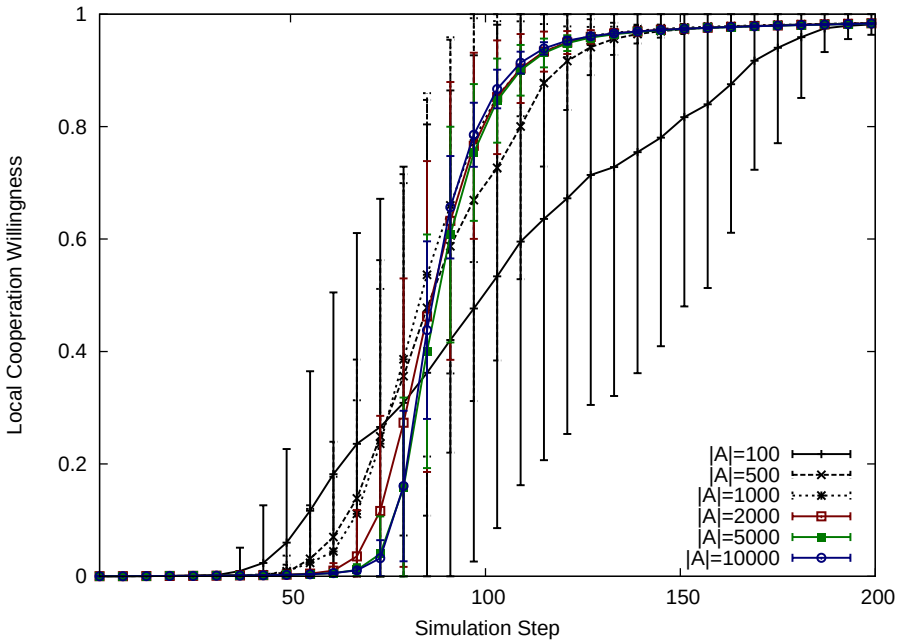


Figure 5.57: Local cooperation willingness for different numbers of agents.

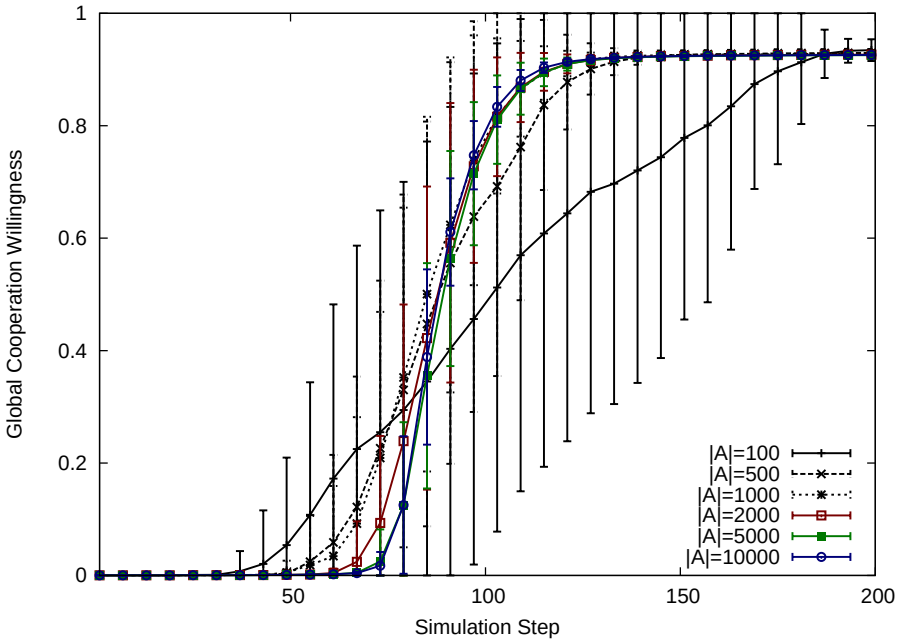


Figure 5.58: Global cooperation willingness for different numbers of agents.

5.17 Conclusion

Let us briefly recall the main conclusions of this chapter's sections:

- Section 5.2: cooperation costs have no influence
- Section 5.3: adaptation strength is important and a trade-off between no-adaptation and value copying has to be chosen
- Section 5.4: larger elite sets lead to slightly lower cooperation
- Section 5.5: large influence of the number of criteria for the cooperation decision concerning speed of development
- Section 5.6: large influence of the agents tolerance
- Section 5.7: one fixed value for the adaptation strength performs best
- Section 5.8: selecting the single best neighbor as role model performs best but if it is not computable, a set of two randomly chosen neighbors can compete with this strategy
- Section 5.9: large influence of the neighborhood sizes
- Section 5.10: job complexity has large influence on the achievable level of completed jobs and the convergence speed
- Section 5.11: large influence of the skill set sizes
- Section 5.12: minimum number of interactions needed for development
- Section 5.13: underlined the influence of the neighborhood sizes and the skill set sizes and identified the combination of both as very influential
- Section 5.14: the average neighborhood size and the sizes of the skill sets have to be chosen carefully with respect to each other
- Section 5.15: different random distributions for agents' skills and tasks' skills have only influence on the development speed
- Section 5.16: system is shown to be highly scalable in the number of agents

To sum up the conclusions of the rigorous experimental analysis, we can say that we gained good insights from the experimental results about the influence of the system's parameters. We could see that the objectives identified in Section 1.1 have been met and that this new local adaptation mechanism really favors the decision to cooperate. It is highly scalable as the number of agents has no influence on the overall cooperation rate. The most influential parameters for the approach are the adaptation strength, the number of criteria that have to be met and the tolerance of the agents towards the difference to other agents. We could see that even for high numbers of criteria the approach results in high cooperation rates. However, too intolerant agents that only cooperate with almost identical agents could be one reason for failures of the approach.

According to our formal model, we implemented the proposed approach. It can be found online with a description for the needed configuration file(s) and all used configuration files of this and the following chapters under
`http://homepages.uni-paderborn.de/mepb2002/diss/SimDissEberling.zip`.

6 Influence of Network Structures

We deal with multiagent systems where agents form a network. As stated before, this network can be represented as a graph and, therefore, graph theory can be used for analyzing specific properties of the agent network. We start with formal definitions required to specify the network properties that are useful for our analysis and define afterwards specific network structures that are important in this thesis.

In previous chapters, we only concentrated on random networks which follow the definition of Erdõs and Rényi provided in (Erdõs and Rényi, 1959). To ease the analysis, this is a valid assumption as those random networks are easy to construct and, additionally, they provide nice properties like connectivity depending on the connection probability or a specific average node degree. However, in real applications, networks are hardly random but follow other construction laws (Albert and Barabási, 2002; Barabási and Albert, 1999; Newman, 2003). Therefore, we first give the definitions of the mostly used network types and their properties and then examine the influence of those network types on the system's performance experimentally. Section 6.1 deals with basic concepts of graph theory. Section 6.2 deals with the network structures for which the influence on the proposed learning algorithm is analyzed in this thesis. Readers familiar with these concepts and network properties can, therefore, skip these sections and continue reading in Section 6.3.

We do not intent to analyze network structures as such. What we are interested in is the influence on the cooperation that emerges between the agents due to specific network structures that impact their neighborhoods. As stated before, we use networks to specify the neighborhoods of agents and thus networks and their properties play a key role in the emerging cooperation in the proposed multiagent system.

The main results of this chapter are the following:

- We show the influence of different network structures on the system's performance.
- We show that the Erdõs-Rényi random networks show the best results in the case of static networks.
- We show that networks tend to degenerate to randomly generated networks in the dynamic case.
- For ring network structures we show that clusters of ratings emerge with high cooperation inside and low cooperation between clusters.
- For Erdõs-Rényi networks we show that the rating vectors tend to group in a small area.
- We show that clusters or groups emerge in lattice networks with high cooperation inside

and low cooperation between the groups.

6.1 Graph Theory and Important Network Measurements

This section deals with the most important definitions of graph theory and important measurements (Harris et al., 2008; Wallis, 2007) that will be used in analyzing specific network structures. As a network is a graph, we start with the formal definition of a graph:

Definition 6.1 (Graph (Wallis, 2007)): *A (un)directed Graph is a pair $G = (V, E)$, where V is a finite set of nodes and $E \subseteq V \times V$ is the set of edges for a directed graph. The set of edges in undirected graphs is slightly different defined. There, the set of edges is defined as $E \subseteq \{\{u, v\} \mid u, v \in V\}$. If the graph is directed we denote an edge $e \in E$ between two nodes i and j as a pair $e = (i, j)$. In undirected graphs the representation of the edges are sets of two nodes (i.e. $e \in E$ with $e = \{i, j\}$ for the edge between the two nodes i and j).*

Since we concentrate on interaction networks as defined in Definition 3.1, we will only deal with bidirectional graphs. Bidirectional graphs can also be viewed as undirected graphs. The degree of a node specifies how many other nodes are directly connected to this node. In the context of agent networks this means how many direct neighbors a single agent has. Therefore, we will give a brief definition on different measurable degrees that are important for our analysis:

Definition 6.2 (Node degrees and Graph degree (Wallis, 2007)): *In undirected graphs the degree of a node $v \in V$ is the number of adjacent edges of v:*

$$\mathsf{deg}(v) := |\{e \mid e \in E \wedge v \in e\}|$$

In directed graphs we distinguish between the indegree and the outdegree of a node v:

$$\mathsf{deg}_{\text{in}}(v) := |\{(i, v) \mid (i, v) \in E, i \in V\}|$$

$$\mathsf{deg}_{\text{out}}(v) := |\{(v, i) \mid (v, i) \in E, i \in V\}|$$

The degree of a node in a directed graph is the maximum of the in- and the outdegree:

$$\mathsf{deg}(v) := \max\{\mathsf{deg}_{\text{in}}(v), \mathsf{deg}_{\text{out}}(v)\}$$

The degree of a Graph $G = (V, E)$ is the maximum node degree:

$$\mathsf{deg}(G) := \max_{v \in V} \mathsf{deg}(v)$$

The average degree of a graph is the average of all node degrees:

$$\mathsf{avgdeg}(G) := \frac{\sum_{v \in V} \mathsf{deg}(v)}{|V|}$$

For many measurements it is important to consider the neighbors of a node. In literature there exist two different types of the neighborhood, namely the *open* and the *closed* neighborhood. Definition 6.3 gives the formal description of both types.

Definition 6.3 (Neighborhood of Node (Harris et al., 2008)): *Given an undirected Graph $G = (V, E)$ and a node $v \in V$. The* open *neighborhood* $\mathsf{OPN}(v)$ *of the node v is the set of all nodes that are adjacent to v, formally*

$$\mathsf{OPN}(v) := \big\{u \mid \{u, v\} \in E\big\}.$$

The closed *neighborhood* $\mathsf{CLN}(v)$ *of the node v is defined as the set of all nodes that are adjacent to v and node v itself, formally*

$$\mathsf{CLN}(v) := \big\{u \mid \{u, v\} \in E\big\} \cup \{v\}.$$

When analyzing network structures the diameter is another important measurement. It is defined as the longest shortest path between any pair of nodes. Definition 6.4 defines the diameter in a formal way.

Definition 6.4 (Diameter (Harris et al., 2008)): *Given a connected, undirected graph $G = (V, E)$. Let $L_{u,v}$ denote the length of the shortest path between the nodes u and v. Then, the diameter* $\mathsf{diam}(G)$ *of the graph is defined as*

$$\mathsf{diam}(G) := \max_{u,v \in V} L_{u,v}.$$

Two other interesting measurements are the *characteristic path length* and the *cluster coefficient*. The characteristic path length is the average distance between two nodes, where the average is taken over all possible pairs of distinct nodes. Definition 6.5 gives the formal specification of the characteristic path length.

Definition 6.5 (Characteristic Path Length (Harris et al., 2008)): *Let $L_{u,v}$ be the length of a shortest path between the nodes u and v of a connected, undirected graph $G = (V, E)$ with $|V| = n$. Then, the* characteristic path length CPL *is defined as*

$$\mathsf{CPL} := \frac{2}{n \cdot (n-1)} \sum_{u,v \in V} L_{u,v}$$

In literature two different definitions can be found for the cluster coefficient. Both definitions are important but express slightly different issues. The definition provided in (Harris et al., 2008; Newman, 2003) gives a probability of nodes' transitivity whereas the definition given in (Albert and Barabási, 2002; Wallis, 2007; Watts and Strogatz, 1998) gives a probability of nodes' "cliquishness" (Barrat and Weigt, 2000). The later definition is what we will call the cluster coefficient as our intention behind this measurement is best represented in this definition.

The other definition is also provided here but under the term graph transitivity like it is done in (Wasserman and Faust, 1994). The following two definitions give formal descriptions of both terms. In both definitions we will denote the number of edges in an induced sub-graph by the node set X as $|X|_E$. All edges which have both endpoints within the node set X are counted with this operator.

Definition 6.6 (Cluster Coefficient (Watts and Strogatz, 1998)): *The* cluster coefficient $\mathsf{cc}(v)$ of a node $v \in V$ *is defined as*

$$\mathsf{cc}(v) := \frac{2 \cdot |\mathsf{OPN}(v)|_E}{\deg(v) \cdot (\deg(v) - 1)}.$$

The cluster coefficient $\mathsf{CC}(G)$ of a graph $G = (V, E)$ *is the arithmetic mean of the cluster coefficients of all its nodes:*

$$\mathsf{CC}(G) := \frac{1}{|V|} \sum_{v \in V} \mathsf{cc}(v).$$

Note that the cluster coefficient is not well defined for $\deg(v) = 0$ and $\deg(v) = 1$. Therefore, we define $\mathsf{cc}(v) := 0$ for $\deg(v) = 0$ and $\deg(v) = 1$, which is also done in literature (Wallis, 2007).

Definition 6.7 (Transitivity (Wasserman and Faust, 1994)): *The* transitivity $\mathsf{trans}(v)$ of a node $v \in V$ *is defined as*

$$\mathsf{trans}(v) := \frac{2 \cdot |\mathsf{CLN}(v)|_E}{\deg(v) \cdot (\deg(v) + 1)}.$$

The transitivity $\mathsf{trans}(G)$ of a graph $G = (V, E)$ *is the arithmetic mean of the transitivity of all its nodes:*

$$\mathsf{trans}(G) := \frac{1}{|V|} \sum_{v \in V} \mathsf{trans}(v).$$

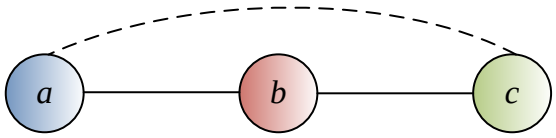


Figure 6.1: Three nodes forming a line in a graph G.

To illustrate the cluster coefficient and the transitivity let us consider the graph shown in Figure 6.1. Table 6.1 gives the cluster coefficients and the transitivities for the graph. Note that the cluster coefficient is zero for all nodes and that the transitivity is close to one. If we add one additional edge $\{a, c\}$ to the graph to the graph—represented as a dashed line in the figure—we would get a ring structure. In the considered example, this has great influence on the cluster coefficients, which are given with the transitivities in Table 6.2.

Table 6.1: Cluster coefficient and transitivity for the graph in Figure 6.1.

v	a	b	c	G
$\mathsf{cc}(v)$	0	0	0	0
$\mathsf{trans}(v)$	1	$\frac{2}{3}$	1	$\frac{8}{9}$

Table 6.2: Cluster coefficient and transitivity for the graph in Figure 6.1 with an additional edge $\{a, c\}$ forming G'.

v	a	b	c	G'
$\mathsf{cc}(v)$	1	1	1	1
$\mathsf{trans}(v)$	1	1	1	1

Transitivity and cluster coefficient may also be set into relation. It is easy to see that

$$|\mathsf{CLN}(v)|_E = |\mathsf{OPN}(v)|_E + \deg(v) \tag{6.1}$$

holds. Let us use the following simplifications to analyze the relation between these two measurements:

- $n = |\mathsf{OPN}(v)|_E$
- $d = \deg(v)$
- Let X denote the difference between transitivity and the cluster coefficient of node v.

Then, the following holds:

$$\begin{aligned} & X &=& \mathsf{trans}(v) - \mathsf{cc}(v) \\ \Rightarrow\ & X &=& \tfrac{2\cdot(n+d)}{d\cdot(d+1)} - \tfrac{2\cdot n}{d\cdot(d-1)} \\ \Leftrightarrow\ & X &=& \tfrac{2\cdot(n+d)\cdot(d-1)-2\cdot n(d+1)}{(d-1)\cdot d\cdot(d+1)} \end{aligned}$$

which leads to

$$X = \frac{2(d^2 - d - 2n)}{d^3 - d}. \tag{6.2}$$

Of course, Equation 6.2 holds only for $d > 1$ like the definition of the cluster coefficient, i.e. if and only if the node has at least two neighboring nodes.

Thus, we are able to write the relation between the cluster coefficient of a node and its transitivity:

$$\mathsf{cc}(v) + \frac{2\big(\mathsf{deg}(v)^2 - \mathsf{deg}(v) - 2\,|\mathsf{OPN}(v)|_E\big)}{\mathsf{deg}(v)^3 - \mathsf{deg}(v)} = \mathsf{trans}(v) \tag{6.3}$$

Let us now examine how close the cluster coefficient and the transitivity can be, i.e. how large the difference is in general. As it not only depends on the degree $\deg(v)$ of a node v but also on the number of edges in its open neighborhood $\mathsf{OPN}(v)$ we will first consider lower and upper bounds for the number of edges in the open neighborhood. Obviously, the lower bound for the number of edges in the open neighborhood is

$$|\mathsf{OPN}(v)|_E^{\min} = 0 \tag{6.4}$$

The upper bound is reached if the open neighborhood forms a complete graph, i.e. every distinct pair of nodes is directly connected with an edge. Therefore, the upper bound for the number of edges in the open neighborhood is

$$|\mathsf{OPN}(v)|_E^{\max} = \frac{\deg(v) \cdot (\deg(v) - 1)}{2} \tag{6.5}$$

It directly follows from Equations 6.3, 6.4 and 6.5 that $\forall v \in V$ with $\mathsf{deg}(v) \geq 2$ the following holds:

$$0 \leq \frac{2\big(\mathsf{deg}(v)^2 - \mathsf{deg}(v) - 2\,|\mathsf{OPN}(v)|_E\big)}{\mathsf{deg}(v)^3 - \mathsf{deg}(v)} \leq \frac{2}{\mathsf{deg}(v) + 1} \tag{6.6}$$

Lemma 6.1: *For any undirected, connected graph $G = (V, E)$ and any node $v \in V$ it holds that*

$$\mathsf{cc}(v) \leq \mathsf{trans}(v).$$

Proof: The lemma directly follows from Equation 6.3 and 6.6. □

The density of a graph is another interesting property of a graph. It is defined as the quotient of the existing edges and the possible edges of a graph. Definition 6.8 gives the formal specification of density.

Definition 6.8: *Given an undirected graph $G = (V, E)$. Then, the* density $\mathsf{dens}(G)$ *of the graph G is defined as*

$$\mathsf{dens}(G) := \frac{2\,|E|}{|V| \cdot (|V| - 1)}.$$

Lemma 6.2: *For any connected graph $G = (V, E)$ it holds for the density* $\mathsf{dens}(G)$ *that*

$$\frac{2}{|V|} \leq \mathsf{dens}(G) \leq 1.$$

Proof: Any connected graph consists of at least $|V| - 1$ edges and at most $\frac{|V| \cdot (|V|-1)}{2}$ edges. Thus, the lemma follows directly from Definition 6.8. □

Another important property is, that it holds for any complete graph that the cluster coefficients and the transitivities of all nodes are identical, which is stated in the next lemma.

Lemma 6.3: *For any complete graph $G = (V, E)$ it holds that*

$$\forall v \in V : \mathsf{cc}(v) = \mathsf{trans}(v)$$

and thus

$$\mathsf{CC}(G) = \mathsf{trans}(G).$$

Proof: Consider any complete graph $G = (V, E)$ with $|V| = n$. The open neighborhood of node v is again a complete graph consisting of $n - 1$ nodes. Therefore, we have

$$\mathsf{cc}(v) = \frac{2 \cdot \left(\frac{\mathsf{deg}(v) \cdot (\mathsf{deg}(v)-1)}{2}\right)}{\mathsf{deg}(v) \cdot (\mathsf{deg}(v) - 1)} = 1$$

for any node $v \in V$.

For any node $v \in V$ the closed neighborhood is the whole graph G. Thus, we know

$$\mathsf{trans}(v) = \frac{2 \cdot \left(\frac{\mathsf{deg}(v) \cdot (\mathsf{deg}(v)+1)}{2}\right)}{\mathsf{deg}(v) \cdot (\mathsf{deg}(v) + 1)} = 1$$

for any node $v \in V$.

Finally, it follows from Definitions 6.6 and 6.7:

$$\mathsf{CC}(G) = \frac{1}{|V|} \sum_{v \in V} \mathsf{cc}(v) = \frac{|V|}{|V|} = 1 = \frac{|V|}{|V|} = \frac{1}{|V|} \sum_{v \in V} \mathsf{trans}(v) = \mathsf{trans}(G)$$

□

As the cluster coefficients and transitivities are identical in complete graphs, we claim that in dense graphs the distinction between both measurements is not important.

Proposition 6.1: *For dense networks both measurements—the cluster coefficient* $\mathsf{cc}(v)$ *and the transitivity* $\mathsf{trans}(v)$*—have very close values and are interchangeable. If the network is medium dense or sparse both measurements should be examined.*

In the next section, we define special network structures that gained the interest in analyzing networks. We also discuss the measurements we have presented in this section.

6.2 Network Types

In this section, we give definitions of special network structures that gained interest in literature when analyzing firm networks, social networks or artificial societies. As we are interested in modeling different types of interaction networks based on specific network structures we only concentrate on undirected graph structures in this section.

6.2.1 Random Networks

The simplest network which is often used as a benchmark network is a random network. We follow Erdõs and Rényi's definition of a random network (Erdõs and Rényi, 1959):

Definition 6.9 (Erdõs-Rényi Random Network): *In the $G(n, \mathsf{Pr}_{\mathsf{con}})$ model a graph with n nodes is constructed by adding each (possible) edge with probability $\mathsf{Pr}_{\mathsf{con}}$. This model is called* Erdõs-Rényi random network.

Given this definition we want to discuss the properties defined in Section 6.1.

We start with the number of edges. As the probability for the existence of a specific edge is $\mathsf{Pr}_{\mathsf{con}}$ and the total number of possible edges in an undirected graph containing $|V| = n$ nodes is $\frac{n\cdot(n-1)}{2}$ we get the expected number of edges in a random graph as:

$$\mathrm{E}(|E|) = \mathsf{Pr}_{\mathsf{con}} \cdot \frac{n \cdot (n-1)}{2} \tag{6.7}$$

which gives us an average degree of

$$\mathsf{avgdeg}(G(n, \mathsf{Pr}_{\mathsf{con}})) = 2 \cdot \frac{\mathsf{Pr}_{\mathsf{con}} \cdot \frac{n\cdot(n-1)}{2}}{n} = \mathsf{Pr}_{\mathsf{con}} \cdot (n-1) \simeq \mathsf{Pr}_{\mathsf{con}} \cdot n \tag{6.8}$$

Algorithm 6.1 Generating an Erdõs-Rényi random network

Input: number of nodes n, connection probability $\mathsf{Pr}_{\mathsf{con}}$
Output: An Erdõs-Rényi random network

```
1:  procedure GENERATEERDOSRENYINETWORK(n, Pr_con)
2:      inititialize set of nodes V with n different nodes
3:      E ← ∅
4:      for i = 0 to n − 1 do
5:          for j = i + 1 to n − 1 do
6:              with probability Pr_con: E ← E ∪ {i, j}
7:          end for
8:      end for
9:      return graph G(V, E)
10: end procedure
```

The simplification of the average node / graph degree, i.e. $\mathsf{avgdeg}(G) \simeq \mathsf{Pr}_{\mathsf{con}} \cdot n$, holds for $n \to \infty$ which is a suitable assumption for large networks. Albert and Barabási have discussed several properties of random graphs constructed via the method of Erdõs and Rényi (Albert and Barabási, 2002). They found that the diameter of a random graph, which is the greatest distance between any two nodes, is usually concentrated around

$$\mathsf{diam}(G(n, \mathsf{Pr}_{\mathsf{con}})) = \frac{\ln(n)}{\ln(\mathsf{Pr}_{\mathsf{con}} \cdot n)} \tag{6.9}$$

As the cluster coefficient gives the probability that the neighbors of a node are also neighbors, it follows that the cluster coefficient of a node v in an Erdõs-Rényi random networks is

$$\forall v \in V : \mathsf{cc}(v) = \mathsf{Pr}_{\mathsf{con}} \tag{6.10}$$

and, therefore, for the whole graph

$$\mathsf{CC}(G(n, \mathsf{Pr}_{\mathsf{con}})) = \mathsf{Pr}_{\mathsf{con}}. \tag{6.11}$$

Additionally, Erdõs and Rényi have shown in (Erdõs and Rényi, 1960) that there is a critical connection probability $\mathsf{Pr}^{\star}_{\mathsf{con}}$ which determines whether the resulting graph is connected or not. This critical connection probability can be calculated as

$$\mathsf{Pr}^{\star}_{\mathsf{con}} = \frac{\ln(n)}{n} \tag{6.12}$$

They have shown that for $\mathsf{Pr}_{\mathsf{con}} > \mathsf{Pr}^{\star}_{\mathsf{con}}$ the graph is connected with high probability, and for $\mathsf{Pr}_{\mathsf{con}} < \mathsf{Pr}^{\star}_{\mathsf{con}}$ the graph is not connected with high probability (Erdõs and Rényi, 1960).

Algorithm 6.1 gives a simple version of generating an undirected random network based on the definition of Erdõs and Rényi. It is easy to see that the algorithm will return a random network that satisfies the discussed properties. Figure 6.2 visualizes a random network constructed via the method of Erdõs and Rényi with 50 nodes and a connection probability of 0.14.

6.2.2 Regular Networks

We now want to discuss another type of often used networks, i.e. regular networks. Regular networks are mostly characterized to be networks where each node has exactly the same degree and the graph having a homogeneous connection pattern (Gaston, 2005). The class of regular networks includes lattices, hypercubes and complete graphs.The following definition defines a regular graph in a formal way.

Definition 6.10 (Regular Graph): *A regular graph $G = (V, E)$ is a graph where each node has the same degree, i.e. it holds that*

$$\forall u, v \in V : \deg(u) = \deg(v).$$

In the next paragraphs we want to analyze three well-defined regular networks. Namely, these are the K-ring network, the hypercube and the two-dimensional Lattice.

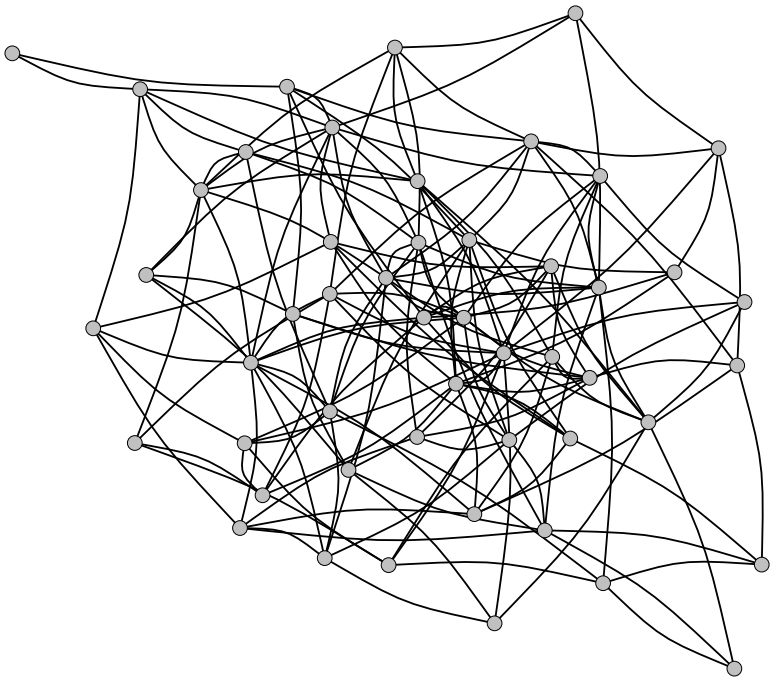


Figure 6.2: Illustration of an Erdõs-Rényi random network generated by Algorithm 6.1 with $n = 50$ nodes and a connection probability of $\mathsf{Pr}_{\mathsf{con}} = 0.14$.

K-ring Graph The K-ring graph $\mathsf{ring}(K, n)$ is a special regular network. The graph consists of n nodes which are arranged in a ring structure and each node is connected to its $2 \cdot K$ nearest nodes (i.e. K to each side). This can be done through labeling the nodes with natural numbers $[0, n-1] \subset \mathbb{N}$. If the difference between two labels is at most K then there is an edge in this graph for these two nodes. The parameter K is often called the *coordination number* of such a graph. Algorithm 6.2 presents a mechism to create such a $\mathsf{ring}(K, n)$ network.

Definition 6.11 (The $\mathsf{ring}(K, n)$ network): *A $\mathsf{ring}(K, n)$ network is a graph consisting of n nodes arranged in a ring structure, where each node is adjacent to the K nearest nodes in both directions of the ring structure.*

It is easy to see that the number of edges in a $\mathsf{ring}(K, n)$ network is $K \cdot n$. For the average node degree we know that it is $2 \cdot K$ as each node has in each of the two directions of the ring an edge to the K nearest neighbors. In such a network the diameter and the characteristic path length are the same due to the regular connection pattern. For both it holds that $\mathsf{diam}(\mathsf{ring}(K, n)) = \mathsf{CPL}(\mathsf{ring}(K, n)) = \frac{n}{2 \cdot K}$. Additionally, it follows that the cluster coefficient and the transitivity are the same for every node due to the construction process and, therefore, it holds that $\mathsf{cc}(v) = \mathsf{CC}(\mathsf{ring}(K, n))$ and $\mathsf{trans}(v) = \mathsf{trans}(\mathsf{ring}(K, n))$. To calculate the cluster coefficient and the transitivity of a specific node v we need to know how many edges are in the open neighborhood of v, i.e. we need $|\mathsf{OPN}(v)|_E$. This is provided in the following lemma:

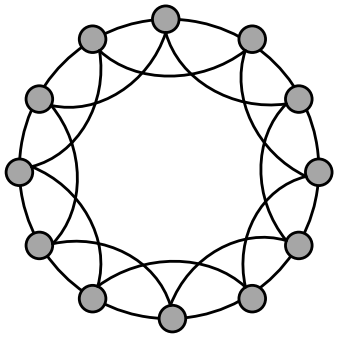


Figure 6.3: Example of a ring regular network with 12 nodes and $K = 2$, i.e. $\mathsf{ring}(2, 12)$.

Lemma 6.4: *For any node $v \in V$ of a ring $\mathsf{ring}(K, n)$ with coordination number K it holds that the number of edges in the open neighborhood of the node v is*

$$|\mathsf{OPN}(v)|_E = \frac{3 \cdot K(K-1)}{2}.$$

Proof: Consider a $\mathsf{ring}(K, n)$ network with coordination number K and n nodes. Due to the regular connection pattern it holds that

$$\forall u, v \in V : |\mathsf{OPN}(u)|_E = |\mathsf{OPN}(v)|_E \,. \tag{6.13}$$

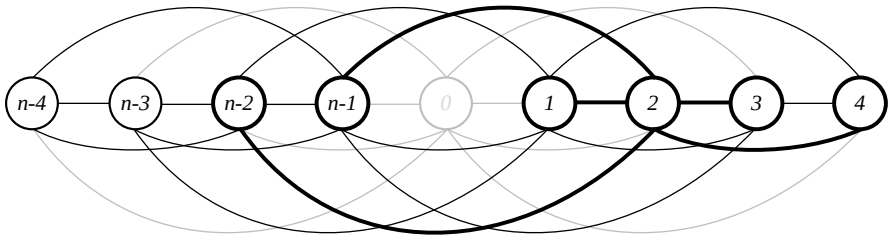


Figure 6.4: Illustration of the subgraph induced by $\mathsf{OPN}(0)$ of a $\mathsf{ring}(4, n)$. The thick edges are adjacent to node $i = 2$ and the thick nodes are the nodes that are connected to node $i = 2$ in the subgraph.

Let us consider the number of edges in the open neighborhood of node 0. The open neighborhood of node 0 is

$$\mathsf{OPN}(0) = \{1, 2, \ldots, K, n-K, \ldots, n-1\}. \tag{6.14}$$

We know that $|\mathsf{OPN}(0)| = 2 \cdot K$. Every node i with $0 < i \leq K$ out of $\mathsf{OPN}(0)$ is connected to the nodes $\{n-K+i, n-K+(i+1), \ldots, n-1, 1, \ldots, i-1, i+1, \ldots, K\} \subset \mathsf{OPN}(0)$ which is a set containing $2 \cdot K - 1 - i$ elements (cf. Figure 6.4 as an example). Therefore, $\mathsf{deg}(i) = 2 \cdot K - 1 - i$. If we sum up the degrees of all nodes i with $0 < i \leq K$ we get

Algorithm 6.2 Generating a K-ring network

Input: coordination number K, number of nodes n
Output: A K-ring network

```
1:  procedure GENERATERING(K, n)
2:      initialize set V with n nodes
3:      label the nodes consequently with natural numbers from [0, n − 1]
4:      for all u ∈ V do
5:          for all v ∈ V do
6:              if difference between (label(u) and label(v)) mod n ≤ K then
7:                  if {u, v} ∉ E then
8:                      E ← E ∪ {u, v}
9:                  end if
10:             end if
11:         end for
12:     end for
13:     return graph G(V, E)
14: end procedure
```

$$\begin{aligned}\sum_{i=1}^{K}(2\cdot K-1-i) &= K(2\cdot K-1)-\sum_{i=1}^{K} i \\ &= K(2\cdot K-1)-\tfrac{K(K+1)}{2} \\ &= \frac{3\cdot K(K-1)}{2}\end{aligned}$$

It follows, that the sum of degrees of all nodes in the induced subgraph based on $\mathsf{OPN}(0)$ is $3\cdot K(K-1)$. As the number of edges is half the sum of degrees we obtain the result [1]. □

As conclusions of Lemma 6.4 we get the formulas for the cluster coefficient

$$\mathsf{CC}(\mathsf{ring}(K,n)) = \frac{3K(K-1)}{2K(2K-1)} \tag{6.15}$$

and for the transitivity

$$\mathsf{trans}(\mathsf{ring}(K,n)) = \frac{4K+3K(K-1)}{2K(2K+1)}. \tag{6.16}$$

Finally, Algorithm 6.2 gives a mechanism how to generate a $\mathsf{ring}(K,n)$ network and Figure 6.3 illustrates the $\mathsf{ring}(2,12)$ network.

[1] This proof was influenced by a similar proof published in (Wallis, 2007, p. 221).

Hypercube In geometry a hypercube is a $\dim$-dimensional analogy of a square ($\dim = 2$) and a cube ($\dim = 3$). For defining a hypercube, we first need to define the *Hamming distance*. The Hamming distance of two bit strings is required, as the construction process of a hypercube can easily be described using this mean.

Definition 6.12 (Hamming Distance): *Given two bit strings s_1 and s_2, the* Hamming distance hd *is the number of different bits for corresponding positions:*

$$\mathsf{hd}(s_1, s_2) = \sum_{i=1}^{|s_1|} \mathsf{comp}(s_1[i], s_2[i])$$

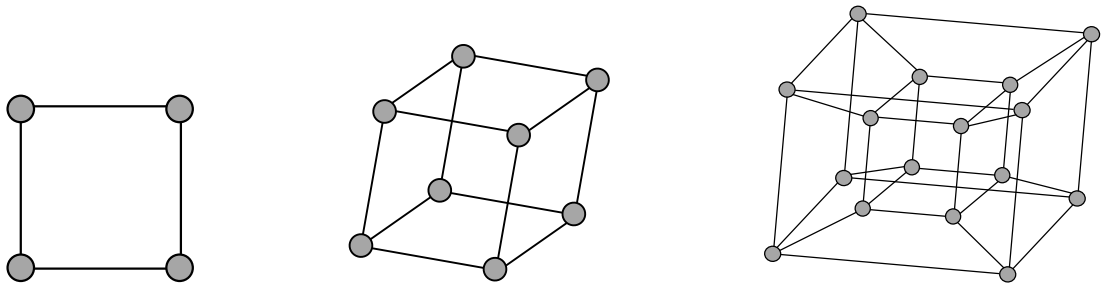
with

$$\mathsf{comp}(x, y) = \begin{cases} 1 & x \neq y \\ 0 & x = y \end{cases}$$

The Hamming distance is only defined for strings of equal length. If the length of the strings differs one can add leading zeros to the shorter string in order to calculate the Hamming distance.

Given the Hamming distance we can now define the $\dim$-dimensional hypercube:

Definition 6.13 (Hypercube): *Given $2^{\dim}$ nodes with bit strings as identifiers. Connect two nodes, if the Hamming distance of their identifiers is equal to 1. The resulting graph is a* hypercube $Q_{\dim}$.



(a) Hypercube Q_2. (b) Hypercube Q_3. (c) Hypercube Q_4.

Figure 6.5: Hypercubes of dimension 2,3 and 4.

The next lemma deals with the diameter of a hypercube.

Lemma 6.5: *For the hypercube $Q_{\dim}$ it holds that the diameter is*

$$\mathsf{diam}(Q_{\dim}) = \dim . \tag{6.17}$$

Proof: Given a hypercube $Q_{\dim}$ of dimension $\dim$. Let v_0 and v_k be two arbitrary nodes of the hypercube. Let the path between two nodes be $v_0, v_1, \ldots, v_k$. For any two nodes v_i and v_{i+1}

Algorithm 6.3 Generating a hypercube of dimension $\dim$.

Input: the dimension $\dim$ of the hypercube
Output: A hypercube of dimension $\dim$

```
1:  procedure GENERATEKHYPERCUBE(dim)
2:      inititialize set V with 2^dim nodes with binary labels
3:      for all u ∈ V do
4:          for all v ∈ V do
5:              if hd(label(u), label(v)) = 1 then
6:                  if {u, v} ∉ E then
7:                      E ← E ∪ {u, v}
8:                  end if
9:              end if
10:         end for
11:     end for
12:     return graph G(V, E)
13: end procedure
```

on the path it holds that the hamming distance is one. Otherwise there would be no such edge $\{v_i, v_{i+1}\}$. Counting the edges on the path gives the desired result.

In other words: As the distance between two nodes is the number of edges of the shortest path and the edges follow the construction of Definition 6.13, the difference between two bit strings of length $\dim$ is at most $\dim$. □

Each node has exactly $\dim$ neighbors as for each bit in the bit string there exists a neighbor where the complement bit string is the only difference in the representation of these two nodes. Therefore, the average degree of the hypercube $Q_{\dim}$ is

$$\mathsf{avgdeg}(Q_{\dim}) = \dim \tag{6.18}$$

From this it follows for the number of edges of the hypercube:

$$|E| = \frac{2^{\dim} \cdot \dim}{2} = \dim \cdot 2^{\dim - 1} \tag{6.19}$$

For the cluster coefficient it holds that it is zero for every node and therefore zero for the hypercube. This follows from the construction of the hypercube. As two neighbors of a specific node each differ in one bit, they differ in two bits, when we compare their bit string representations. Thus, two neighbors never have an edge between them which leads to the conclusion that the cluster coefficients are zero for each node.

Therefore, the formula for the transitivity of a hypercube is:

$$\mathsf{trans}(Q_{\dim}) = \frac{2\dim}{\dim \cdot (\dim + 1)} \tag{6.20}$$

Lattice Graph Lattices in general are a generalization of the K-ring network structure. There, we distinguish the dimension of a lattice. The K-ring network structure is a one-dimensional lattice and a grid network is a two-dimensional lattice. In general a lattice of dimension n is a n-dimensional grid where each two points are connected if and only if their Manhattan distance is less or equal to the coordination number K.

Definition 6.14 (Manhattan Distance): *The Manhattan distance between two points is the sum of the absolute differences of their coordinates. Given two n-dimensional points $p_1 = (p_1^1, p_1^2, \ldots, p_1^n)$ and $p_2 = (p_2^1, p_2^2, \ldots, p_2^n)$ the Manhattan distance* $\mathsf{manhattan}(p_1, p_2)$ *is defined as*

$$\mathsf{manhattan}(p_1, p_2) := \sum_{i=1}^{n} \left| p_1^i - p_2^i \right| .$$

Definition 6.15 (Lattice Graph): *A lattice graph $L_{\dim,K} = (V, E)$ is an undirected graph which has* $\dim$ *dimensions and where the nodes are placed on physical positions, given by the* $\dim$ *dimensions. An edge between two nodes i and j exists, when the Manhattan distance between the two nodes is less or equal to the coordination number K. More formally:*

$$e = \{i, j\} \in E \leftrightarrow \mathsf{manhattan}(i, j) \leq K$$

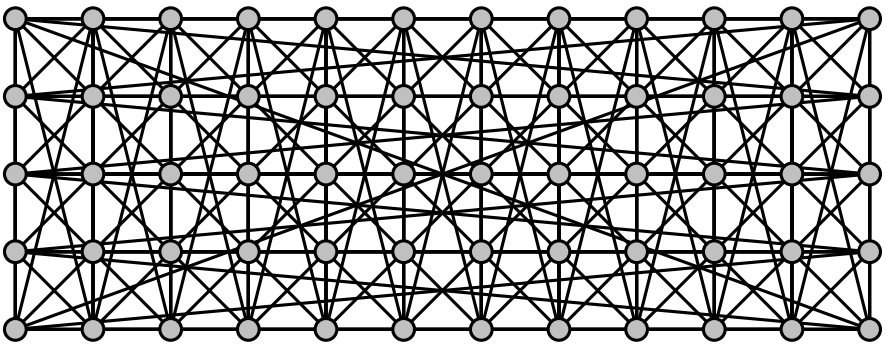


Figure 6.6: Illustration of a two dimensional toroidal lattice with 12 x 5 nodes and $K = 2$.

Note that we will only consider lattice graphs where the borders are connected to each other. When we think about an $L_{1,2}$ lattice graph of dimension 1 with coordination number 2, we obtain a line. When we connect the first and the last node, then we get a cycle which is the $\mathsf{ring}(K, N)$. As we are only interested in lattice graphs that form regular networks we will only consider those $L_{\dim,K}$ graphs where the boundary nodes are connected to each other which is typically called a toroidal lattice. If this is not the case, the resulting network is not regular, as some nodes will have lower degrees than the majority of nodes. Additionally we will only focus on lattice graphs of dimension two as toroidal lattices of dimension $\dim$ have a $\dim+1$ graphical representation.

Typically, one wants to specify a lattice not only in the form presented in Definition 6.15 but

with a particular number of nodes per dimension. Then, it is possible to define the lattice with a vector $\mathbf{dim}$ instead of only the number of dimensions $\dim$. For example the lattice $L_{\mathbf{dim},2}$ with $\mathbf{dim} = (100, 20)$ is a two dimensional lattice with coordination number 2 and 100 nodes in the first and 20 nodes in the second dimension.

The degree of a node in a lattice $L_{2,K}$ is given by the recursive definition

$$\deg(v)_K = \deg(v)_{K-1} + 4 \cdot K \tag{6.21}$$

with $\deg(v)_0 = 0$, as it is the empty graph. For the number of edges it follows that

$$|E_K| = \frac{n \cdot (\deg(v)_{K-1} + 4 \cdot K)}{2}. \tag{6.22}$$

The diameter of a two dimensional lattice is given by the largest Manhattan distance between two nodes and, thus, we get

$$\mathsf{diam}(L_{2,K}) = \left\lceil \frac{n_1 - 1}{2 \cdot K} \right\rceil + \left\lceil \frac{n_2 - 1}{2 \cdot K} \right\rceil \tag{6.23}$$

where n_1 denotes the number of nodes in the first and n_2 the number of nodes in the second dimension.

Algorithm 6.4 specifies a mechanism that generates a K-lattice based on the vector $\mathbf{dim}$.

Algorithm 6.4 Generating a K-lattice of dimension $|\mathbf{dim}|$.

Input: coordination number K, array of number of nodes per dimension $\mathbf{dim}[]$
Output: A K-lattice of dimension $|\mathbf{dim}|$

```
1:  procedure GENERATEKLATTICE(K, dim[])
2:      n ← ∏_{i=1}^{|dim|} dim[i]
3:      initialize set of n nodes V with positions in the |dim|-sphere
4:      for all u ∈ V do
5:          for all v ∈ V do
6:              if manhattan(pos(u), pos(v)) ≤ K and u ≠ v then
7:                  if {u, v} ∉ E then
8:                      E ← E ∪ {u, v}
9:                  end if
10:             end if
11:         end for
12:     end for
13:     return graph G(V, E)
14: end procedure
```

6.2.3 Scale-free Networks

Different to the networks considered before, so called scale-free networks have no formal definition. A network is only defined to be scale-free through its properties. The previously discussed models assume, that we start with a fixed number of nodes n and then connect them through specific mechanisms without changing the number of nodes. In contrast, most real world networks describe open systems that grow like the WWW (Albert and Barabási, 2002). Two mechanism seem to be responsible for the emergence of scale-free networks (Barabási and Albert, 1999). The first mechanism is growth of a network and the second is preferential attachment. A new website will link to existing ones and the probability of linking to a well-known website is greater that the probability of linking to another very new website. The same holds for the graph representing research literature as new scientific article cite other established articles and well-known technical literature (Albert and Barabási, 2002). Definition 6.16 gives an informal definition for scale-free networks.

Definition 6.16 (Scale-free Network): *A* scale-free network SF *is a network whose degree distribution follows a power law, at least asymptotically. Few nodes have high degree, many nodes have a low degree.*

Scale-free networks can be generated with the help of the mentioned mechanism growth and preferential attachment (Albert and Barabási, 2002):

1. *Growth*: Starting with a small number m_0 of nodes, at every timestep we add a new node with $m \leq m_0$ edges that link the new node to m different nodes already present in the system.

2. *Preferential attachment*: When choosing the nodes to which the new node connects, we assume that the probability Pr that a new node u will be connected to node v depends on the degree $\mathsf{deg}(v)$, such that

$$\mathsf{Pr}(\{u, v\} \in E) = \frac{\mathsf{deg}(v)}{\sum_{w \in V} \mathsf{deg}(w)}.$$

Algorithm 6.5 shown an approach for creating a scale-free network. The first step is to select the m_0 so called seed nodes to which one random node v_{rand} is connected to. After the seed nodes are connected to the newly introduced node each non-connected node will connect to m nodes with preferential attachment according to the node degrees. Figure 6.7 illustrates a scale-free network constructed with Algorithm 6.5 containing 100 nodes, 15 seed nodes and an establishment of a single link for all unconnected nodes in the preferential attachment phase.

Let us now analyze how many edges are created in a scale-free network constructed via Algorithm 6.5. After the seed generation (lines 3–7) m_0 edges have been generated. At that point $n - (m_0 + 1)$ many nodes are still unconnected and for each we create m new edges. Therefore,

Algorithm 6.5 Generating a scale-free network.

Input: number of nodes n, number of seed nodes m_0, number of connections per node m during growing phase
Output: A scale-free network

procedure GENERATESCALEFREE(n, m_0, m)
  inititialize set of n nodes V
  $S \leftarrow$ set of m_0 randomly selected nodes from V
  $v_{\text{rand}} \leftarrow$ randomly selected node from $V \setminus S$
  for all $u \in S$ **do**
    $E \leftarrow E \cup \{u, v_{\text{rand}}\}$
  end for
  for all $u \in V$ with $\mathsf{deg}(u) = 0$ **do**
    $C \leftarrow$ set of m nodes randomly selected with probabilities according to $\frac{\mathsf{deg}(c)}{\sum_{w \in V} \mathsf{deg}(w)}$
    for all $c \in C$ **do**
      $E \leftarrow E \cup \{u, c\}$
    end for
  end for
  return graph $G(V, E)$
end procedure

we have for the number of edges in a scale-free network

$$|E| = m_0 + m \cdot \big(n - (m_0 + 1)\big). \tag{6.24}$$

For the average node degree we have

$$\mathsf{avgdeg}(\mathsf{SF}) = \frac{2 \cdot \Big(m_0 + m \cdot \big(n - (m_0 + 1)\big)\Big)}{n} \tag{6.25}$$

and it was shown in (Albert and Barabási, 2002) that the degree distribution follows the power law distribution

$$\mathsf{Pr}(k) \sim c \cdot k^{-3} \tag{6.26}$$

where $\mathsf{Pr}(k)$ gives the fraction of nodes in the networks which have degree k and c is some constant.

6.2.4 Small-World Network

Small-World networks were firstly introduced by Watts and Strogatz in (Watts and Strogatz, 1998) and are also considered in (Watts, 1999, 2003). Many real-world networks have a small-world character like random graphs. However, they have usually large cluster coefficients (Al-

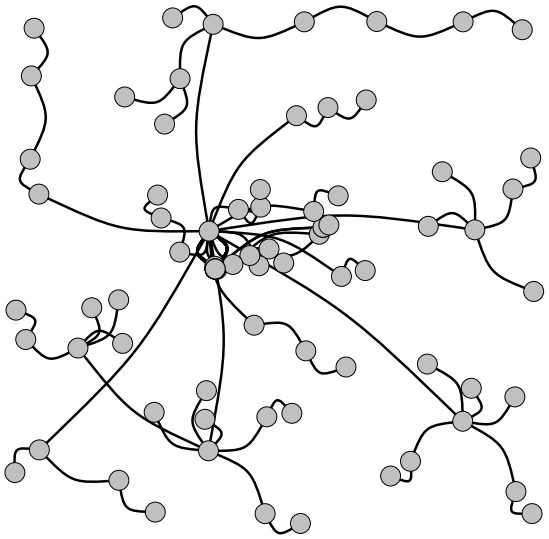


Figure 6.7: Illustration of a scale-free network following Algorithm 6.5 with $n = 100$, $m_0 = 15$ and $m = 1$.

bert and Barabási, 2002). The idea behind those networks is the following. Consider two neighbors in a street. They have many neighbors in common, which cannot be reproduced with random networks. Such phenomena cannot only be found in social networks but also for example in the connections of neural networks (Watts and Strogatz, 1998). Definition 6.17 provides a definition of a small-world network following (Watts and Strogatz, 1998)

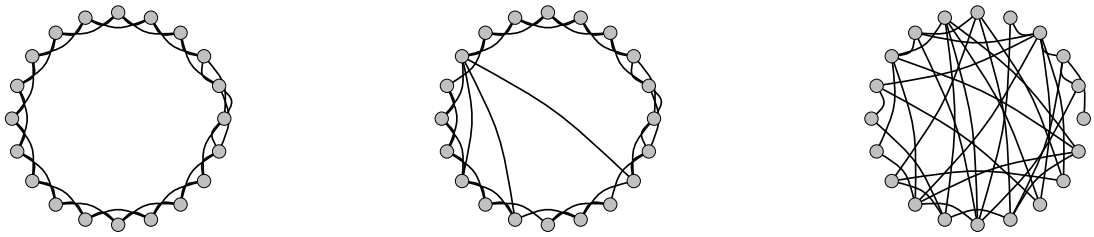
Definition 6.17 (Small-World Network): *A graph with a small characteristic path length and a large cluster coefficient is called a* small-world network.

Watts and Strogatz have presented a mechanism to create such small-world networks (Watts and Strogatz, 1998):

1. *Start with order*: create a $\mathsf{ring}(K, n)$ with n nodes where each node is connected to the K nearest neighbors in both directions of the ring. They suggested to choose $n \gg \frac{K}{2} \gg \ln(n) \gg 1$ in order to have a sparse but connected network at all times (Watts and Strogatz, 1998).

2. *Randomize*: Randomly rewire each edge with probability $\mathsf{Pr}_{\mathsf{rew}}$ such that self-connections and duplicate edges are excluded.

This process is called the WS-model for Small-Worlds due to the authors' names. Another variant of that model was introduced by Newman and Watts (Newman and Watts, 1999a,b). There, the randomly introduced shortcuts do not replace existing edges. The WS-model will be used in this thesis and is summarized in Algorithm 6.6. Figure 6.8 illustrates the influence of the parameter $\mathsf{Pr}_{\mathsf{rew}}$ on an exemplary network constructed via the method of Watts and Strogatz.



(a) $\mathsf{Pr}_{\mathsf{rew}} = 0.0$. (b) $\mathsf{Pr}_{\mathsf{rew}} = 0.05$. (c) $\mathsf{Pr}_{\mathsf{rew}} = 1.0$.

Figure 6.8: Illustrations of small-world networks following the WS-Model for growing rewiring probability $\mathsf{Pr}_{\mathsf{rew}}$.

Algorithm 6.6 Generating a small-world network.

Input: coordination number K, number of nodes n, rewiring probability $\mathsf{Pr}_{\mathsf{rew}}$
Output: A random network having small-world properties

1: **procedure** GENERATESMALLWORD(K, n, $\mathsf{Pr}_{\mathsf{rew}}$)
2: $G(V, E) \leftarrow$ GENERATERING(K,n)
3: $E' \leftarrow E$
4: **for all** $\{u, v\} \in E$ with probability $\mathsf{Pr}_{\mathsf{rew}}$ **do**
5: $v_{\mathsf{rand}} \leftarrow$ randomly chosen node from $V \setminus (\mathsf{neighbors}(v) \cup \{u, v\})$
6: $E' \leftarrow (E' \setminus \{\{u, v\}\}) \cup \{\{u, v_{\mathsf{rand}}\}\}$
7: **end for**
8: **return** graph $G(V, E')$
9: **end procedure**

Obviously, the number of edges in the WS-model $\mathsf{SW}(K, n, \mathsf{Pr}_{\mathsf{rew}})$ constructed by Algorithm 6.6 is the same as for the $\mathsf{ring}(K, n)$, i.e. $|E| = K \cdot n$. Note that the second step removes $\mathsf{Pr}_{\mathsf{rew}} \cdot K \cdot n$ edges and introduces the same number of shortcuts to the network. The same holds for the average node degree due to the rewiring mechanism, i.e. $\mathsf{avgdeg}(\mathsf{SW}(K, n, \mathsf{Pr}_{\mathsf{rew}})) = 2 \cdot K$.

Barrat and Weigt (2000) have shown that the cluster coefficient of a Small-World network generated with the WS-model is given by

$$\mathsf{CC}(\mathsf{SW}(K, n, \mathsf{Pr}_{\mathsf{rew}})) = \mathsf{CC}(\mathsf{ring}(K, n)) \cdot (1 - \mathsf{Pr}_{\mathsf{rew}})^3. \tag{6.27}$$

Due to the relatedness of cluster coefficients and transitivity, a similar conclusion holds for the latter property:

$$\mathsf{trans}(\mathsf{SW}(K, n, \mathsf{Pr}_{\mathsf{rew}})) = \mathsf{trans}(\mathsf{ring}(K, n)) \cdot (1 - \mathsf{Pr}_{\mathsf{rew}})^3. \tag{6.28}$$

6.3 Experimental Setup

In this section, we describe the setup for the experimental evaluation of the influence of different network structures. The previous section dealt with the different types of networks that we want to analyze. Now, we analyze the influence of different network structures on the emerging cooperation in the remainder of this chapter. As the agents are only allowed to interact with their direct neighbors specified by the networks, these networks have great influence on the interaction possibilities and thus have great influence on the emergence of cooperation. Table 6.3 gives a summary of the considered networks and their experimental setups.

Table 6.3: Considered networks and their parameter settings.

Network type	Settings
Erdõs-Rényi random network	$n = 1000$, $\mathsf{Pr}_{\mathsf{con}} = 0.014$
Regular ring network	$n = 1000$, $K = 7$
Two-dimensional toroidal lattice	100×10 nodes, with $K = 2$
Scale-free network	$n = 1000$, $m_0 = 40$ and $m = 7$
Small-world network	$n = 1000$, $K = 7$ and $\mathsf{Pr}_{\mathsf{rew}} = 0.05$

Table 6.4 gives the configuration for the experiments in this section. The parameter-values are gained from the base configuration on page 55 or are adapted because of better results from the experimental analysis of Chapter 5. We will consider an agent system of 1000 agents with five skills in the system and one skill per agent, i.e. $|\mathcal{S}| = 5$ and $|\mathcal{S}_a| = 1$. The maximum number of allowed neighbors is not equal for every agent but is fixed from the initialization. This is due to the fact that especially scale-free networks have nodes with very high degrees and those nodes should be allowed to have this high number throughout the simulations. The number of propositions is set to $m = 10$ which can be rated in the interval $[0, 100]$ and the tolerance values for the agents are drawn from $(0, 50]$, which leads to medium-tolerant agents in the system. The jobs consist of exactly three tasks which lead to a payoff of one utility unit and having a cost factor of -0.25 and in each simulation step $10{,}000$ jobs should be handled by the agents. We set the adaptation strength to $\eta = 0.5$, the number of ideals to one and the elite set size to four. Finally, we will also take a look at the network dynamics and, therefore, we will vary the probability of executing the social networking step. This will be one part in the analysis. Nevertheless, if the social networking is performed only one neighbor may be changed by the agents. Table 6.5 gives the theoretical main characteristics for the network types under consideration.

Table 6.4: Configuration setup for network analysis.

Parameter	Meaning	Value
$\lvert\mathcal{A}\rvert$	population size	1000
$\lvert\mathcal{S}\rvert$	number of skills in the system	5
$\lvert\mathcal{S}_a\rvert$	number of skills per agent	1
$\mathcal{N}_{\max}$	maximum number of neighbors allowed per agent	as initial value
m	number of propositions	10
$v_{\max}$	maximum rating for a proposition	100
$\Theta_{\max}$	maximum tolerance of an agent	50
$q_{\min}$	minimum payoff for task fulfillment	1
$q_{\max}$	maximum payoff for task fulfillment	1
$t_{\min}$	minimum number of tasks a job consists of	3
$t_{\max}$	maximum number of tasks a job consists of	3
k	job factor	10
c	cost-factor for charging helpers	-0.25
η	adaptation strength	0.5
ε	number of elite agents	4
$\lvert\mathcal{I}\rvert$	number of ideal agents	1
$\mathsf{Pr}_{\mathcal{N}}$	probability of executing social networking	not fixed
r	number of replaced agents in social networking	1

6.4 Static Networks

We first examine the influence of the different network structures if they are static. Therefore, we set $\mathsf{Pr}_{\mathcal{N}} = 0$. The results are presented in Figures 6.9–6.11.

As can be seen, the networks can be divided into three different groups with same results. There is the scale-free network $\mathsf{SF}(1000, 40, 7)$ which quickly leads to a high rate of completed jobs. In none of the other networks, the agents are able to produce such high job completion rates within early simulation steps. Then, the next group only contains the lattice structure $L_{[100,10],2}$ which does not converge to high rates of completed jobs within the 200 simulation steps. Only an average completion rate of approximately 25% is reached at the end. The third group of

Table 6.5: Overview on networks' properties given by the setting.

Network	$\|E\|$	$\mathsf{avgdeg}(v)$	$\mathsf{CC}(G)$
$\mathrm{ER}(1000, 0.014)$	6993	14	0.014
$\mathsf{ring}(1000, 7)$	7000	14	≈ 0.6923
$L_{[100,10],2}$	6000	12	—
$\mathsf{SF}(1000, 40, 7)$	6753	13.5	—
$\mathsf{SW}(1000, 7, 0.05)$	7000	14	≈ 0.5936

network structures contains the Erdõs-Rényi random network $\mathrm{ER}(1000, 0.014)$, the regular ring network $\mathsf{ring}(1000, 7)$ and the small-world network $\mathsf{SW}(1000, 7, 0.05)$ which all lead to the same high level of completed jobs of about 90% and have roughly similar developments over the simulation steps. Indeed, in the end all three perform better compared to the scale-free network (cf. Figure 6.9).

When considering the local cooperation willingness we observe that in all networks, except for the lattice, we reach high levels of mutual cooperation. Compared to the global cooperation willingness it is interesting to see that the ring produces high levels of local but low levels of global cooperation willingness. Therefore, we assume that in the ring structure we do not get areas in the ratings-sphere, where nearly every agent is within the tolerance area of all other agents but many small areas where mutual cooperation exists to directly connected agents but nearly no cooperation is possible between unlinked agents. In all other kinds of networks, it converges to more or less mutual cooperation on the global level (cf. Figure 6.10 and Figure 6.11).

To conclude the influence of the network structures in the static case, we state that all networks but the lattice structure are able to achieve high levels of job completion rates. This is due to mutual cooperation on the local level but may also be very different on the global level as it is the case for the regular ring structure. In the next section, we consider the same network structures under the influence of the network dynamics.

6.5 Dynamic Networks

In this section, we examine the influence of different dynamic network structures. To introduce dynamics, we use the social networking step. Therefore, we use $\mathsf{Pr}_{\mathcal{N}} = 0.04$ for low dynamics and $\mathsf{Pr}_{\mathcal{N}} = 0.16$ for high dynamics. We are also interested in the question whether the network properties evolve to those obtained in random networks. To answer this question we will not only take a look on the job completion rate but we will also consider the average degree of the nodes, the cluster coefficient and the characteristic path length of the networks.

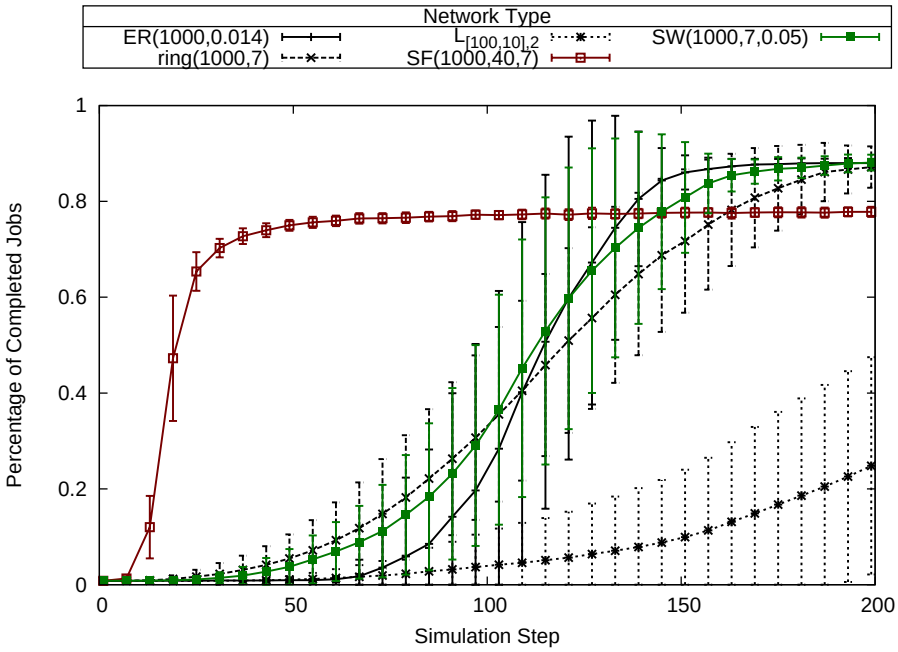


Figure 6.9: Percentage of completed jobs for the considered network structures.

6.5.1 Networks with Low Dynamics

In this experiment, we consider networks with little dynamics by setting $\mathsf{Pr}_{\mathcal{N}} = 0.04$. The results are given in Figures 6.12–6.15. Compared to networks without dynamics from the previous section, it is easy to see that the job completion rate is only slightly influenced by the dynamics (cf. Figure 6.9 and Figure 6.12). The reached level of completed jobs is slightly lower than in the static case.

If we consider the evolution of the networks due to the dynamics we observe that the scale-free network falls apart into different connected components (cf. Figure 6.13). The same holds for the Erdõs-Rényi random network but there the mean number of connected components does not exceed 1.5. In all other networks, we hardly can observe this development.

Figure 6.14 shows the development of the cluster coefficients of the networks. As can be seen for the Erdõs-Rényi network and the scale-free network there is no change in the cluster coefficient throughout the simulation. But we can observe a decrease for all other types of networks. Here, we see that the greater the cluster coefficient in the beginning is the greater is the decrease of the cluster coefficient within the considered 200 simulation steps.

The development of the characteristic path length shows similar results (cf. Figure 6.15). For the Erdõs-Rényi network, the small-world and the scale-free network we can observe that the characteristic path length remains roughly constant throughout the simulation. The greatest characteristic path length is given in the regular ring network in the very beginning. But the

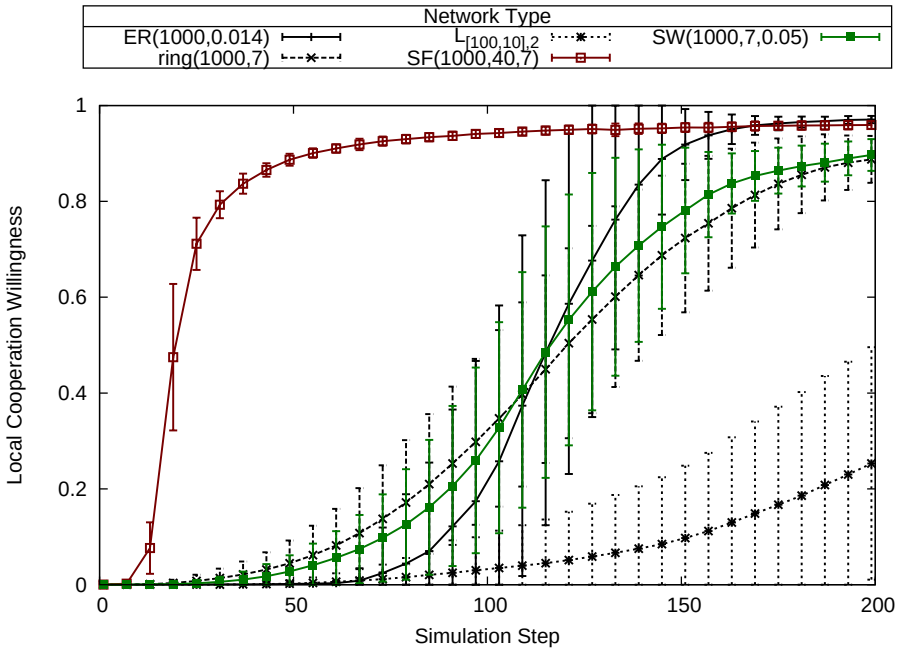


Figure 6.10: Local cooperation willingness for the considered network structures.

value of approximately 70 is decreased to less than 10 in simulation step 200. The second highest characteristic path length can be observed for the lattice network. There, we have a value of 29 in the beginning and 11 at the end of the simulation.

From these results we can conclude that the networks tend to lose their specific properties and gain the properties of the Erdõs-Rényi random networks. This will also be highlighted in the next section where the dynamics of the networks are very high.

6.5.2 Networks with High Dynamics

In highly dynamic networks with $\mathsf{Pr}_{\mathcal{N}} = 0.16$, we observe roughly the same developments as for less dynamic networks with $\mathsf{Pr}_{\mathcal{N}} = 0.04$. Again, the job completion rate is only slightly effected as can be seen in Figure 6.16. Like for the networks with low dynamics, the reachable level of completed jobs is slightly lower than in the case without dynamics.

What we can observe is that the networks faster fall apart into different connected components compared to networks with $\mathsf{Pr}_{\mathcal{N}} = 0.04$. In particular, this can be seen for the scale-free network which falls apart in the approximately 25^{th} simulation step in Figure 6.17. Also the cluster coefficient and the characteristic path length decrease faster than in the low dynamical case (cf. Figure 6.18 and Figure 6.19).

To conclude on the dynamics of networks we can observe that the networks tend to degen-

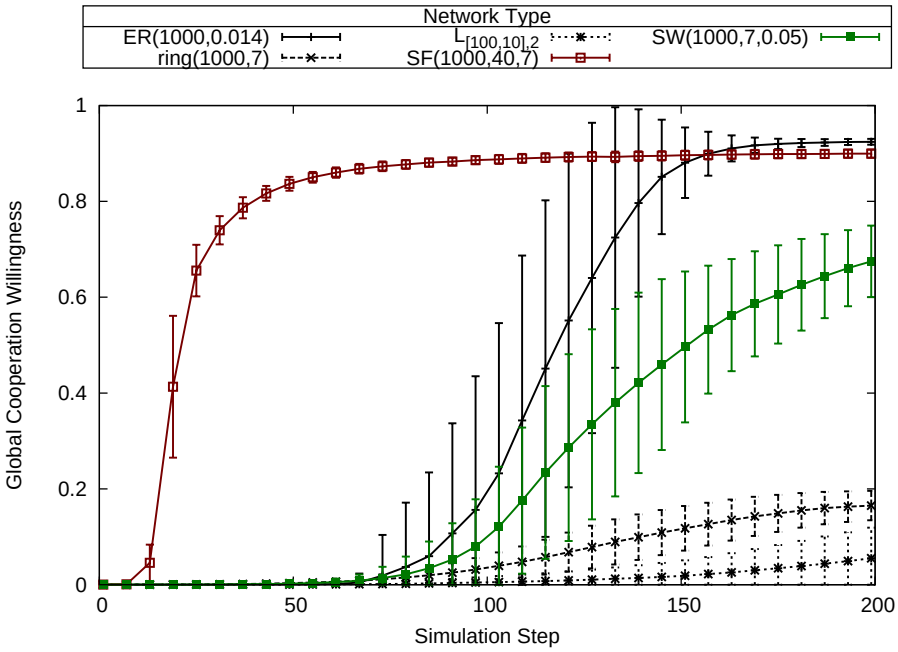


Figure 6.11: Global cooperation willingness for the considered network structures.

erate to random networks. The small-world network loses its property of having a high cluster coefficient. Besides the loss of high cluster coefficients, the regular ring and the lattice structure do also lose their high characteristic path length. Although the fast development to high levels of job completion rates is significant for scale-free networks, the scale-free networks have the problem that they are not able to reach a higher level like the Erdõs-Rényi random networks, the small-world network or the regular ring network. This is due to the fact that a quite high number of agents has a very small number of neighboring agents and, therefore, it is hard to have all skills available in the vicinity of the agents.

In the next sections, we look at cooperation in two specific regular networks. First, we examine the cooperation willingness in a static regular ring compared to an Erdõs-Rényi random network. Second, we consider cooperation in a lattice network.

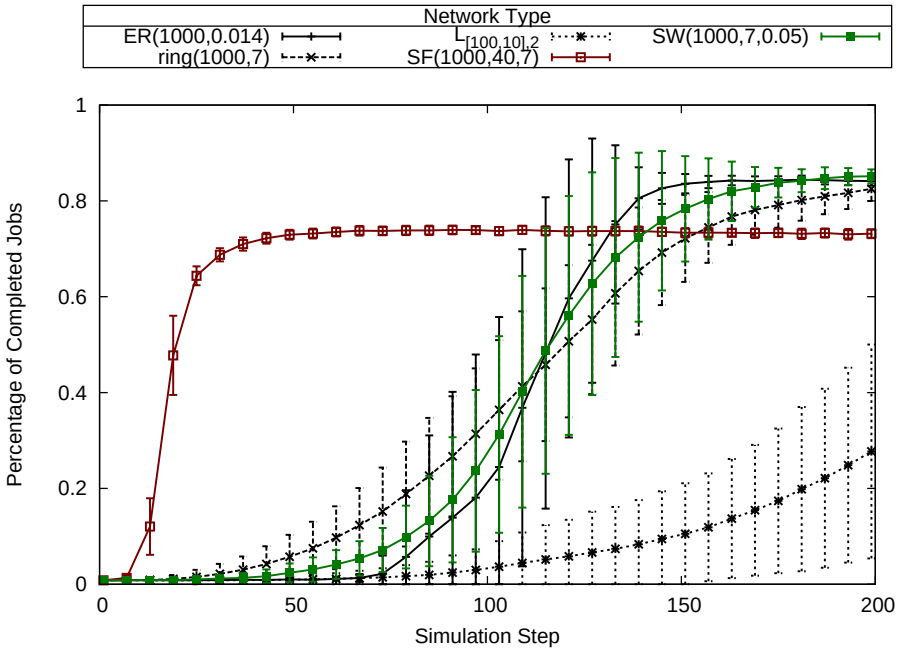


Figure 6.12: Percentage of completed jobs for low dynamic networks.

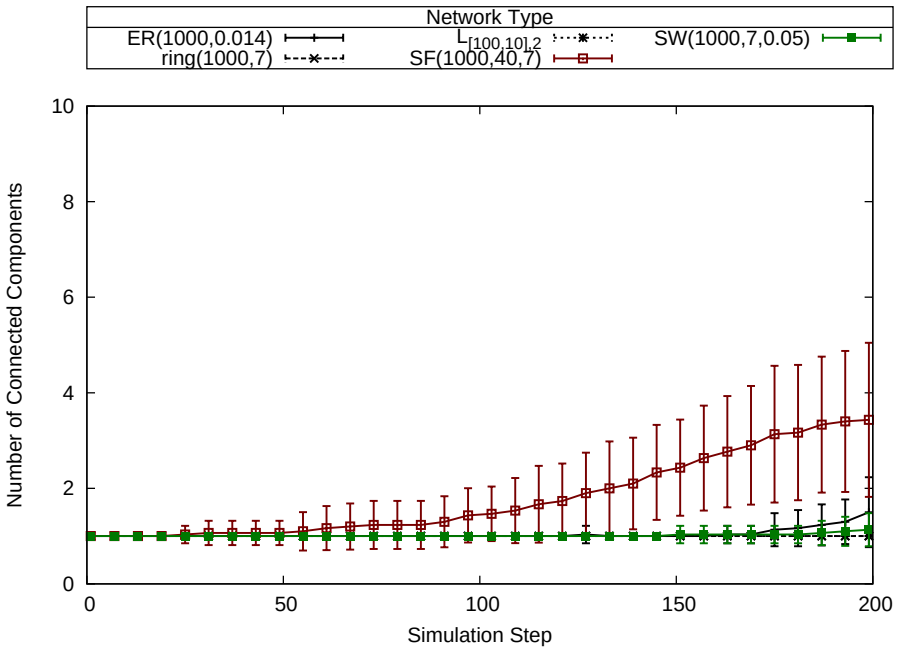


Figure 6.13: Development of the number of connected components for low dynamic networks.

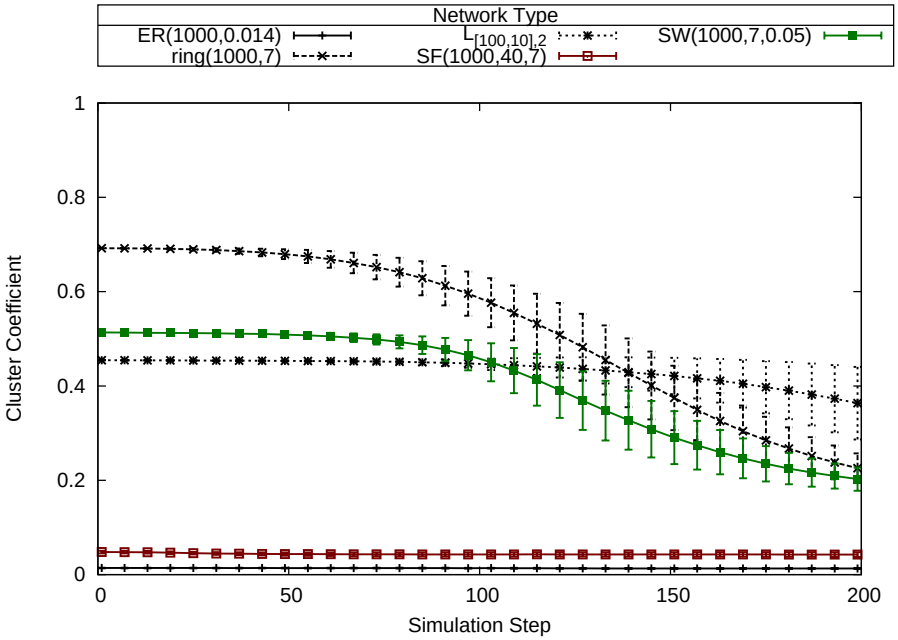


Figure 6.14: Development of the cluster coefficient for low dynamic networks.

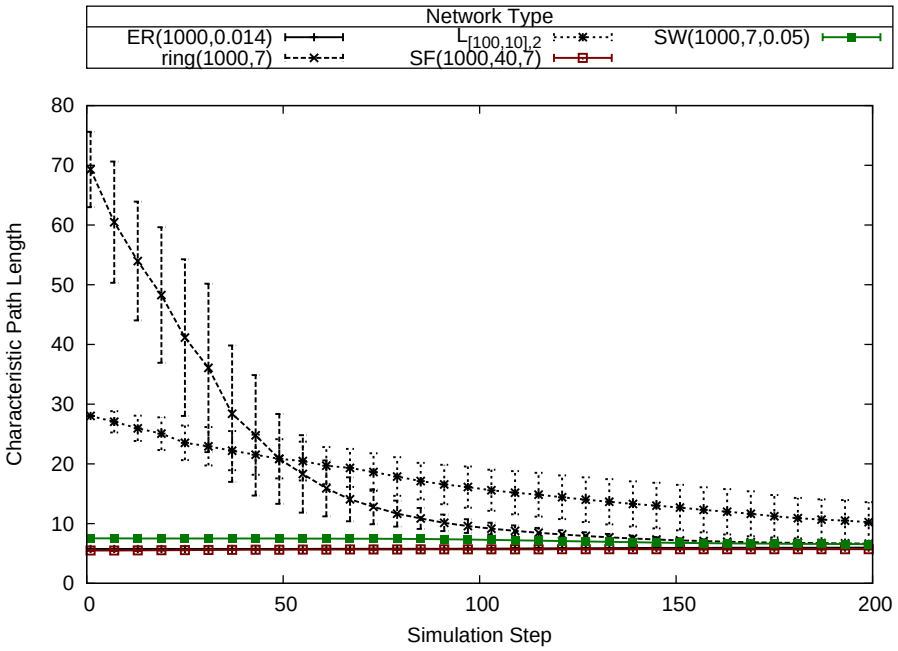


Figure 6.15: Development of the characteristic path length for low dynamic networks.

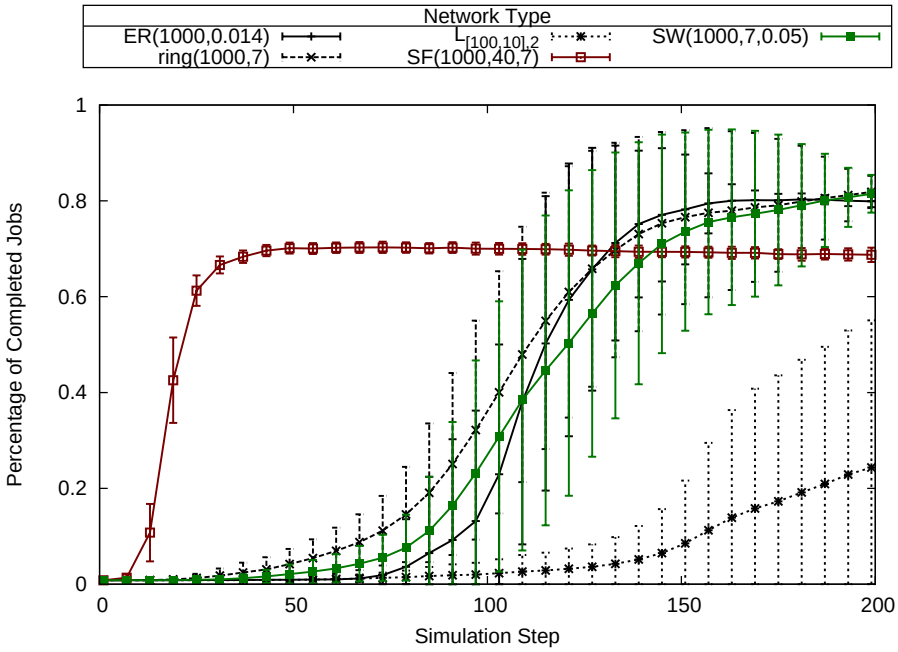


Figure 6.16: Percentage of completed jobs for highly dynamic networks.

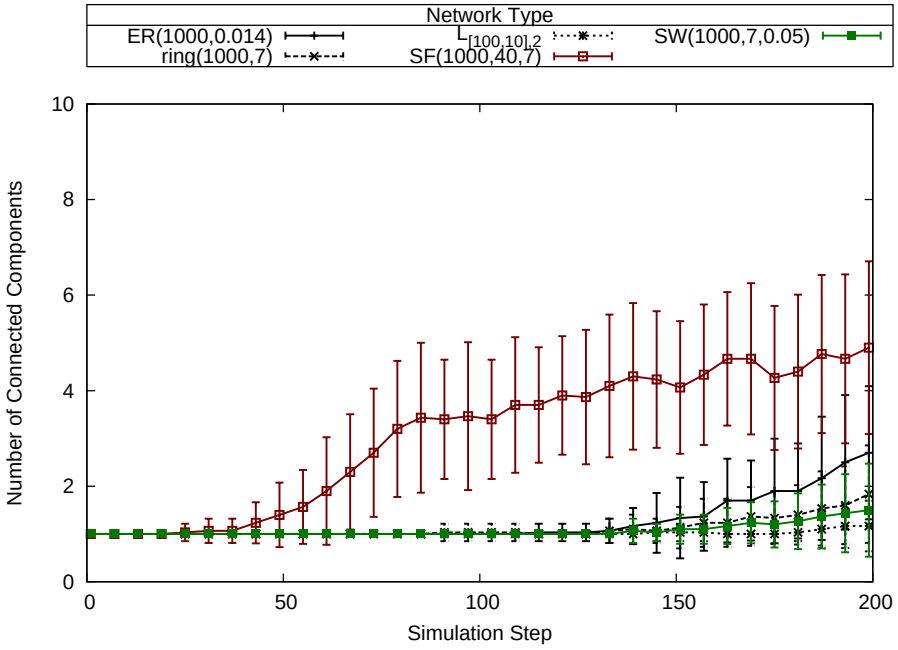


Figure 6.17: Development of the number of connected components for highly dynamic networks.

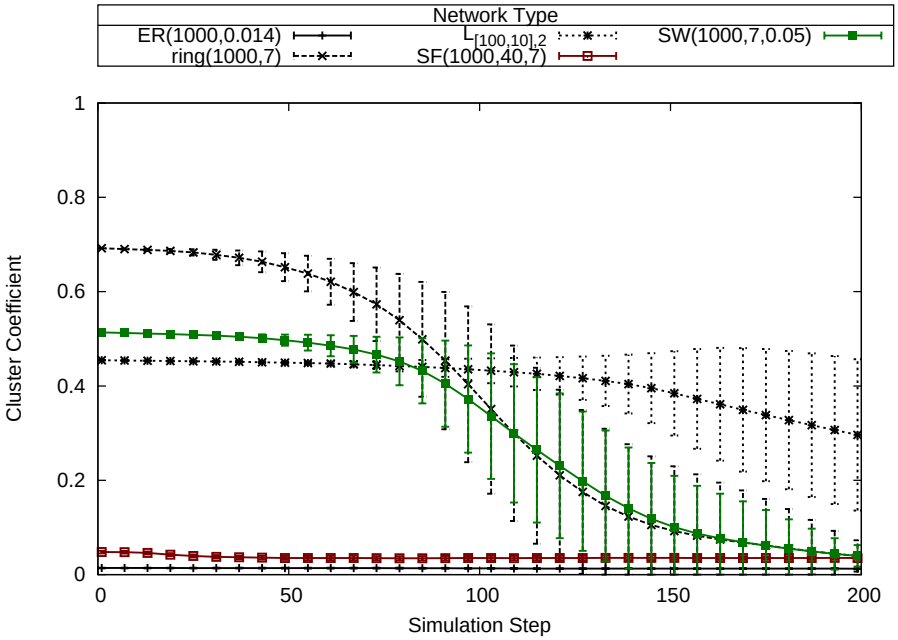


Figure 6.18: Development of the cluster coefficient for highly dynamic networks.

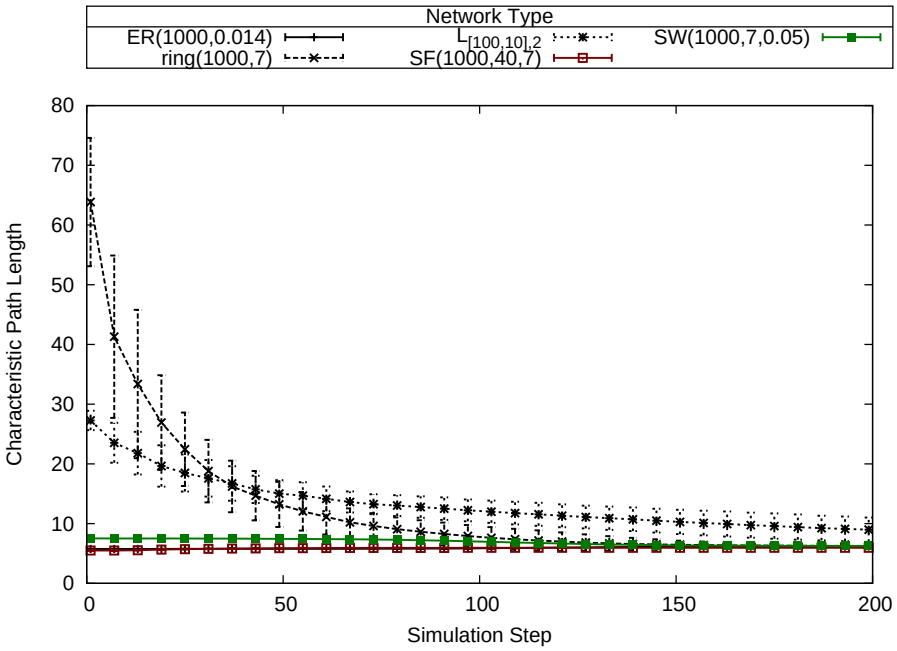


Figure 6.19: Development of the characteristic path length for high dynamical networks.

6.6 Regular Ring and Ratings for Cooperation

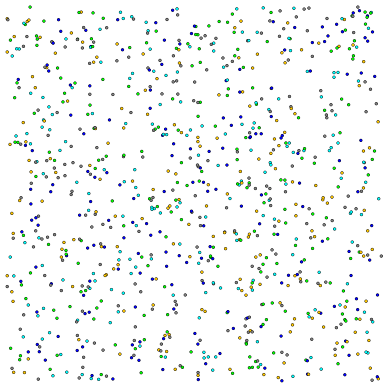
As we have assumed in previous sections, the cooperation willingness in a regular ring structure favors the local relationship between the agents and does produce groups of mutually cooperative agents, which are not cooperative to non-group members. To show that this is true, we compare the vector positions in the rating space in a typical run for a static regular ring and for an Erdõs-Rényi random network.

We will use a regular ring $\mathsf{ring}(1000, 7)$ and an Erdõs-Rényi random network $\mathrm{ER}(1000, 0.014)$ as we did in the experimental analysis of Section 6.4. For better visualization we will only consider a small set of propositions, i.e. $|\mathcal{P}| = 2$. This allows us to visualize the rating vectors as points in a two dimensional Euclidean space. Figure 6.20 shows the visualization of the rating vectors.

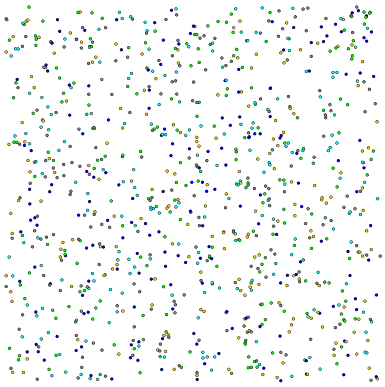
The initialization phase produces the same starting position of the agents' value vectors, which can be seen in Figure 6.20a and Figure 6.20b. This is due to the fact that the ratings are generated in exactly the same way so that we can compare the two network structures properly.

After 50 simulation steps the positions of the rating vectors differ significantly. In the case of the regular ring structure we observe the expected groupings and see that those groups tend to cluster in different areas of the rating space (cf. Figure 6.20c). For the Erdõs-Rényi random network we see that the vectors cluster more or less in the middle of the space (cf. Figure 6.20d).

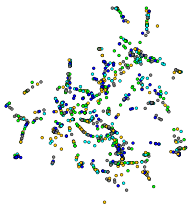
As can be observed after additional 150 simulation steps the position of the ratings stay roughly constant. We assume that this overall behavior is due to the fact that the diameters of the networks differ that much. In the case of this specific Erdõs-Rényi random network we have a diameter of 5 and a diameter of 72 for this specific regular ring structure. Also the cluster coefficients differ a lot as we have $\mathsf{CC} \approx 0.692$ for the ring and $\mathsf{CC} \approx 0.014$ for the random network. As a conclusion, we argue that high diameters and large cluster coefficients together lead to high rates of clustering of the value vectors which we call *polarization* and we get *globalization* if the diameter is quite low paired with low cluster coefficients.



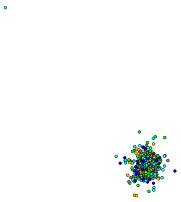
(a) Initialization regular ring.



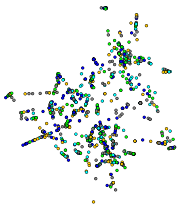
(b) Initialization Erdõs-Rényi network.



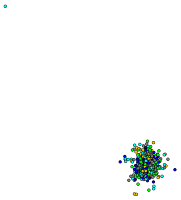
(c) Regular ring after 50 steps.



(d) Erdõs-Rényi network after 50 steps.



(e) Regular ring after 200 steps.



(f) Erdõs-Rényi network after 200 steps.

Figure 6.20: Illustrations of typical runs for a regular ring (left-hand side) and Erdõs-Rényi random network (right-hand side). In all figures the rating vectors are shown in the two dimensional rating-space.

6.7 Cooperation in Lattice Structures

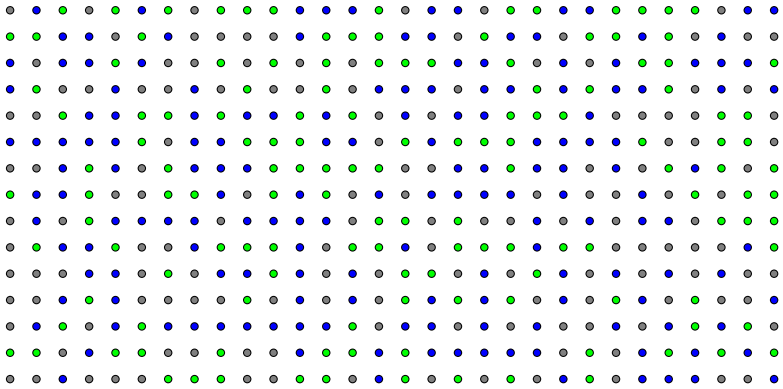
In this section, we investigate groupings in a two-dimensional lattice. We will consider a specific lattice of 30×15 agents with $K = 1$. Different to the lattices considered in previous sections the lattice here is not a torus. We decided to use a non-toroidal lattice for better visualization without enabling the social networking step. Thus, the lattice is static. As each agent has only between 2 and 4 neighbors we will set the number of total skills in the system to three. The number of rated propositions is set to $|\mathcal{P}| = 10$ and the ratings were randomly chosen out of $[0, 100]$ and the threshold values out of $(0, 25]$, both uniformly distributed. Thus, we have relatively intolerant agents. Each agent wants to be at least the second best agent in its vicinity and thus will adapt to the best neighbor if it is not satisfied with its achieved profit. Figure 6.21 shows a typical run for this setting. The colors indicate the skill of the agent and the directed links show the cooperation willingness where the end of the link is the agent that will receive cooperation from the start of the link.

In the beginning after the initialization phase, there is no cooperation willingness in the system. Due to a quite high number of propositions and a quite low tolerance range, none of the agents is willing to cooperate with its neighbors. This is visualized in Figure 6.21a.

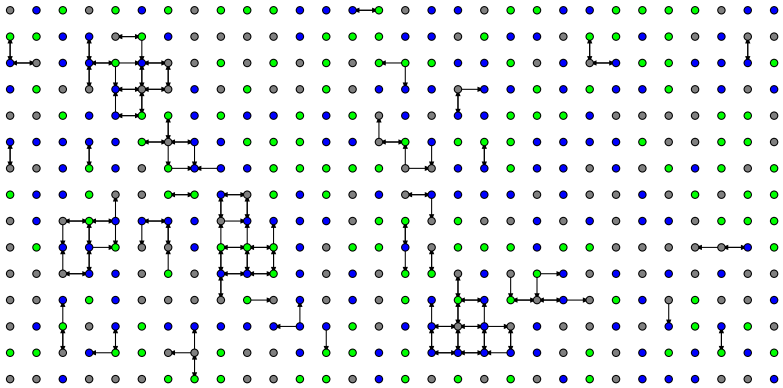
An intermediate state of the system is shown in Figure 6.21b. There, we see the beginning of group formation after the 50^{th} simulation step. Agents start forming cooperative subgroups with mutual cooperation patterns but some nodes are still isolated in the graph formed by the cooperation relation.

Figure 6.21c shows the final cooperation patterns after 200 simulation steps. Most of the links are bidirectional, which means they visualize mutual cooperative agents. There, we can distinguish between the inner-group cooperation which is at a quite high level and the non-cooperative behavior between the groups. As shown, there are quite distinguishable borders between the highly cooperative groups of agents.

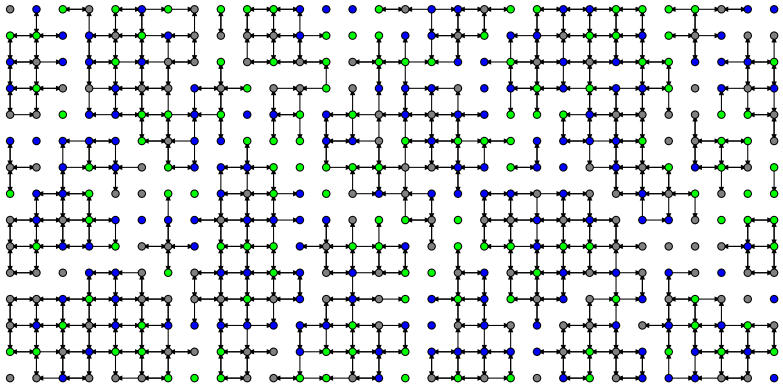
This example with a fixed network structure with low clustering and very intolerant agents results in highly cooperative groups. The agents group together and form mutually cooperative subgraphs. To non-members of a group, the whole group does not have any cooperative links, which shows that these subgroups are very cooperative inside but uncooperative from a global perspective.



(a) Lattice after initialization.



(b) Lattice after 50 simulation steps.



(c) Lattice after 200 simulation steps.

Figure 6.21: Illustrations of a typical run for the considered lattice. One can see the emerging groups with clear boundaries.

6.8 Conclusion

In this chapter, we have investigated the influence of different network structures on the system's performance. First, we introduced the reader to basic definitions and terminology from graph theory. Then, some relations between the properties considered in this thesis have been proven. Secondly, we introduced the considered network structures and provided algorithms to construct these networks.

In the experimental part of this chapter, it was shown how the structures influence the performance of the system in the static and dynamic case. For the dynamic case, we have seen that the network structures degenerate into random networks at least considering the properties cluster coefficient and characteristic path length. We considered the ratings diffusion in regular ring networks and compared it to the diffusion in Erdõs-Rényi random networks. As result, we showed that the Erdõs-Rényi random networks tend to move all rating vectors in a small area of the rating space whereas in regular ring structures smaller groups of rating vectors occur which build clusters with high cooperation in them and no cooperation between the clusters. For cooperation in lattice structures, we showed that we can observe similar behavior. There, groups of agents have formed, which had very clear boundaries to other groups or isolated agents.

7 Introducing Capacity Constraints

In this chapter, we introduce capacity constraints to the agents and to the tasks. Up to now every agent may process as many tasks as it wants to since no task requires a specific amount of skill units as resource requirements. As we have seen in previous chapters, a minimum number of *positive* interactions is needed to reach emergent cooperative behavior. Positive interactions are those, which lead to a job completion as only then payoff is given to the agents. We are interested in the influence of capacity constraints as we assume that tight constraints lead to less positive interactions and thus to lower cooperation rates between the agents. The structure of this chapter is as follows. First, we expand the formal model presented in Chapter 3 to formally define the capacity constraints. Then, we analyze the expanded model and show some properties. Finally, we provide experimental results.

In this chapter we will present

- a formal definition of capacity constraints.
- a proof that the JOBPROCESSING algoritm of Chapter 3 always produces *complete* and *local* task allocations.
- which part of the algorithm has to be changed to add *correctness* to the produced task allocations.
- an experimental analysis that the constructed agent system can deal with capacity constraints.
- an experimental analysis that knowledge about the set of jobs and especially the efficiency of jobs does not strongly increase the social welfare.

7.1 Extension of Formal Model

In many real world problems, the assumption that an agent can process an infinite number of tasks is not suitable. There are usually constraints such as bounded bandwidth, bounded energy support or bounded memory capacities that have to be considered. Therefore, we will now consider agents that are not able to process an arbitrary number of tasks in one simulation step. The formal model used up to now does not support the modeling of agents with capacity constraints. This section provides an extension of the formal model introduced in Chapter 3 so that capacity constraints and requirements can be modeled. First, we introduce a capacity function cap_a for each agent, which is formally given in Definition 7.1.

Definition 7.1 (Capacity Function): *Let $\mathcal{S}$ be the set of skills. The* capacity function cap_a *of agent a with the signature $\mathsf{cap}_a : \mathcal{S} \to \mathbb{N}$ describes how many units of each skill are available. The function can be split into two different functions used_a and unused_a with*

- $\mathsf{used}_a : \mathcal{S} \to \mathbb{N}$ *describes how many units are used*
- $\mathsf{unused}_a : \mathcal{S} \to \mathbb{N}$ *describes how many units are unused*

such that the function cap_a is the sum of this two functions:

$$\forall s \in \mathcal{S} : \mathsf{cap}_a(s) := \mathsf{used}_a(s) + \mathsf{unused}_a(s)$$

The capacity function is fixed for every agent in each simulation step. Again we omit the time index for better readability. During execution of one task, the functions used_a and unused_a change as some amounts of the provided skill are used for the execution. As each task only requires a single skill we will model the requirements for tasks as a function $\mathsf{req}(t)$ mapping tasks to natural numbers. The meaning is that the skill s_t specified in task $t = (s_t, q_t)$ is used $\mathsf{req}(t)$ times. Definition 7.2 gives the formal description of the requirement function.

Definition 7.2 (Requirement Function): *For a task $t = (s_t, q_t)$ the requirement function $\mathsf{req}(t)$ is given as*

$$\mathsf{req}(t) : \mathcal{T} \to \mathbb{N}$$

and specifies the amount of units of skill s_t that are needed for the task fulfillment. An agent a can fulfill the task t if and only if $s_t \in \mathcal{S}_a$ and $\mathsf{unused}_a(s_t) \geq \mathsf{req}(t)$.

In the next section, we give analytical results of the resulting extented model.

7.2 Analytical Results

In the considered scenario, jobs are allocated to randomly chosen agents. These agents are then responsible for the tasks' execution and search for cooperative neighbors that are willing to process the tasks. From that point of view, it is obvious that the agents compute a task allocation starting from a random job allocation. The allocation of tasks to agents and vice versa as well as the job allocation are formally defined in Definition 7.3 and Definition 7.4 respectively.

Definition 7.3 (Task Allocation): *A task allocation ψ is a function $\psi : \mathcal{T} \to \mathcal{A}$ which defines to which agent a task is allocated. $\psi^{-1} : \mathcal{A} \to Pow(\mathcal{T})$ is a function, which gives the set of tasks allocated to an agent (i.e. $t \in \psi^{-1}(a) \Leftrightarrow \psi(t) = a$).*

Definition 7.4 (Job Allocation): *A job allocation ϕ is a function $\phi : \mathcal{J} \to \mathcal{A}$ which defines to which agent a job is allocated. The tasks that have been allocated through a task allocation ψ have been processed. For the set of completed jobs $\mathcal{J}_C \subseteq \mathcal{J}$ the following holds:*

$$\forall j \in \mathcal{J}_C \;\; \forall t \in j \;\; \exists a \in \mathcal{A} : \psi(t) = a$$

Per definition, Job allocations are always valid allocations as they do not consider the jobs' execution. But not every task allocation is a valid task allocation as task allocations always represent the task processing. Therefore, the required skills have to be available at the agents and also the number of available units at the agent has to be at least that great as the allocated tasks require. Like in literature (de Weerdt and Zhang, 2008; de Weerdt et al., 2007), we will define properties of a *valid* task allocation, i.e. correctness, completeness and locality:

Definition 7.5 (Valid Task Allocation): *Given a set of allocation jobs $\mathcal{J}$. Let $\mathcal{J}_C \subseteq \mathcal{J}$ be the subset of jobs that have been completed based on the task allocation ψ. The task allocation ψ is valid, if it satisfies three properties:*

1. correctness*: A task allocation is correct if the sum of all skill requirements of all tasks that are located at an agent a is not greater than a's capacities for the skills:*

$$\forall a \in \mathcal{A}, \forall s \in \mathcal{S}_a : \left(\sum_{\substack{t \in \psi^{-1}(a) \\ s_t = s}} \mathsf{req}(t) \right) \leq \mathsf{cap}_a(s)$$

2. completeness*: A task allocation is complete if for all tasks of all completed jobs there exists an agent to which the task is allocated:*

$$\forall j \in \mathcal{J}_C : \forall t \in j : \exists a \in \mathcal{A} : \psi(t) = a$$

3. locality*: A task allocation is local if the tasks of all completed jobs are allocated to the agent to which the jobs are allocated or to its direct neighbors:*

$$\forall j \in \mathcal{J}_C : \phi(j) = a \Rightarrow \forall t \in j : \psi(t) \in \mathcal{N}_a \cup \{a\}$$

Given Definition 7.5, one could ask if Algorithm 3.2 JOBPROCESSING presented on page 31 produces a complete and local task allocation. This is stated, in the following lemma:

Lemma 7.1: *Algorithm 3.2 (*JOBPROCESSING*) is guaranteed to produces a complete and local task allocation.*

Proof: Assume the resulting task allocation is not complete. Then there has to be a job of which some tasks are not allocated and others are. This is not possible as the real allocation is the last step of the algorithm after for each task of the job a potential helper has been found.

Assume the resulting task allocation would not be local. Then there has to be at least one agent b to which a task t is allocated which belongs to a job j allocated to agent a which is not a direct neighbor of b, i.e. $a \notin \mathcal{N}_b$. However, this is not possible as the algorithm only allocates tasks to other agents which are neighbors of agent a. Since the neighborhood relation is symmetric this would be a contradiction. Thus, the lemma is proven. □

We are able to produce a local and complete task allocation with the proposed algorithm of Chapter 3 but up to now we are not able to produce a *correct* allocation as the capacities of the agents are not taken into account. Thus, we have to adjust the algorithm to produce a correct allocation. This slight change has only to be done in line six of Algorithm 3.2. The line has to be changed to:

$$\mathcal{N}_a(s_t) \leftarrow \{b \in \mathcal{N}_a \mid s_t \in \mathcal{S}_b \wedge \mathsf{cap}_a(s_t) \geq \mathsf{req}(t)\} \tag{7.1}$$

It is easy to see that this will result in a valid task allocation.

There exist many metrics to analyze the performance of the proposed algorithm. In earlier experiments we only concentrated on the percentage of completed jobs to show how good the mechanisms work. de Weerdt et al. (2007) used another metric that is based on the efficiency of the completed tasks. We will introduce this metric here and will, therefore, first define the efficiency of a job in Definition 7.6 and then we will define an efficient job processing in Definition 7.7.

Definition 7.6 (Job's Efficiency): *The* efficiency function $e : \mathcal{J} \rightarrow \mathbb{R}$ *gives the efficiency for a job. The efficiency* $e(j)$ *of a job is defined as the quotient of the profit that can be achieved by job fulfillment and the needed skill units to fulfill the job. Formally:*

$$e(j) := \frac{\sum_{t \in j} q_t}{\sum_{t \in j} \mathsf{req}(t)}$$

The efficiency of the set of completed jobs $e(\mathcal{J}_C)$ *is defined as the sum of the jobs' efficiencies:*

$$e(\mathcal{J}_C) := \sum_{j \in \mathcal{J}_C} e(j)$$

Definition 7.7 (Efficient Job Processing): *The utility gain of a set of completed jobs* $\mathcal{J}_C$ *is given by*

$$U(\mathcal{J}_C) := \sum_{a \in \mathcal{A}} \mathsf{profit}(a).$$

A job processing is efficient *if it is the result of the completed jobs* $\mathcal{J}_C \subseteq \mathcal{J}$*—which are given by a valid task allocation—and for every set of completed jobs* $\mathcal{J}'_C \subseteq \mathcal{J}$ *of a valid task allocation holds that:*

$$U(\mathcal{J}'_C) \leq U(\mathcal{J}_C)$$

A task allocation can be valid but not efficient if tasks have been processed that are not that efficient and, therefore, the utility gain for the whole set of agents is not as high as it could be. The utility gain is also called the *social welfare* (Weiss, 1999). To optimize the social welfare, i.e. the sum of all profits of all agents, agents have to reason about the job-set to process. If the jobs are generated and processed as it is done in our scenario, then the agents are not able to reason about it as they are not aware of all jobs that are allocated to them.

Let us now investigate the required skill amount per agent in an analytical way. We always have $10 \cdot |\mathcal{A}|$ jobs per simulation step, which leads to an average allocation of 10 jobs per agent. Each job consists of three tasks. Every task requires one of five skills and each agent is equipped with a single skill. These values are given in the basic setup in most of our experiments. Let us assume, that agent has 15 neighbors on average. Then, the tasks of 16 agents are in the agent's vicinity. This leads to the conclusion that on average

$$16 \cdot 10 \cdot 3 \cdot \frac{1}{5} = 96 \tag{7.2}$$

tasks are in the vicinity of agent a with the correct skill as on average every fifth task requests the skill that this agent can provide. However, as $30{,}000$ tasks are generated in each round and if we consider a set of 1000 agents, then every agent should process 30 tasks on average. The 96 tasks that have been calculated in Equation 7.2 do not consider that there are other agents in the job holder's vicinity that also provide the requested skill. Therefore, processing 30 tasks per agent is more realistic.

The next section provides experimental results concerning the introduced capacities.

7.3 Experimental Results

In the experimental analysis, we first consider the required skill amount per agent to fulfill a sufficiently large number of jobs, i.e. more than 80%. The second part considers the reasoning about the job-set if it is known to maximize the social welfare. The last part will deal with reasoning about unknown job-sets based on the efficiency strategy introduced in this section.

7.3.1 Processing Jobs with Capacities

The setup of the experiments is basically the same as the basic configuration of Chapter 5. Additionally, each task requires one skill unit and leads to a reward of one utility unit. Therefore, each job processing is rewarded with a payoff of 3 and needs exactly three skill units, as each job consists of exactly three tasks. Thus, the global skill requirements are $30{,}000$ skills in each simulation step. We will derive the available skill units per agent through taking approximated ratios of required and supplied skills. As the provided capacities are natural numbers, we round the value, if the ratio results in a rational number. For the ratios we use

$$\mathsf{ratio} \in \{2.0, 1.5, 1.0, 0.75, 0.5, 0.25\} \tag{7.3}$$

which leads to agents' capacities of

$$\forall a \in \mathcal{A} : \mathsf{cap}_a \in \{60, 45, 30, 22, 15, 8\} \tag{7.4}$$

Figures 7.1–7.3 present the results of these experiments. As can be seen, the percentage of completed jobs grows with larger capacities of the agents (cf. Figure 7.1). For all capacities with $\mathsf{cap}_a > 30$ the results are nearly identical. For $\mathsf{cap}_a = 30$ slightly worse results are achieved. Thus, it follows that if the agents provide the same number of skill units as it is required for the job-set, the agents are able to process a high number of jobs. If they have twice as many skill units as needed, no improvement can be achieved. For all other capacities it is observable that each experiment converged to a percentage that is comparable to the capacity difference.

This can also be seen in Figure 7.2 where the percentage of jobs that have not been processed because of missing capacities is shown. All experiments with $\mathsf{cap}_a \leq 30$ show that the reason for lower percentages of completed jobs is based on this fact. As a conclusion, we state that in most cases the required skill as well as a cooperative neighbor was available, as we only count the missing capacity case, if the other two cases would not cause a job to be uncompleted. We also observe that about 7% of the jobs cannot be processed because of missing capacities if the capacity is set to $\mathsf{cap}_a = 30$. If we consider these agents, we see in Figure 7.3 that in the adaptive case nearly the maximum possible number of jobs is processed with this ratio of skill units. The horizontal line represents a system, where the agents always cooperate with each other. Thus, the amount of jobs that is not processed is caused by the fact that not all skills are in the agents' vicinity, besides missing capacities. The adaptive case is slightly worse. We assume that this is the result of sub-optimal social networking steps as the social networking is not directed into good neighbor selection.

As a conclusion we state that the mechanism can also deal with capacity constraints for the agents and that this has large influence on the system's performance if constraints have to be fulfilled. In the next sections we consider more advanced agents that are able to reason about the order of the jobs to be processes with the aim of achieving high social welfare.

7.3.2 Reasoning about Known Job-Sets

In this section, we examine reasoning about known job-sets. The jobs are generated and allocated to the agents but the processing will not start until the generation of jobs is completed. Then, the agents are able to sort their jobs based on the efficiency. To get different efficiencies for the jobs we again consider jobs consisting of three tasks but for the payoff of a task we use

$$q_t \in \mathcal{U}[1, 20] \tag{7.5}$$

and for the task's capacity requirements we use

$$\mathsf{req}(t) \in \mathcal{U}[1, 10] \tag{7.6}$$

Thus, it follows that the expected requirement for a task is 5.5. As we consider a system with 10000 jobs each consisting of 3 tasks, we have an expected total requirement of 165000 skill units. Therefore, for the capacity of the agents we set $\mathsf{cap}_a = 165$ to have a ratio of provided and required skill units of 1.0 as we again consider a system with 1000 agents. Figures 7.4 and

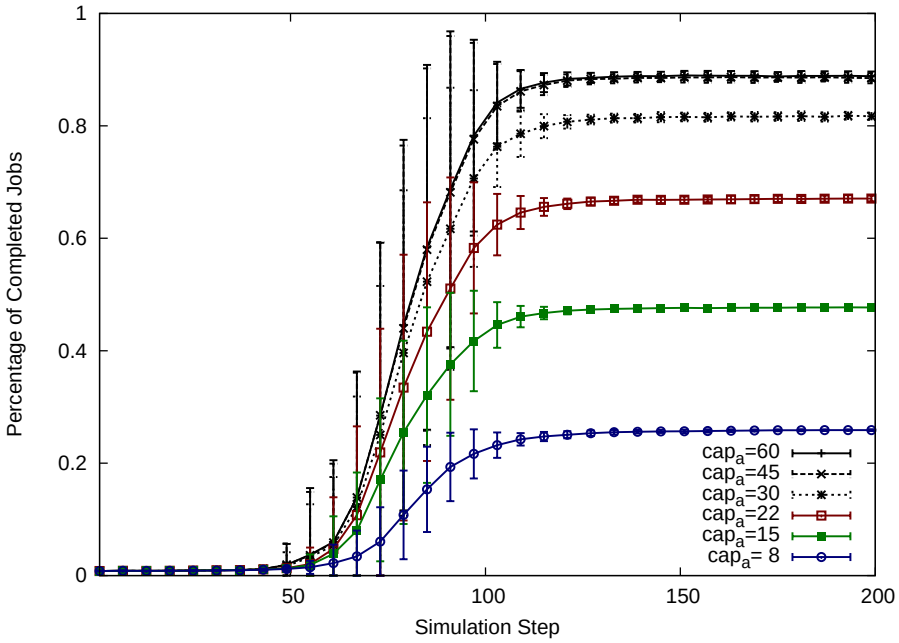


Figure 7.1: Percentage of completed jobs for different agent capacities.

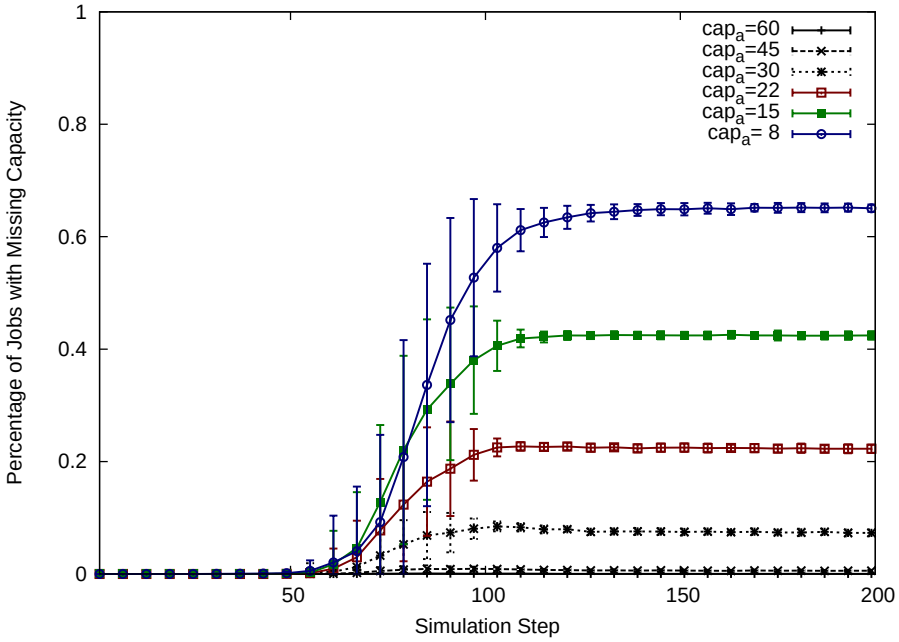


Figure 7.2: Percentage of jobs for which no capacity was left for different agent capacities.

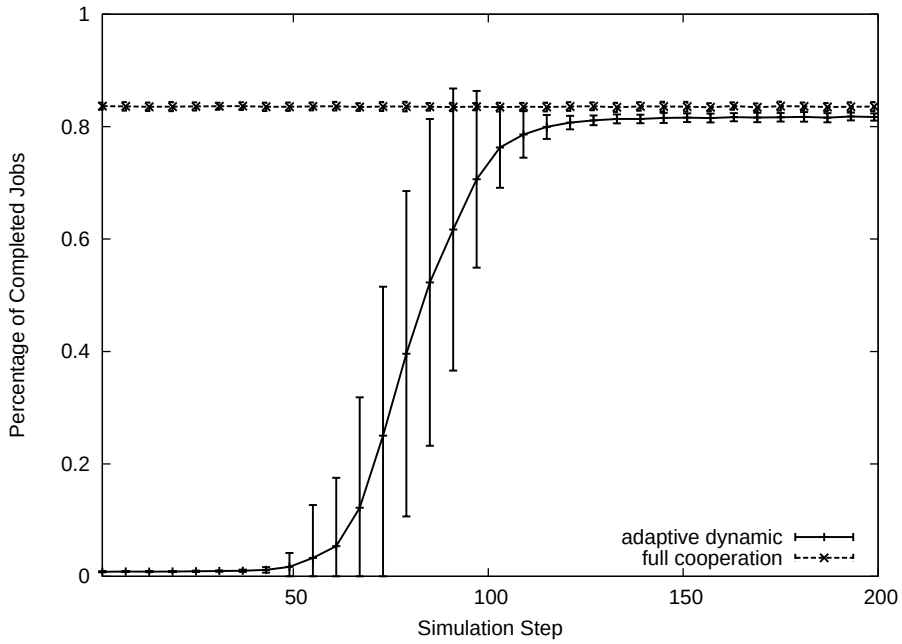


Figure 7.3: Comparison of agents having capacity of 30 for the adaptive and fully cooperative case.

7.5 show the results of these experiments. As can be observed the percentage of completed jobs is not influenced by the reasoning process. With about 82% at the end the same high level is reached in both settings (cf. Figure 7.4). The main result of reasoning about the job-set can be seen in Figure 7.4. If the agents consider the efficiencies of the jobs, the social welfare is much higher than in the case where they do not reason about the efficiency. There is no large improvement, but the difference is significant as the error-bars show.

As a conclusion, we claim that reasoning about the efficiencies increases the social welfare of the system but has only marginal influence on the percentage of completed jobs. The point is that the agent have to be enabled to reason about the jobs. They have to know about the jobs that are allocated to them in the whole simulation step. In the case where the agents do not consider the efficiency the jobs are generated and processed in one step. No knowledge is needed to process such a high number of jobs. The achieved social welfare is only about 1% less than in the case of knowing all jobs. So we claim that the effort that is needed to reason about the jobs does not pay back.

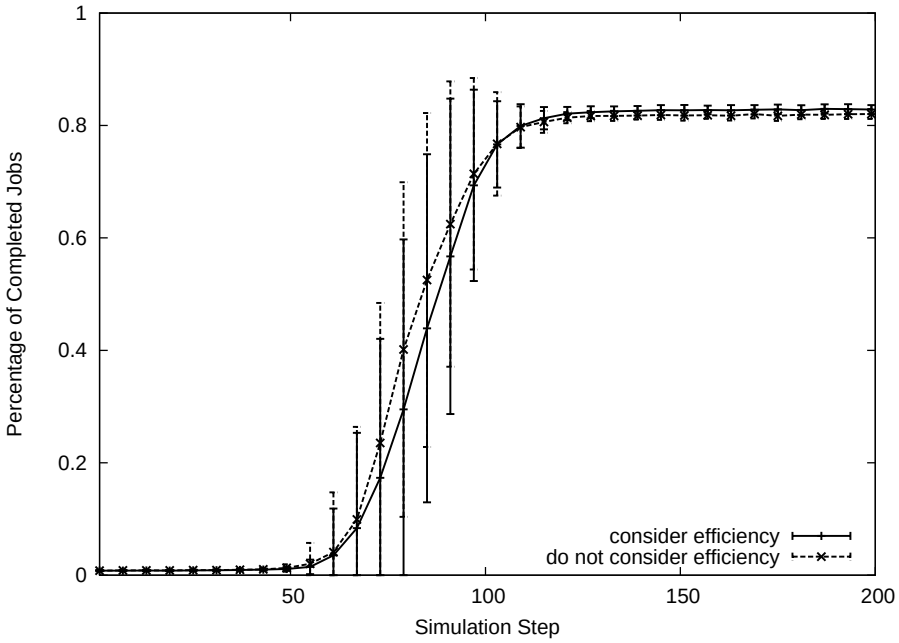


Figure 7.4: Percentage of completed jobs with and without reasoning about the job-set.

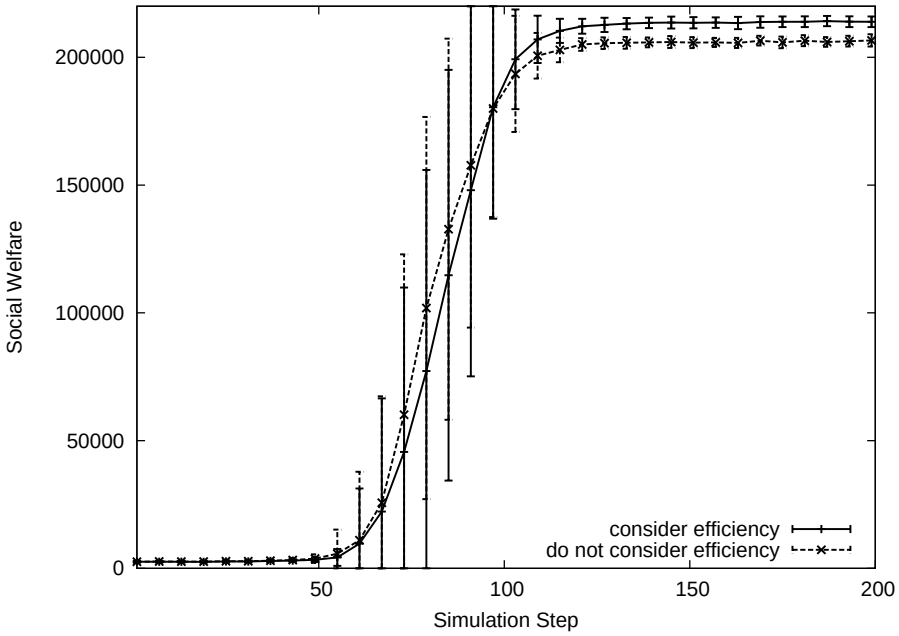


Figure 7.5: Achieved social welfare with and without reasoning about the job-set.

7.4 Conclusion

In this chapter, we introduced capacity constraints to the considered multiagent system. We showed how the formal model has to be extended and what properties the resulting formal model has. The properties *locality* and *completeness* are fulfilled by the algorithms presented in Chapter 3, which was proven in this chapter. We also proposed a slight change in the algorithm to achieve correct task allocations.

In the experimental analysis we showed that the system can deal with capacity constraints and that a ratio of 1.0—concerning the provided and required skills—is sufficient to produce high levels of completed jobs. For the same ratio we showed that if the agents have much more knowledge and know all jobs of a simulation step in advance they may reason about the jobs' efficiencies in order to achieve a larger social welfare. For this case, we showed that there is no great improvement compared to the costs of knowing the whole job set. Therefore, we claim that the algorithm is able to produce high job completion rates and high levels of social welfare without knowing the job set in advance.

8 Strategies for Social Networking

In this chapter, we deal with intelligent social networking strategies. In previous chapters, the agents changed their neighborhood by randomly rewiring connections. The results showed that one reason for non-processed jobs is that specific skills have been missing in the vicinity of the agent. Thus, a new neighbor which provides this skill would be beneficial for the agent instead of a randomly chosen agent. We are interested in investigating strategies that promote the emergence of cooperation and that help the agents to fulfill high numbers of jobs.

The main results of this chapter are the following:

- Random selection strategy is outperformed by the *skill-based* and *smart strategy*.
- Skill-based and smart strategy reach higher levels of completed jobs.
- The convergence speed is much faster for the skill-based and smart strategy.
- Random selection and skill-based strategy need the set of all agents to be known to a single agent (global knowledge).
- Only the smart strategy can be called local as no global knowledge is needed.

8.1 Advanced Strategies

The main reason for reaching a job completion rate below 100% in our approach is that for most of the non-processed jobs the required skill was missing in the neighborhood of the agent. Therefore, we propose some strategies for performing the social networking step.

Random Strategy The random strategy is the strategy used in previous chapters. It simply replaces r randomly chosen uncooperative neighbors by r agents from the whole population. If the number of uncooperative neighbors is less than r then all uncooperative neighbors are replaced by the same number of randomly chosen agents from the population.

Skill-based Strategy In the skill-based strategy the agents replace neighbors with respect to the skills in the agent's vicinity. This means they use a history to record which skills have been missing in the current simulation step. Then they choose agents as new neighbors that provide these skills, or, if no skill was missing, that provide skills with minimal occurrence in their current neighborhood. As replaceable neighbors they choose those (uncooperative) neighbors that provide a skill that is very often present in the neighborhood. This strategy is formalized in Algorithm 8.1, which is now described in detail.

First, the agents calculate the set of missing skills MS that have not been provided by their neighbors based on their history his. If this set is empty, i.e. all requests have never been declined because of a missing skill, the agent selects the skill(s) with the minimal occurrence in its neighborhood. Then, the agent calculates the set of skills MOS that have maximal occurrence and the set of uncooperative neighbors $\mathsf{UC}_{\mathsf{MOS}}$ that have skills of MOS. If the cardinality of this set is less then the number of neighbors r that should be replaced then another set of agents is randomly selected from the agent's neighborhood with cardinality $r - |\mathsf{UC}_{\mathsf{MOS}}|$. This set is called MISS. The agents that have to be replaced are randomly selected from the set $\mathsf{UC}_{\mathsf{MOS}}$ or are all agents from $\mathsf{UC}_{\mathsf{MOS}}$ and MISS. The set of agents ADD that should be added to agent a's neighborhood are randomly selected from the whole population but their skill sets have to contain at least one skill out of the set MS. The last step is to replace all agents from REM by the set of agents from ADD.

Smart Replacement The smart strategy differs from the skill-based strategy in one aspect. New neighbors are not arbitrary agents from the set of agents but they are neighbors of the current neighbors. The idea is that they probably will cooperate with the agent as they cooperated with an agent that is similar to itself, i.e. a current neighbor. The algorithm for the smart strategy is given in Algorithm 8.2, which is described now in detail.
As can be seen the algorithm is similar to the former presented algorithm. The smart strategy follows the skill-based strategy but there is another set of possible neighbors POS created. This set contains all neighbors of neighbors, which provide at least one of the skills of the set MS and that are no neighbors of agent a, yet. From this set the r new neighbors are randomly selected.

Each of these strategies follows specific intentions. The random strategy is the easiest strategy. The problem of that strategy is that no knowledge is used for selecting the neighbors that are removed from the neighborhood or for the selection of the new neighbors. The agents need access to the whole set of agents or have to use advanced selection strategies like gossig-based peer sampling (Jelasity et al., 2004, 2007) to select randomly agents out of the partially observable set of agents. The same holds for the skill-based strategy. However, this strategy considers the requests that have been declined in the past because of missing skills. The intention is to get new neighbors that have these skills and to remove (uncooperative) neighbors that are no longer needed as their skills occur often in the agent's neighborhood. The third strategy, the smart replacement, does basically the same as the skill-based strategy but only takes the set of neighbors of neighbors as possible new neighbors. Thus, this strategy is additionally intended to care about the cooperation relationships as agents that are neighbors of that agent's neighbors will probably have ratings that lead to receiving help from them. This strategy follows a local view as the set of all agents does not have to be known to the agents.

A commonality of all three strategies is that the agents do not care about having all possible skills in their vicinities. For the random selection strategy this is obvious. For the other two strategies the agents only care about those skills that have been requested by the tasks but which could not be provided. Additionally, we slightly changed the condition of being satisfied for the agents, which do not follow the random strategy. The previous condition for agent a being

Algorithm 8.1 Skill-based replacement algorithm executed by agent a.

Input: the set of neighbors $\mathcal{N}_a$ of agent a
number of neighbors to replace r
set of all agents $\mathcal{A}$
history of missing skills his

procedure SKILLBASEDREPLACEMENT($\mathcal{N}_a$, r, $\mathcal{A}$, his)
  $\mathsf{MS} \leftarrow$ set of missing skills from the history his
  if $\mathsf{MS} = \emptyset$ **then**
    $\mathsf{MS} \leftarrow$ set of skills with min occurrence in neighborhood
  end if
  $\mathsf{MOS} \leftarrow$ set of skills with max occurrence in neighborhood
  $\mathsf{UC}_{\mathsf{MOS}} \leftarrow$ set of uncooperative neighbors b with skill set $\mathcal{S}_b \subseteq \mathsf{MOS}$
  if $|\mathsf{UC}_{\mathsf{MOS}}| \geq r$ **then**
    $\mathsf{REM} \leftarrow$ set of r randomly selected agents of $\mathsf{UC}_{\mathsf{MOS}}$
  else
    $\mathsf{MISS} \leftarrow$ set of $r - |\mathsf{UC}_{\mathsf{MOS}}|$ randomly selected agents of $\mathcal{N}_a \setminus \mathsf{UC}_{\mathsf{MOS}}$
    $\mathsf{REM} \leftarrow \mathsf{UC}_{\mathsf{MOS}} \cup \mathsf{MISS}$
  end if
  $\mathsf{ADD} \leftarrow$ set of r randomly selected agents b of $\mathcal{A} \setminus \mathcal{N}_a$ with $\mathcal{S}_b \cap \mathsf{MS} \neq \emptyset$
  $\mathcal{N}_a \leftarrow (\mathcal{N}_a \setminus \mathsf{REM}) \cup \mathsf{ADD}$
end procedure

unsatisfied was

$$a \notin \mathcal{E}_a \wedge \exists b \in \mathcal{N}_a : (b, a) \notin \mathcal{C} \tag{8.1}$$

which we changed to

$$a \notin \mathcal{E}_a \wedge \exists b \in \mathcal{N}_a : (b, a) \notin \mathcal{C} \vee \mathsf{his} \neq \emptyset. \tag{8.2}$$

Both conditions make the agent a unsatisfied if it is not in its elite set $\mathcal{E}_a$ and if there is at least one neighboring agent that would not cooperate with agent a. If the agent does not follow the random strategy for the social networking step it is also unsatisfied if there is at least one request which had to be declined in the current simulation step because of a missing skill. Thus, the agent wants to get these skills into its vicinity. In the next section we will present experimental results.

8.2 Experimental Results

The experiments were conducted with $\mathsf{Pr}_{\mathcal{N}} = 0.01$ as the probability for executing the social networking step as we did in previous experiments and $r = 1$ agent is replaced in one social networking step. The results are shown in Figures 8.1–8.3.

Algorithm 8.2 Smart replacement algorithm executed by agent a.

Input: the set of neighbors $\mathcal{N}_a$ of agent a
number of neighbors to replace r
history of missing skills his

1: **procedure** SMARTREPLACEMENT($\mathcal{N}_a$,r,his)
2: MS $\leftarrow$ set of missing skills from his
3: **if** MS $= \emptyset$ **then**
4: MS $\leftarrow$ set of skills with min occurrence in neighborhood
5: **end if**
6: MOS $\leftarrow$ set of skills with max occurrence in neighborhood
7: $\mathsf{UC}_{\mathsf{MOS}} \leftarrow$ set of uncooperative neighbors b with skill set $\mathcal{S}_b \subseteq$ MOS
8: **if** $|\mathsf{UC}_{\mathsf{MOS}}| \geq r$ **then**
9: REM $\leftarrow$ set of r randomly selected agents of $\mathsf{UC}_{\mathsf{MOS}}$
10: **else**
11: MISS $\leftarrow$ set of $r - |\mathsf{UC}_{\mathsf{MOS}}|$ randomly selected agents of $\mathcal{N}_a \setminus \mathsf{UC}_{\mathsf{MOS}}$
12: REM $\leftarrow \mathsf{UC}_{\mathsf{MOS}} \cup$ MISS
13: **end if**
14: POS $\leftarrow \{c \in \mathcal{N}_b \mid b \in \mathcal{N}_a \wedge c \notin \mathcal{N}_a \wedge \mathcal{S}_c \cap \mathsf{MS} \neq \emptyset\}$
15: ADD $\leftarrow$ set of r randomly selected agents of POS
16: $\mathcal{N}_a \leftarrow (\mathcal{N}_a \setminus \mathsf{REM}) \cup \mathsf{ADD}$
17: **end procedure**

As can be seen, the percentage of completed jobs is drastically effected if the strategy for social networking does not follow the random strategy (cf. Figure 8.1). In both cases—skill-based strategy and smart strategy—we can observe, that the percentage of completed jobs grows earlier and a bit faster than in the random case and the achievable level is significantly higher. No significant difference is observable between the smart strategy and the skill-based strategy.

Figure 8.2 shows the percentage of jobs that had not been completed due to missing skills. It can be seen that in all three setting the percentage grows. However, for the skill-based and smart strategy we can observe that this percentage does not reach such a high level as if the random selection strategy is used, which is about twice the high. This is due to the fact that the agents reason about what skills the new neighbors should have. Nevertheless, it can happen that an agent loses skills in its vicinity because that agent could cut the link by itself. There is no negotiation about what links can be rewired. The percentage of uncompleted jobs due to missing cooperative neighbors in Figure 8.3 shows that the agents following the skill-based or smart strategy faster create cooperative groups. The reason is, that their satisfaction is also effected by the skills in their vicinity and, thus, they tend to be unsatisfied as the agents following the random strategy do not care about this fact.

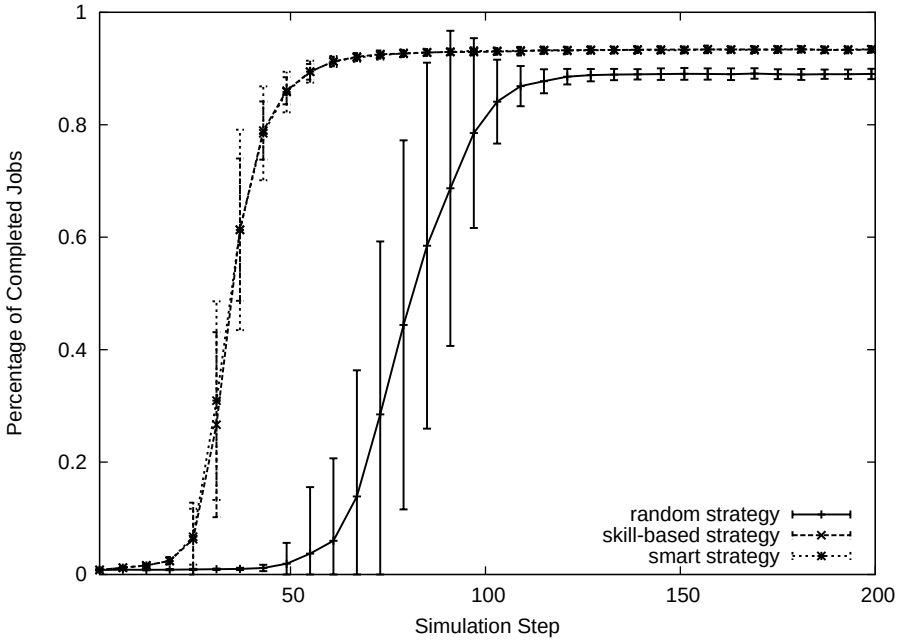


Figure 8.1: Percentage of completed jobs for different social networking strategies.

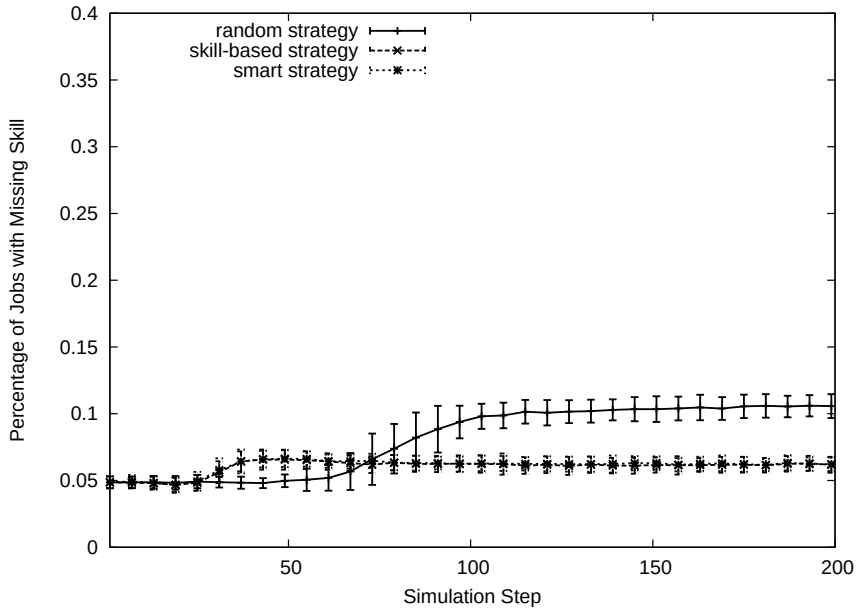


Figure 8.2: Percentage of missing skills for different social networking strategies.

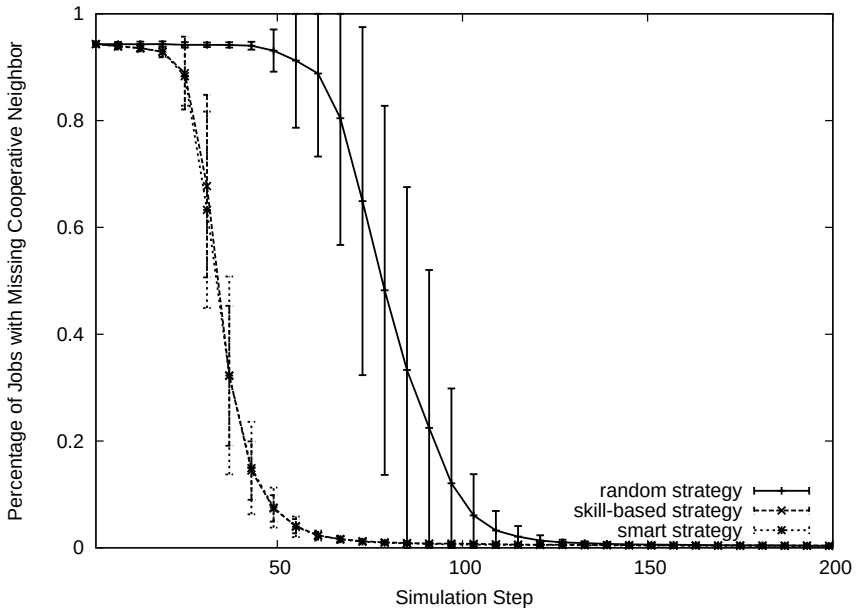


Figure 8.3: Percentage of missing cooperative neighbors for different social networking strategies.

8.3 Conclusion

We have presented different strategies for the social networking phase. Namely, the random strategy, the skill-based strategy and the smart strategy. The skill-based and smart strategy consider required skills in their vicinity in order to select new neighbors. The difference between both strategies is that the smart strategy does not use any kind of global knowledge, whereas the skill-based strategy needs to now which agents are in the system as the set of possible neighbors is basically the whole population.

We have shown that the more advanced strategies—skill-based and smart strategy—lead to nearly identical results. Both outperform the random strategy in speed and height of reachable levels of completed jobs. However, the prize is the larger knowledge that is required to follow these strategies. In both strategies, the required skills of tasks, that lead to rejected jobs because of absence of that skill, have to be recorded. The smart strategy has the advantage that it can be called a local strategy as only the neighbors of neighbors have to be known to the agents. This can be achieved through message transfer and thus no global knowledge is used for this strategy.

9 Conclusion and Future Work

9.1 Conclusion

In this thesis, we presented new local adaptation-based learning mechanisms that lead to high levels of cooperation in an emergent way. No rules are encoded to the agents that they have to cooperate in a certain way. The decisions to cooperate are solely based on similarities of agents as it can be observed in human cooperative behavior.

We formally defined the considered multiagent system and a benchmark scenario that describes a job/task model in Chapter 3. The agents have to search for cooperative neighbors if tasks of an assigned jobs require skills that are not provided by the agent. In the formal analysis (Chapter 4) we considered the required neighborhood size in order to obtain a high probability of having all required skills in the vicinity of the agents. Additionally, we analyzed the convergence of the proposed learning algorithm concerning value propagation through the network. We showed that the proposed algorithm has linear complexity in the number of neighbors from the agents' local view. In a rigorous experimental analysis in Chapter 5 we examined the influence of the mechanism's parameters and showed that the approach is scalable and robust against changes of the population size.

In Chapter 6, we considered different network structures and experimentally showed that highly dynamical networks tend to degenerate to randomly generated networks. In order to have a more realistic model, we extended the model in Chapter 7, where we introduced capacity constraints. We showed, that the approach can deal with restricted numbers of skills available to each agent. The social networking, which is a core aspect of the proposed mechanism, was examined in Chapter 8 where we presented different strategies to actively change the neighborhood of agents. We presented that more advanced strategies which require more knowledge than a random selection strategy perform best. However, the drawback of these strategies are the knowledge requirements of a single agent.

To summarize, the main contributions of this thesis are:

- a new local adaptation-based learning approach
- approach inspired from social science based on agents' similarities
- rigorous analysis—formally and experimentally—of the proposed mechanism
- insights into emergent cooperation based on local decisions
- simplicity, adaptivity and simple knowledge of the agents

9.2 Future Work

For future work there are several directions that are worth to consider. The following extensions seem to be most promising:

Computational Trust Mechanisms The decision to cooperate could be enriched with the help of computational trust mechanisms. Besides the proposed mechanism based on agents' similarities, previous experiences with specific agents could be taken into account. An agent could rate previous behavior of its neighbors and decline a request if this neighbor had declined requests in the early past.

Reputation Mechanism For the selection of new neighbors a reputation mechanism seems to be promising. If an agent selects another agent to replace an old neighbor the agent's reputation value could be taken into account. The agent could ask its neighbors if someone has experiences with the other agent.

Mutual Networking Decisions In the proposed mechanism incoming connection requests have to be answered positively. The agents are not allowed to decline a request. It would be interesting if the agents would be able to decline based on some mechanisms, e.g. trustworthiness or reputation. Negotiation about rewiring connections would also be an interesting expansion of the approach.

Unreliable Agents As the proposition-values are a core aspect of the proposed algorithm one could think about agents lying about their values. This could lead to agents exploiting the mechanism to gain more cooperative neighbors. One possibility of exploitation would be if an agent, that asks for help, transmits another value-vector in order to be sure to get the other agent to cooperate. Another exploitation would be that an ideal agent—if it can sense that another agents wants to adapt to it—could transmit a wrong value-vector in order to not to have to cooperate with the adapting agent. It would be interesting to see if the mechanism can deal with such additional aspects and what has to be changed to make the approach to work with unreliable agents.

(Dynamic) Quality of Service In the proposed scenario each agent can handle tasks with the same quality of service (QoS) and the QoS does not influence the achieved payoff for a job. It would be possible to extent the model that QoS could differ on different dimensions. One possibility would be that the quality of executing a task differs from agent to agent but is constant for a specific agent over time. Then it would be the challenge of finding good executers for the tasks. Another possibility would be to have changing QoS for a single agent over time. The process of finding executers for the tasks would be enriched by another challenge as the dynamics of quality-changes might not be predictable. Then it would be interesting to find intelligent mechanisms that may help to receive a high average value for the execution quality.

Node Failure One assumption of the considered approach is that tasks that have been allo-

cated will never need a re-allocation as none of the nodes may become unreachable after the allocation step. Thus, one could think of two different scenarios that may cause an execution to fail. A single node may be reachable but dead and, thus, could no longer execute the tasks although the cooperation request was positively answered. The second aspect could be that the network may change within the task allocation and processing step and thus a node may no longer be locally reachable. Then, strategies have to be constructed that help the system deal with such problems. One possibility would be that each agent tries to prevent such cases by considering two different agents for the same task such that the task could be send to the second agent if the first fails.

Additional Application Analysis In this thesis the proposed mechanism is solely tested in a job/task allocation scenario. Additional applications could be studied in which the mechanism can be integrated. This would help to get more insights into the class of problems to which the approach is applicable. Especially the properties of these problems would be of great interest. Some other possible application scenarios would be peer-to-peer file sharing system or the simulation of supply chain networks.

Index

List of Figures

List of Tables

Own Publications

Eberling, M., 2009: Towards determining cooperation based on multiple criteria. *KI 2009*, B. Mertsching, M. Hund, and M. Z. Aziz, Eds., Springer, Berlin / Heidelberg, Lecture Notes in Computer Science, Vol. 5803, 548–555.

Eberling, M. and H. Kleine Büning, 2010a: Convergence analysis of a multiagent cooperation model (extended version). Tech. Rep. TR-RI-10-321, University of Paderborn.

Eberling, M. and H. Kleine Büning, 2010b: Self-adaptation strategies to favor cooperation. *Agent and Multi-Agent Systems: Technologies and Applications, 4th International Conference, AMSTA-10*, P. Jedrzejowicz, N. T. Nguyen, R. J. Howlett, and L. C. Jain, Eds., Springer, Heidelberg, Lecture Notes in Artificial Intelligence, Vol. 6071, 223–232.

Eberling, M. and H. Kleine Büning, 2011: Convergence analysis of a multiagent cooperation model. *Proceedings of the Third International Conference on Agents and Artificial Intelligence (ICAART 2011)*, 167–172.

Heinrich, S., S. Wermter, and M. Eberling, 2011: Determining cooperation in multiagent systems with cultural traits. *Proceedings of the Third International Conference on Agents and Artificial Intelligence (ICAART 2011)*.

Huber, D., M. Eberling, C. Laroque, and W. Dangelmaier, 2008: Stochastic generation of discrete-event simulation models. *Proceedings of the Tenth International Conference on Computer Modeling and Simulation (UKSIM'08)*.

Laroque, C., B. Urban, and M. Eberling, 2010: Parameteroptimierung von materialflusssimulationen durch partikelschwarmalgorithmen. *Proceedings of the German Multikonferenz Wirtschaftsinformatik 2010 (MKWI'10)*.

Lettmann, T., M. Baumann, M. Eberling, and T. Kemmerich, 2011: Modeling agents and agent systems, accepted for publication in Transactions on Computational Collective Intelligence, Springer.

Priesterjahn, S. and M. Eberling, 2008: Imitation learning in uncertain environments. *Proceedings of the 10th International Conference on Parallel Problem Solving From Nature (PPSN'08)*, G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, Eds., Springer, Berlin / Heidelberg, Lecture Notes in Computer Science, Vol. 5199, 950–960.

Priesterjahn, S., A. Weimer, and M. Eberling, 2008: Real-time imitation-based adaptation of gaming behaviour in modern computer games. *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*.

Schmidt, T., M. Eberling, and H. Kleine Büning, 2010: The effects of local trust cooperation in multiagent systems. *Agent and Multi-Agent Systems: Technologies and Applications, 4th International Conference, AMSTA-10*, P. Jedrzejowicz, N. T. Nguyen, R. J. Howlett, and L. C. Jain, Eds., Springer, Heidelberg, Lecture Notes in Artificial Intelligence, Vol. 6071, 233–242.

Bibliography

Abdallah, S. and V. Lesser, 2004: Organization-based cooperative coalition formation. *Proceedings of the IEEE/ WIC/ ACM International Conference on Intelligent Agent Technology (IAT 2004)*, 162–168.

Abramowitz, M. and I. A. Stegun, (Eds.) , 1972: *Handbook of Mathematical Functions (10th printing)*. Dover Publications.

Adar, E. and B. Huberman, 2000: Free riding on gnutella. *First Monday*, **5 (10)**, 2–13.

Adler, M. R., A. B. Davis, R. Wehmayer, and R. W. Worrest, 1989: *Conflict resolution strategies for nonhierarchical distributed agents*, 139–161. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Albert, R. and A. L. Barabási, 2002: Statistical mechanics of complex networks. *Reviews of modern physics*, **74 (1)**, 47–97.

Association for Computing Machinery, 1998: The acm computing classification system. ACM, `http://www.acm.org/about/class/ccs98-html`.

Axelrod, R., 1984: *The Evolution of Cooperation*. Basic Books.

Axelrod, R., 1997: *The complexity of cooperation: Agent-based models of competition and collaboration*. Princeton University Press.

Azevedo, C. R. B. and V. S. Gordon, 2009: Adaptive terrain-based memetic algorithms. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO 2009)*, ACM, 747–754.

Barabási, A. L. and R. Albert, 1999: Emergence of scaling in random networks. *Science*, **286 (5439)**, 509–512.

Barr, A. and E. A. Feigenbaum, (Eds.) , 1982: *The handbook of artificial intelligence*. William Kaufmann.

Barrat, A. and M. Weigt, 2000: On the properties of small-world network models. *The European Physical Journal B-Condensed Matter and Complex Systems*, **13 (3)**, 547–560.

Bell, G., 2008: *Selection: the mechanism of evolution*. Oxford University Press, USA.

Berkowitz, L. and J. Macaulay, 1970: *Altruism and helping behavior: social psychological studies of some antecedents and consequences*. Academic Press.

Billard, A. and M. J. Matari, 2001: Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, **37 (2)**, 145–160.

Blackmore, S., 1998: Imitation and the definition of a meme. *Journal of Memetics-Evolutionary Models of Information Transmission*, **2 (11)**, 159–170.

Boloni, L. and D. C. Marinescu, 2000: *An object-oriented framework for building collaborative network agents*, 31–64. Kluwer Academic Publishers.

Bonabeau, E., M. Dorigo, and G. Theraulaz, 1999: *Swarm intelligence: from natural to artificial systems*. Oxford University Press.

Brualdi, R. A. and K. P. Bogart, 1977: *Introductory combinatorics*. North-Holland Amsterdam.

Buchanan, D. and A. Huczynski, 1997: *Organisational Behaviour: An Introductory Text*. Prentice-Hall.

Bulka, B., M. E. Gaston, and M. desJardins, 2007: Local strategy learning in networked multi-agent team formation. *Autonomous Agents and Multi-Agent Systems*, **15 (1)**, 29–45.

Castelfranchi, C., R. Conte, and M. Paolucci, 1998: Normative reputation and the costs of compliance. *Journal of Artificial Societies and Social Simulation*, **1 (3)**.

Cecconi, F. and D. Parisi, 1998: Individual versus social survival strategies. *Journal of Artificial Societies and Social Simulation*, **1 (2)**.

Chan, C. and H. Leung, 2009: Multi-auction Approach for Solving Task Allocation Problem. *Multi-Agent Systems for Society*, Springer, Lecture Notes in Artificial Intelligence, Vol. 4078, 240–254.

Chang, H. W. D. and W. J. B. Oldham, 1995: Dynamic task allocation models for large distributed computing systems. *IEEE Transactions on Parallel and Distributed Systems*, **6 (12)**, 1301–1315.

Chapman, A. C., R. A. Micillo, R. Kota, and N. R. Jennings, 2009: Decentralised dynamic task allocation: a practical game: theoretic approach. *Proceedings of The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Vol. 2, 915–922.

Chapman, A. C., R. A. Micillo, R. Kota, and N. R. Jennings, 2010: Decentralized dynamic task allocation using overlapping potential games. *The Computer Journal*, **53 (9)**, 1462–1477.

Chase, I., 1991: Vacancy chains. *Annual Review of Sociology*, **17**, 133–154.

Chase, I. D., M. Weissburg, and T. H. Dewitt, 1988: The vacancy chain process: a new mechanism of resource distribution in animals with application to hermit crabs. *Animal behaviour*, **36 (5)**, 1265–1274.

Cicirello, V. A. and S. F. Smith, 2001: Wasp-like agents for distributed factory coordination. Tech. Rep. CMU-RI-TR-01-39, Robotics Institite, Carnegie Mellon University, Pittsburgh.

Clarke, E. H., 1971: Multipart pricing of public goods. *Public Choice*, **11 (1)**, 17–33.

Conte, R. and C. Castelfranchi, 1993: Simulative understanding of norm functionalities in social groups. *Simulating Societies-93: Pre-proceedings of the 1993 International Symposium on Approaches to Simulating Social Phenomena and Social Processes*.

Dahl, T. S., M. J. Mataric, and G. S. Sukhatme, 2003: Multi-robot task-allocation through vacancy chains. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2003)*, IEEE, Vol. 2, 2293–2298.

Dai, T., G. Lai, K. Sycara, and R. Glinton, 2009: On the value of commitment flexibility in dynamic task allocation via second-price auctions. *Proceedings of the AAMAS 2009 Workshop on Multi-Agent Sequential Decision-Making in Uncertain Domains*, 22–29.

Dawkins, R., 1976: *The Selfish Gene*. Oxford University Press.

Dawkins, R., 1982: *The Extended Phenotype*. Oxford University Press.

de Vries, S. and R. Vohra, 2003: Combinatorial auctions: A survey. *INFORMS Journal on Computing*, **15 (3)**, 284–309.

de Weerdt, M. and Y. Zhang, 2008: Preventing under-reporting in social task allocation. *Proceedings of the 10th workshop on Agent-Mediated Electronic Commerce (AMEC-X)*, H. La Poutre and O. Shehory, Eds., IFAAMAS.

de Weerdt, M., Y. Zhang, and T. Klos, 2007: Distributed task allocation in social networks. *Proceedings of the sixth International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS'07*, ACM.

Deng, X. and C. Papadimitriou, 1999: Decision-making by hierarchies of discordant agents. *Mathematical programming*, **86 (2)**, 417–431.

Dur, R. and H. Roelfsema, 2010: Social exchange and common agency in organizations. *Journal of Socio-Economics*, **39 (1)**, 55–63.

Durfee, E. H., 1988: *Coordination of distributed problem solvers*. Kluwer Academic Publishers.

Durfee, E. H., 1996: Planning in distributed artificial intelligence. *Foundations of Distributed Artificial Intelligence*, G. M. P. O´Hare and N. R. Jennings, Eds., John Wiley & Sons LTD, 231–245.

Durfee, E. H. and V. R. Lesser, 1987: Using partial global plans to coordinate distributed problem solvers. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI 1987)*, 875–883.

Eberling, M., 2009: Towards determining cooperation based on multiple criteria. *KI 2009*, B. Mertsching, M. Hund, and M. Z. Aziz, Eds., Springer, Heidelberg, Lecture Notes in Computer Science, Vol. 5803, 548–555.

Eberling, M. and H. Kleine Büning, 2010a: Convergence analysis of a multiagent cooperation model (extended version). Tech. Rep. TR-RI-10-321, University of Paderborn.

Eberling, M. and H. Kleine Büning, 2010b: Self-adaptation strategies to favor cooperation. *Agent and Multi-Agent Systems: Technologies and Applications, 4th International Conference (AMSTA 2010)*, P. Jedrzejowicz, N. T. Nguyen, R. J. Howlett, and L. C. Jain, Eds., Springer, Heidelberg, Lecture Notes in Artificial Intelligence, Vol. 6071, 223–232.

Eberling, M. and H. Kleine Büning, 2011: Convergence analysis of a multiagent cooperation model. *Proceedings of the Third International Conference on Agents and Artificial Intelligence (ICAART 2011)*, 167–172.

Eiben, A. E. and J. E. Smith, 2003: *Introduction to Evolutionary Computing*. Springer.

Encyclopædia Britannica, 2010: Learning. Encyclopædia Britannica Online, Retrieved 20 May 2011 from Britannica.com: `http://www.britannica.com/EBchecked/topic/333978/learning`.

Ephrati, E. and J. S. Rosenschein, 1993: Multi-agent planning as a dynamic search for social consensus. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI 1993)*, 423–429.

Erdõs, P. and A. Rényi, 1959: On random graphs. *Publicationes Mathematicae*, **6**, 290–297.

Erdõs, P. and A. Rényi, 1960: On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, **5**, 17–61.

Ferber, J., 1999: *Multi-Agent Systems: An Introduction to distributed artificial intelligence*. Addison Wesley Longman Inc.

Finin, T., R. Fritzson, D. McKay, and R. McEntire, 1994a: KQML as an agent communication language. *Proceedings of the third international conference on Information and knowledge management (CIKM 1994)*, ACM, 456–463.

Finin, T., J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, J. McGuire, S. Shapiro, and C. Beck, 1994b: Specification of the kqml agent-communication language. *The DARPA Knowledge Sharing Initiative External Interfaces Working Group*.

FIPA: Foundation for Intelligent Physical Agents, 2002: Fipa acl message structure specification. FIPA, `http://www.fipa.org/specs/fipa00061/SC00061G.pdf`.

Fog, A., 2005: Simulation models for biological and cultural evolution. *Proceedings of the Joint Symposium on Socially Inspired Computing (AISB 2005)*, 21–25.

Fogel, D. B., 2006: *Evolutionary computation: toward a new philosophy of machine intelligence*. Wiley-IEEE Press.

Gabora, L., 1996: A day in the life of a meme. *Philosophica*, **57 (1)**, 53–90.

Galliers, J. R., 1988: A theoretical framework for computer models of cooperative dialogue, acknowledging multi-agent conflict. Ph.D. thesis, Open University, UK.

Gardner, M., 1970: Mathematical games: The fantastic combinations of john conway's new solitaire game 'life'. *Scientific American*, **223 (4)**, 120–123.

Gasser, L., C. Braganza, and N. Herman, 1987: Mace: A flexible testbed for distributed ai research. *Distributed artificial intelligence*, **1**, 119–152.

Gaston, M. E., 2005: Organizational learning and network adaptation in multi-agent systems. Ph.D. thesis, University of Maryland.

Gaston, M. E. and M. desJardins, 2005a: Agent-organized networks for dynamic team formation. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS 2005)*, ACM, 230–237.

Gaston, M. E. and M. desJardins, 2005b: Agent-organized networks for multi-agent production and exchange. *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, 77–82.

Gaston, M. E. and M. desJardins, 2008: The effect of network structure on dynamic team formation in multi-agent systems. *Computational Intelligence*, **24 (2)**, 122–157.

Genereseth, M. R. and R. E. Fikes, 1992: Knowledge interchange format-version 3.0: reference manual. Tech. Rep. Logic-92-1, Stanford University.

Ghizzioli, R., S. Nouyan, M. Birattari, and M. Dorigo, 2004: An ant-based algorithm for the dynamic task allocation problem. Tech. Rep. TR/IRIDIA/2004-16, Université Libre de Bruxelles.

Gilbert, N. and R. Conte, (Eds.) , 1995: *Artificial Societies: the Computer Simulation of Social Life*. UCL Press.

Gillies, D. B., 1959: Solutions to general non-zero-sum games. *Contributions to the Theory of Games*, **4**, 47–85.

Goebels, A., 2007: *Agent Coordination Mechanism for Solving a Partitioning Task*. Logos.

Goldman, C. V. and J. S. Rosenschein, 1994: Emergent coordination through the use of cooperative state-changing rules. *Proceedings of the Eleventh National Conference On Artificial Intelligence (AAAI 1994)*, John Wiley & Sons LTD, 408–408.

Groves, T., 1973: Incentives in teams. *Econometrica*, **41 (4)**, 617–631.

Guestrin, C., D. Koller, and R. Parr, 2002: Multiagent planning with factored mdps. *Advances in neural information processing systems*, **14**, 1523–1530.

Gupta, A. K. and G. W. Greenwood, 1996: Static task allocation using (μ,λ) evolutionary strategies. *Information Sciences*, **94**, 141–150.

Haggith, M., R. Prabhu, C. J. P. Colfer, B. Ritchie, A. Thomson, and H. Mudavanhu, 2003: Infectious ideas: Modelling the diffusion of ideas across social networks. *Small-Scale Forestry*, **2 (2)**, 225–239.

Hales, D., 2001: Tag based co-operation in artificial societies. Ph.D. thesis, University of Essex.

Hales, D., 2002a: The evolution of specialization in groups. *Proceedings of the First International Workshop on Regulated Agent-Based Social Systems (RASTA 2002)*, G. Lindemann, D. Moldt, and M. Paolucci, Eds., Spinger, Berlin, Lecture Notes in Artificial Intelligence.

Hales, D., 2002b: Evolving specialisation, altruism and group-level optimisation using tags. *Multi-Agent-Based Simulation II*, J. Sichman, F. Bousquet, and P. Davidsson, Eds., Springer, Berlin, Lecture Notes in Artificial Intelligence 2581, Vol. 2934.

Hales, D., 2005: Emergent group level selection in a peer-to-peer network. *Proceedings of the European Conference on Complex Systems (ECCS 2005)*.

Harris, J. M., J. L. Hirst, and M. J. Mossinghoff, 2008: *Combinatorics and Graph Theory (2nd Edition)*. Springer.

Henrich, J., R. Boyd, and P. J. Richerson, 2008: Five misunderstandings about cultural evolution. *Human Nature*, **19 (2)**, 119–137.

Hinde, R. A. and J. Groebel, (Eds.) , 1991: *Cooperation and Prosocial Behavior*. Cambridge University Press.

Hodgson, G. M. and T. Knudsen, 2004: The firm as an interactor: firms as vehicles for habits and routines. *Journal of Evolutionary Economics*, **14 (3)**, 281–307.

Hodgson, G. M. and T. Knudsen, 2008: Information, complexity and generative replication. *Biology and Philosophy*, **23 (1)**, 47–65.

International Encyclopedia of the Social Sciences, 1968: Learning. encyclopedia.com, Retrieved 20 May 2011 from Encyclopedia.com: `http://www.encyclopedia.com/doc/1G2-3045000696.html`.

Jain, R. K., 1991: *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley/Interscience, URL `http://www.cse.wustl.edu/~jain/books/perfbook.htm`.

Jelasity, M., R. Guerraoui, A. M. Kermarrec, and M. Van Steen, 2004: The peer sampling service: experimental evaluation of unstructured gossip-based implementations. *Proceedings of the fifthth ACM/IFIP/USENIX International Conference on Middleware*, Springer, 79–98.

Jelasity, M., S. Voulgaris, R. Guerraoui, A. M. Kermarrec, and M. Van Steen, 2007: Gossip-based peer sampling. *Transactions on Computer Systems (TOCS)*, **25 (3)**.

Jennings, N. R., 1993: Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, **8 (3)**, 223–250.

Jennings, N. R., 2000: On agent-based software engineering. *Artificial Intelligence*, **117 (2)**, 277–296.

Jung, D. and D. Lake, 2008: Markets, hierarchies, and networks: an agent-based organizational ecology. *Annual Meeting of the American Political Science Association, Boston*.

Kaelbling, L. P., M. L. Littman, and A. R. Cassandra, 1998: Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, **101 (1-2)**, 99–134.

Kalenka, S. and N. R. Jennings, 1999: Socially responsible decision making by autonomous agents. *Cognition, agency, and rationality: Proceedings of the Fifth International Colloquium on Cognitive Science*, K. Korta, E. Sosa, and X. Arrazola, Eds., Kluwer Academic Publishers, 135–149.

Kirkpatrick, S., 1984: Optimizarion by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, **34 (5/6)**, 975–986.

Kittock, J. E., 1993: Emergent conventions and the structure of multi-agent systems. *Proceedings of the 1993 Santa Fe Institute Complex Systems Summer School*.

Kok, J. R., M. T. J. Spaan, and N. Vlassis, 2003: Multi-robot decision making using coordination graphs. *Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003)*, Vol. 3, 1124–1129.

Kramer, R. M. and M. B. Brewer, 1984: Effects of group identity on resource use in a simulated commons dilemma. *Journal of Personality and Social Psychology*, **46 (5)**, 1044–1057.

Kraus, S., O. Shehory, and G. Taase, 2003: Coalition formation with uncertain heterogeneous information. *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, 1–8.

Krebs, D., 1982: Psychological approaches to altruism: An evaluation. *Ethics*, **92 (3)**, 447–458.

Kuperman, M. N., M. Ballard, and F. Laguna, 2006: Dynamic domain networks. *The European Physical Journal B*, **50 (3)**, 513–520.

Lander, S., V. R. Lesser, and M. E. Connell, 1991: Conflict resolution strategies for cooperating expert agents. *Proceedings of the International Working Conference on Cooperating Knowledge Based Systems (CKBS 1990)*, 183–200.

Langdon, W. B., 2005: Pfeiffer—A distributed open-ended evolutionary system. *Proceedings of the Joint Symposium on Socially Inspired Computing (AISB 2005)*, 7–13.

Lerman, K., C. Jones, A. Galstyan, and M. J. Mataric, 2006: Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, **25 (3)**, 225–242.

Lewis, M., B. Alidaee, and G. Kochenberger, 2003: Modeling and solving the task allocation problem as an unconstrained quadratic binary program. Tech. Rep. HCES-09-03, Hearing Center for Enterprise Science.

Leyens, J. P., V. Yzerbyt, and G. Schadron, 1994: *Stereotypes and social cognition*. Sage Publications, Inc.

Lloyd, E. and W. Ledermann, (Eds.) , 1985: *Handbook of Applicable Mathematics*, Vol. 5. John Wiley & Sons.

Lo, V. M., 1988: Heuristic algorithms for task assignment in distributed systems. *IEEE Transactions on Computers*, **37 (11)**, 1384–1397.

Manisterski, E., E. David, S. Kraus, and N. R. Jennings, 2006: Forming efficient agent groups for completing complex tasks. *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, 834–841.

Mitchell, T., 1997: *Machine Learning*. McGraw-Hill.

Montada, L. and H. W. Bierhoff, (Eds.) , 1991: *Altruism in Social Systems*. Hogrefe & Huber Publishers.

Nash, J. F., 1950: Non-cooperative games. Ph.D. thesis, Princeton University.

Newman, M. E. J., 2003: The structure and function of complex networks. *SIAM Review*, **45 (2)**, 167–256.

Newman, M. E. J. and D. J. Watts, 1999a: Renormalization group analysis of the small-world network model. *Physics Letters A*, **263 (4-6)**, 341–346.

Newman, M. E. J. and D. J. Watts, 1999b: Scaling and percolation in the small-world network model. *Physical Review E*, **60 (6)**, 7332–7342.

Nguyen, Q. H., Y. S. Ong, and M. H. Lim, 2008: Non-genetic transmission of memes by diffusion. *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO 2008)*, ACM, 1017–1024.

Nisan, N., T. Roughgarden, E. Tardos, and V. V. Vazirani, (Eds.) , 2007: *Algorithmic game theory*. Cambridge University Press.

Nwana, H., 1996: Software agents: An overview. *The Knowledge Engineering Review*, **11 (03)**, 205–244.

Oakes, P. J., S. A. Haslam, and J. C. Turner, 1994: *Stereotypes and social reality*. Blackwell, Oxford.

Peitz, U., 2002: Struktur und entwicklung von beziehungen in unternehmensnetzwerken. Ph.D. thesis, European Business School Oestrich-Winkel.

Peleg, B. and P. Sudhölter, 2007: *Introduction to the theory of cooperative games*. Springer.

Pennington, D. C., 2002: *The Social Psychology of Behavior in Small Groups*. Psychology Press, New York.

Priesterjahn, S., 2008: Online imitation and adaptation in modern computer games. Ph.D. thesis, University of Paderborn.

Richert, W., 2009: Learning and imitation in heterogeneous robot groups. Ph.D. thesis, University of Paderborn.

Riolo, R. L., M. D. Cohan, and R. Axelrod, 2001: Evolution of cooperation without reciprocity. *Nature*, **414 (6862)**, 441–443.

Rocha, A. F., E. Massad, A. Pereira, and A. Pereira Jr, 2005: *The brain: fuzzy arithmetic to quantum computing*. Springer.

Rosenschein, J. S. and G. Zlotkin, 1994: *Rules of encounter: designing conventions for automated negotiation among computers*. The MIT Press.

Russell, S. J. and P. Norvig, 2010: *Artificial intelligence: a modern approach (Third Edition)*. Prentice Hall.

Sawyer, R. K., 2003: Artificial societies: Multiagent systems and the micro-macro link in sociological theory. *Sociological Methods & Research*, **31 (3)**, 325–363.

Schillo, M. and K. Fischer, 2003: Holonic multiagent systems. *KI Zeitschrift*, **4 (54)**, 327–332.

Schmidt, D. O., 1997: *Unternehmenskooperationen in Deutschland*. Deutscher Universitätsverlag.

Sen, S. and G. Weiss, 1999: Learning in multiagent systems. *Multiagent systems: a modern approach to distributed artificial intelligence*, G. Weiss, Ed., MIT Press, 259–298.

Seredynski, M., R. Kotowski, and W. Maka, 2010: Cooperative optimization in cellular automata-based multiagent systems with spatio-temporally generalized prisoner's dilemma model. *Proceedings of the 4th International Conference on Agent and Multi-Agent Systems, Technologies and Applications (AMSTA 2010)*, P. Jedrzejowicz, N. T. Nguyen, R. J. Howlett, and L. C. Jain, Eds., Springer, Lecture Notes in Artificial Intelligence, Vol. 6071, 120–129.

Shapley, L. S., 1953: A value for n-person games. *Contributions to the theory of games*, **2**, 307–317.

Shehory, O. and S. Kraus, 1995: Task allocation via coalition formation among autonomous agents. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 1995)*, 655–661.

Shehory, O. and S. Kraus, 1996: Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents. *Proceedings of the Second International Conference on Multiagent Systems (ICMAS 1996)*, 330–337.

Shehory, O. and S. Kraus, 1998: Methods for task allocation via agent coalition formation. *Artificial Intelligence*, **101 (1-2)**, 165–200.

Sherman, C., 2005: *Google power: Unleash the full potential of Google*. McGraw-Hill Osborne Media.

Shoham, Y. and K. Leyton-Brown, 2009: *Multiagent Systems—Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

Shoham, Y. and M. Tennenholtz, 1992a: Emergent conventions in multi-agent systems: initial experimental results and observations. *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR 1992)*, 225–231.

Shoham, Y. and M. Tennenholtz, 1992b: On the synthesis of useful social laws for artificial agent societies. *Prodeedings of the Tenth National Conference On Artificial Intelligence (AAAI 1992)*, John Wiley & Sons LTD, 276–276.

Stone, H. S., 1977: Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Transactions on Software Engineering*, **3 (1)**, 85–93.

Subiaul, F., J. F. Cantlon, R. L. Holloway, and H. S. Terrace, 2004: Cognitive imitation in rhesus macaques. *Science*, **305 (5682)**, 407–410.

Sydow, J., 1992: *Strategische Netzwerke: Evolution und Organisation*. Gabler, Habilitationsschrift Freie Universität Berlin.

Tajfel, H., M. G. Billig, R. P. Bundy, and C. Flament, 1971: Social categorization and intergroup behavior. *European Journal of Social Psychology*, **1 (2)**, 149–178.

Thagard, P., 2010: Cognitive science. *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., summer 2010 ed.

Trivers, R. L., 1971: The evolution of reciprocal altruism. *The Quarterly Review of Biology*, **46 (1)**, 35–57.

Tucker, A., 2007: *Applied Combinatorics (5th edition)*. John Wiley & Sons.

Upadhyaya, S. and S. Lata, 2008: Task allocation in distributed computing vs distributed database systems: A comparative study. *International Journal of Computer Science and Network Security*, **8 (3)**, 338–346.

Vickrey, W., 1961: Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, **16 (1)**, 8–37.

Vise, D. A. and M. Malseed, 2008: *The Google Story: For Google's 10th Birthday*. Delta Trade Paperbacks.

Vishnumurthy, V. and F. Paul, 2006: On heterogeneous overlay construction and random node selection in unstructured p2p networks. *Proceedings of the 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2006)*, IEEE, 1119–1130.

Vishnumurthy, V. and F. Paul, 2007: A comparison of structured and unstructured p2p approaches to heterogeneous random peer selection. *Proceedings of the USENIX 2007 Annual Technical Conference (USENIX 2007)*, USENIX, 309–322.

Vlassis, N., R. Elhorst, and J. R. Kok, 2004: Anytime algorithms for multiagent decision making using coordination graphs. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (IEEE SMC 2011)*, IEEE, Vol. 1, 953–957.

von Martial, F., 1990: Interactions among autonomous planning agents. *Decentralized AI—Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW 1989)*, 105–120.

Walker, A. and M. Wooldridge, 1995: Understanding the emergence of conventions in multi-agent systems. *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, 384–389.

Wallis, W. D., 2007: *A Beginner's Guide to Graph Theory (2nd Edition)*. Birkhäuser.

Wasserman, S. and K. Faust, 1994: *Social network analysis: Methods and applications*. Cambridge University Press.

Watts, D. J., 1999: Networks, dynamics, and the small-world phenomenon. *American Journal of Sociology*, **105 (2)**, 493–527.

Watts, D. J., 2003: *Small worlds: the dynamics of networks between order and randomness*. Princeton University Press.

Watts, D. J. and S. H. Strogatz, 1998: Collective dynamics of 'small-world' networks. *Nature*, **393 (6684)**, 440–442.

Weiss, G., 1996: Adaptation and learning in multi-agent systems: Some remarks and a bibliography. *Adaption and learning in multi-agent systems*, S. Sen and G. Weiss, Eds., Springer, No. 1042 in Lecture Notes in Artificial Intelligence, 1–21.

Weiss, G., (Ed.) , 1999: *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press, Cambridge, MA, USA.

Wispe, L., (Ed.) , 1978: *Altruism, sympathy, and helping*. Academic Press.

Wooldridge, M., 2009: *An Introduction to MultiAgent Systems - Second Edition*. John Wiley & Sons.

Wooldridge, M. and N. R. Jennings, 1994: Formalizing the cooperative problem solving process. *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence (IWDAI-94)*, 403–417.

Wooldridge, M. and N. R. Jennings, 1995: Intelligent agents: Theory and practice. *Knowledge Engineering Review*, **10 (2)**, 115–152.

Zimmermann, M. G., V. M. Eguiluz, and M. S. Miguel, 2001: *Cooperation, adaptation and the emergence of leadership*, 73–86. Springer.