Socially-Inspired Mechanisms for Restricting Exploitation in Artificial Agent Societies

Sharmila Savarimuthu

a thesis submitted for the degree of

Doctor of Philosophy

at the University of Otago, Dunedin,

New Zealand.

November 9, 2011

Abstract

Human societies have long cultivated the ability to organise themselves into groups and have also established formal or informal rules of behaviour that are expected within these groups. In the field of multi-agent systems, researchers are inspired by this ability of human societies to form groups and establish social control, and they have applied them to solve some of the problems in artificial agent societies.

One of the problems in artificial agent societies is the problem of non-cooperation, where individuals have motivations for not cooperating with other agents. An example of non-cooperation is the issue of freeriding, where some agents do not contribute to the welfare of the society but do consume valuable resources. This can be likened to the "commons" problem. The way to address this problem is by imposing strict rules by centralised institutions. However, centralised solutions suffer from performance bottlenecks, and their scalability is poor. Towards this end, our first objective of this thesis is to investigate decentralised mechanisms for facilitating social control in agent societies. Our second objective is associated with an important attribute of modern artificial societies, which is the openness of such societies. Agents may join and/or leave these societies at any time. Towards this end, our second objective of this thesis is to investigate mechanisms which can handle the dynamism of open agent societies.

Another key aspect in facilitating social control lies in employing appropriate mechanisms that can facilitate such control. In this thesis we are inspired by decentralised social practices found in human societies. This thesis investigates mechanisms that contribute towards the formation (via self-organisation) of different groups in an agent society based on their cooperativeness. It demonstrates that these mechanisms help in achieving the separation of good agents (cooperators) from bad agents (noncooperators) without expelling them from the society.

It demonstrates how the concepts of tags can be used for group formation and how the information about the cooperativeness of agents in the society can be spread based on using socially-inspired mechanisms. It also investigates how monitoring and control mechanisms such as referrals, voting, gossip, resource restriction, and ostracism can be used in artificial agent societies. Thus our focus of this thesis is to develop socially-inspired mechanisms to facilitate self-organisation of groups in agent societies to restrict exploitation. We demonstrate that the formation of groups shield "bad" agents from taking advantage of "good" agents. We also demonstrate that the society is better off if the groups are organised based on their cooperativeness.

Overall, the goal of this thesis is to investigate and demonstrate the new socially-inspired mechanisms for the self-organisation of groups in open, decentralised agent societies. This thesis initially systematically explores closed, centralised societies and gradually moves on to open, decentralised societies, since many real-life societies lie somewhere between these two ends of the open spectrum, with more and more societies lying closer to the end of full openness. We believe the mechanisms explored in this thesis can be applied to open, decentralised agent societies, such as electronic file-sharing societies to help avoid the problem of freeriding. The mechanisms proposed in this thesis could also be applied to organise agents into groups based on their behaviour, in virtual worlds and other online communities.

Acknowledgements

I would like to thank my supervisors Maryam Purvis, Martin Purvis, and Tony Savarimuthu for their guidance and support. This research has benefited from interactions with the members of NZDIS (New Zealand Distributed Information Systems) research group at Otago. I would like to thank each one of them, especially Mariusz Nowostawski for his advice in writing this thesis and Stephen Cranefield for his comments and insights. I thank Michael Winikoff, Brendon Woodford and Jeremiah Deng for their comments in our NZDIS meetings. Many people have indirectly been involved in this work by the process of peer reviewing and they all deserve thanks.

I thank my friends for their support, especially to those who proof-read papers and chapters of this thesis and offered helpful comments and insights. I thank Antoni Moore for reviewing my thesis chapters. I thank Paul Crane for proof-reading my thesis. I thank my officemate Surangika Ranathunga for being there through my ups and downs. I also thank all my officemates (from the past and present). The list is a bit long to name each of them. Here they are: Silvie Fiat, Marcos de Oliveira, Vincent Goh, Peter Li, Julian Muenster, Yuwei Xu, and Tomek Kolasa. I thank the friendly next door fellow postgrads, Rasika Withanawasam, Vivien Yong, Christopher Frantz, Angrosh Mandya, and Femi Aderohunmu. I would also thank my friend Carl Leichter who is sailing the same boat as me (closer to thesis submission). I thank Prajesh Chhanabhai for his support and friendship especially during my initial days at Otago. I thank the Information Science Postgrad Group (ISPG). All you people made my days at Otago memorable.

Tony Savarimuthu deserves a special mention and many thanks for the interest and enthusiasm he had shown in my research over these years. I thank my con-

venor Peter Whigham for his wonderful convening. Thanks to TSG (Technical Support Group) who has provided computer support.

I would like to thank my sponsors for financial support. I thank University of Otago for the PhD scholarship and BuildIT for the student travel award and NZFGW (New Zealand Federation of Graduate Women) for the Otago travel award.

Thanks to my family for their love, care, support, and encouragement. Especially thanks to my beloved parents for their unconditional love. Mum and Dad, thanks for all the things that you have done for me and I will never be able to give back the unsurpassable goodness you have shown to me.

Above all I thank God Almighty who blessed me with his love by surrounding me with lovely people and moreover for making me realise it and mainly for making things possible which I would not have wished or dreamed for myself otherwise.

To my family

Contents

I Introduction and Background **1**

1 Introduction **2**
1.1 Motivation . . . 3
1.2 Research questions and contributions . . . 4
1.2.1 Publications . . . 5
1.3 Thesis structure . . . 5

2 Social control in multi-agent systems **8**
2.1 Agents and multi-agent systems . . . 9
2.1.1 Software agents . . . 9
2.1.2 Multi-agent systems (MAS) . . . 9
2.2 Artificial societies . . . 11
2.2.1 Agent Based Social Simulation (ABSS) . . . 12
2.2.2 Nature of societies . . . 12
2.2.3 Level of control in agent societies . . . 14
2.3 Non-cooperation or selfish behaviour in societies . . . 16
2.3.1 Non-cooperation problem in human societies . . . 16
2.3.1.1 Tragedy of the commons . . . 16
2.3.1.2 Modeling the tragedy of the commons using Prisoner's Dilemma . . . 17
2.3.2 Non-cooperation problem in electronic societies . . . 18

3 Facilitating social control in artificial agent societies **22**
3.1 Why artificial agent societies? . . . 23
3.2 Processes used in managing agent societies . . . 26
3.2.1 Reputation-based mechanisms for social control . . . 27
3.3 Separation of groups within an agent society . . . 28
3.3.1 Group segregation . . . 29
3.4 Research works using different social mechanisms . . . 30
3.4.1 Gossip . . . 30
3.4.2 Ostracism . . . 33
3.4.3 Referral and voting . . . 34
3.4.4 Tagging . . . 35
3.5 Altruism . . . 38
3.6 Contextualising the contribution of this thesis . . . 39

3.7 Summary . . . 42

II Socially-Inspired Models 43

4 Knowledge sharing in agent societies 44
4.1 Introduction . . . 44
4.2 Outline of basic experimental mechanisms . . . 46
4.3 Experimental setup . . . 47
4.3.1 Tagless model . . . 48
4.3.2 Results for the Tagless model experiment . . . 49
4.3.3 Tag model . . . 51
4.3.4 Result for the Tag model experiment . . . 53
4.3.5 Tag-and-trait model . . . 54
4.3.6 Results for the Tag-and-trait model experiment . . . 56
4.3.7 Conversion model . . . 61
4.3.8 Result for the Conversion model experiment . . . 63
4.3.9 Experiment on individual cost bearing versus group cost sharing . . . 65
4.3.10 Result for the individual versus group cost sharing experiment . . . 66
4.4 Discussion . . . 67
4.5 Summary . . . 68

5 Self-organisation in semi-centralised systems 70
5.1 Introduction . . . 70
5.2 System design and experiments . . . 71
5.2.1 System 1 using tags . . . 72
5.2.2 System 2 without using tags . . . 78
5.2.3 System 3 using referral combined with System 1 . . . 80
5.2.4 System 4 using resource restriction combined with System 1 . . . 83
5.2.4.1 Experiment without resource restriction . . . 83
5.2.4.2 Experiment with resource restriction . . . 83
5.3 Summary . . . 85

6 Self-organisation in decentralised closed systems 87
6.1 Introduction . . . 87
6.2 Proposed mechanisms for self-organisation of groups . . . 88
6.3 Agent attributes in our experimental setup . . . 89
6.4 Rank-based grouping mechanism . . . 92
6.4.1 Gossip interaction . . . 92
6.4.2 Leaving a group . . . 93
6.4.3 Choosing a group to join . . . 93
6.4.4 Entry criteria . . . 96
6.4.5 Results of experiments . . . 97
6.5 Dynamic grouping mechanism . . . 100
6.5.1 Gossip interaction . . . 101
6.5.2 Leaving a group . . . 102
6.5.3 Choosing a group to join . . . 102

6.5.4 Entry criteria . . . 103
6.5.5 Results of experiments . . . 106
6.6 Random hopping mechanism . . . 107
6.7 Individual group history mechanism . . . 108
6.7.1 Memory of last group only . . . 108
6.7.2 Memory of all previous groups . . . 109
6.8 Sharing group history mechanism . . . 110
6.8.1 Gossip interaction . . . 110
6.8.2 Leaving a group . . . 111
6.8.3 Choosing a group to join . . . 111
6.8.4 Entry criteria . . . 115
6.8.5 Results of experiments . . . 115
6.9 Summary . . . 119

7 Self-organisation in decentralised open systems 120
7.1 Introduction . . . 120
7.2 Open system . . . 121
7.2.1 Results of experiments . . . 122
7.3 Enhanced open system . . . 124
7.3.1 Experiment 1 - Self-organisation in an open society . . . 125
7.3.2 Experiment 2 - Arrival rate greater than departure rate . . . 128
7.3.3 Experiment 3 - Arrival rate equal to departure rate . . . 130
7.3.4 Experiment 4 - Varying lifespans of agents . . . 132
7.4 Differences . . . 133
7.5 Summary . . . 134

III Discussion and Conclusion 136

8 Discussion 137
8.1 Contributions of this thesis . . . 137
8.1.1 Highlight overview of the chapters . . . 138
8.2 Limitations and Future Work . . . 143

9 Conclusion 145

References 147

A Software Specifications 163

B Source Code 164
B.1 Game.java . . . 164
B.2 Player.java . . . 167
B.3 GameConstants.java . . . 169

List of Tables

1.1 Publications backing the thesis 6
1.2 Three parts of the thesis 7

2.1 Payoff matrix for Prisoner's Dilemma. 17

4.1 The main differences in the four sharing systems. 68

5.1 Payoff values for Prisoner's Dilemma. 72
5.2 Different groups and their performance. 82

6.1 Developed mechanisms for self-organised group composition to reduce exploitation. 91
6.2 Experimental parameters for rank-based grouping mechanism. 92
6.3 Group order and entry value. 97
6.4 Initial and final cooperation value of groups. 99
6.5 Previous group history. 112
6.6 Latest available information. 112

7.1 Experimental parameters for open society. 124

List of Figures

2.1 Nature of societies 12
2.2 Control applied in the society 15
2.3 Progression of the experimental chapters. 21

3.1 Contextualising the experimental chapters. 41

4.1 The knowledge and sharing level in Tagless model. The knowledge of the population represented by K line and the sharing behaviour by the S line. . 51
4.2 Comparison of knowledge in Tag model and Tagless model. 54
4.3 The K line shows the knowledge and the S line shows the sharing (All-sharing). 56
4.4 The K line shows the knowledge and the S line shows the sharing (No-sharing). 57
4.5 Groups satisfying the winning condition. 58
4.6 Size of tag groups (with 3 bit tags). 59
4.7 Keeping S constant and increasing K. 60
4.8 Keeping K constant and increasing S. 60
4.9 The K line shows the knowledge and the S line shows the sharing. 63
4.10 Four types of agents at the end of the iterations. 64
4.11 Tag groups from 2 sets. 66

5.1 Elimination from group. 75
5.2 Selection from pool. 76
5.3 Prisoner's Dilemma with tagging. 77
5.4 Cooperativeness and score of groups. 77
5.5 Prisoner's Dilemma without tagging. 79
5.6 Comparison of results in boxplot. 85

6.1 Self-organisation of groups using rank-based grouping mechanism. 99
6.2 Formula for entry value. 105
6.3 Self-organisation of groups using dynamic grouping mechanism. 106
6.4 No self-organisation of groups based on cooperativeness using random hopping mechanism. 107
6.5 Group history mechanism with memory of last group only. 108
6.6 Group history mechanism with memory of all previous groups. 109
6.7 Self-organisation of groups using sharing group history mechanism. 116
6.8 Composition of groups with agents at the start. 117
6.9 Composition of groups with agents at the end. 118

7.1 Self-organisation of groups based on cooperativeness in open society. . . . 123
7.2 Self-organisation of an open system when agents' arrival and departure rates are dynamic. . . . 126
7.3 Snapshot of the groups at different iterations. . . . 127
7.4 Self-organisation of an open system when agents' arrival rate is increased. . 129
7.5 Self-organisation of an open system when the arrival rate is equal to the departure rate of the agents. . . . 131
7.6 Group formation with minimum TTL = 300. . . . 132
7.7 Group formation with minimum TTL = 500. . . . 133

8.1 Progression of the experimental chapters. . . . 139
8.2 Processes employed in this thesis. . . . 140
8.3 Mechanisms developed in this thesis. . . . 141

List of Algorithms

4.1 Pseudocode for the Tagless model 50
4.2 Pseudocode for the Tag model 52
4.3 Pseudocode for the Tag-and-trait model 55
4.4 Pseudocode for the Conversion model 62
4.5 Pseudocode for the process of cost sharing 65
5.1 Pseudocode for System 1 (using tags) 74
5.2 Pseudocode for System 2 (without using tags) 79
5.3 Pseudocode for System 3 (using referral combined with System 1) 81
5.4 Pseudocode for System 4 (using resource restriction combined with System 1) 84
6.1 Pseudocode for gossip. 93
6.2 Pseudocode of the process of an agent joining another group. 95
6.3 Pseudocode of the filtering process. 97
6.4 Pseudocode for rank-based grouping mechanism. 98
6.5 Pseudocode to avoid the worst player. 101
6.6 Pseudocode for using gossip information to hop between groups. 104
6.7 Pseudocode for sharing group history mechanism. 114

Part I - Introduction and Background

Chapter 1

Introduction

Human societies that we are a part of have evolved over millennia. One of the key components that make human societies a success is the innate ability of their members to organise into groups to cooperate and collaborate with each other and perform tasks in an organised manner. To facilitate cooperation and collaboration, these groups have established formal or informal social structures that specify the rules of interactions between individuals in the groups. Formal structures include the establishment of organisations and governments that operate based on well-formed rules. Informal structures include the use of socially established mechanisms such as reputation, trust and norms.

Either formal or informal, these structures (or mechanisms) facilitate the betterment of the society. These structures facilitate the separation of good individuals from bad. It is from this ability of human societies that we derive the inspiration for this thesis.

Our principal objective in this thesis is to develop several social mechanisms for artificial agent societies that can help to distinguish "good" agents (cooperative) from "bad" ones (uncooperative) in an agent society. This separation of individuals into different groups not only shields from bad agents exploiting the good but leads to the betterment of the society. For example, an electronic file-sharing society containing sharers and non-sharers will benefit from the segregation process that sifts good from bad. Grouping good with good and bad with bad makes a better society, where better services can be made available for good groups. We have developed and experimented with mechanisms that operate under both closed- and open-world assumptions which can be applied to centralised, semi-centralised, or distributed

societies.

1.1 Motivation

With the advancement of Information and Communications Technology (ICT), electronic societies have become popular, and different types of electronic societies exist. Moreover, with the development of Web 2.0 (O'Reilly, 2005), the Internet is being used in a more interactive manner (i.e. a "read-write web" that facilitates better interaction between individuals). Modern electronic societies facilitated by Web 2.0 technologies include social networking societies and communal blogs and wikis. Some of the well-known electronic societies include Massively Multi-player Online Games (MMOGs) such as World of Warcraft (1994), e-commerce societies such as eBay (1995), and virtual societies such as Second Life (2003).

Another important domain of electronic societies is the area of Peer-to-Peer systems (P2P). An important feature or activity that is involved in the P2P domain is file-sharing, which consumes a considerable amount of the traffic on the Internet (Jackson, 2011). People share their digital goods, such as videos, audios, images and ebooks, with others and also receive these goods from them. Thus the sharing of digital goods has led to the development of digital communities whose goal is to facilitate sharing among its members. However, the goal of sharing in digital communities can be undermined by the presence of freeriders. The freeriding problem was first reported in Gnutella by Adar and Huberman (2000). In P2P common usage terms, a freerider is called as a "leacher". Leaching is not a crime, but it is selfish behaviour which results in receiving the benefit from the community without providing reasonable contribution back to the community.

Due to "nature's design", different types of people exist in a society. In online communities the 'types' of users may correspond to different 'levels' of cooperativeness. The achievement of an online community can be determined by the cooperativeness of its members. Agents' cooperativeness differs from one another. This behaviour (cooperativeness) is not the same for all the agents. The level of cooperativeness varies among agents. So, one of the potential actions to take to address the problem of good agents being exploited by bad agents is to separate good agents from bad without expelling the bad from the society. This can be accomplished by separating agents based on their cooperativeness. By doing this we

can avoid cooperators being exploited by freeriding. In order to implement this solution, three features should be considered. They are

1. the nature of the society (closed or open)
2. the level of control in the society (centralised, semi-centralised, or decentralised)
3. the social mechanisms used for separating groups

In this connection we observe that some natural complex systems show cooperative behaviour. For example in animal societies, animals form herds, schools, clans. Additionally, humans have organised themselves into groups to perform various tasks. Individuals in a group cooperate, coordinate and collaborate with each other. Inspired by the organisation of groups in these societies, this thesis investigates how social mechanisms can be developed and used in agent societies to help agents to self-organise themselves into groups (i.e. help in the segregation of groups based on the cooperativeness of agents) which leads to the betterment of the society.

Social mechanisms developed in this thesis are used as components to help build up and maintain social structure in artificial societies. Knowledge sharing and P2P file-sharing are the application domains that have been considered for testing the social mechanisms we are investigating in this thesis. Nevertheless, most of the results and findings presented are general and could also be applied to other systems such as online gaming societies and virtual worlds.

1.2 Research questions and contributions

The overarching research question posed in this thesis is *How can we separate good individuals from bad by dynamically forming different groups based on cooperativeness? (or help the self-organisation of groups based on cooperativeness?)*. The sub-questions are as follows.

1. How can we design and develop systems that have the ability to deal with selfish, non-cooperative agents without excluding them?

2. How can we develop mechanisms that can be used to self-organise societies (i.e. separate them into groups based on cooperativeness?) that are centralised, semi-centralised, or decentralised operating under closed- or open-world assumptions?

3. How can social mechanisms such as referral, voting, tagging, gossip and ostracism be used for this purpose?

The above questions are interrelated to each other. The main focus of the thesis is to provide a decentralised solution for open P2P systems from models inspired by human societies to achieve self-organisation. Our aim is to help establish artificial societies which are open, scalable, adaptive, self-organising, and, decentralised.

1.2.1 Publications

The research presented in this thesis has resulted in several peer-reviewed publications. Table 1.1 shows which publication corresponds to which chapter in the thesis.

1.3 Thesis structure

This thesis has three parts as shown in Table 1.2. The first part of the thesis (Chapters 1-3) provides an introduction and relevant background for the thesis. Chapter 1 presents an introduction to the thesis, its motivation, its research objectives, and its structure. Chapter 2 introduces the relevant topics and concepts to understand the main research of this work. This includes an introduction to multi-agent systems and the characteristics of artificial agent societies. It also discusses different types of system structures, such as closed and open, and levels of control applied on them, such as centralised, semi-centralised and decentralised. It also introduces the freeriding problem. Chapter 3 discusses the social theories employed in this work by introducing findings from the relevant literature. The social mechanisms discussed are tagging, referral, voting, gossip, and ostracism.

The second part, which comprises of Chapters 4-7, presents the newly developed socially-inspired mechanisms along with the experiments, results, and our findings. All our experiments make use of some social mechanisms that we have developed. Chapter 4 presents

No	Paper	Chapter
1	Sharmila Savarimuthu, Maryam Purvis, and Martin K. Purvis. 2008. ***Emergence of Sharing Behaviour in a Multi-agent Society Using Tags***. In Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 02 . IEEE Computer Society, Washington, DC, USA, 527-530.	4
2	Sharmila Savarimuthu, Maryam Purvis, and Martin K. Purvis. 2008. ***Altruistic Sharing using Tags***. The 6th International Workshop on Agents and peer-to-peer computing. Estoril, Portugal, 2008.	4
3	Sharmila Savarimuthu, Maryam Purvis, and Martin K. Purvis. 2009. ***Tag-based model for knowledge sharing in agent society***. In Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1299-1300.	4
4	Sharmila Savarimuthu, Martin K. Purvis, Maryam Purvis, and Mariusz Nowostawski. 2009. ***Mechanisms to restrict exploitation and improve societal performance in multi-agent systems***. Intelligence Integration in Distributed Knowledge Management, IGI Global, Hershey, New York, 182-194.	5
5	Martin K. Purvis, Sharmila Savarimuthu, Marcos De Oliveira, and Maryam Purvis. 2006. ***Mechanisms for Cooperative Behaviour in Agent Institutions***. In Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology. IEEE Computer Society, Washington, DC, USA, 121-124.	5
6	Sharmila Savarimuthu, Maryam Purvis, and Martin K. Purvis. 2009. ***Self-Organization of Peers in Agent Societies***. In Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 02. IEEE Computer Society, Washington, DC, USA, 74-77.	6
7	Sharmila Savarimuthu, Maryam Purvis, Martin K. Purvis, and Bastin Tony Roy Savarimuthu. 2010. ***Mechanisms for the self-organization of peer groups in agent societies***. Multi-Agent-Based Simulation XI - International Workshop - volume 6532 of Lecture Notes in Artificial Intelligence, Springer, Toronto, Canada, May 11, 2010, 93-107.	6 and 7
8	Sharmila Savarimuthu, Martin K. Purvis, Bastin Tony Roy Savarimuthu, and Maryam Purvis. 2010. ***Gossip-based Self-organising Open Agent Societies***. The 13th International Conference on Principles and Practice of Multi-Agent Systems, Kolkata, India, November 12th -15th.	7

Table 1.1: Publications backing the thesis

Parts	Chapters
I - Introduction and Background	1-3
II - Socially-Inspired Models	4-7
III - Discussion and Conclusion	8-9

Table 1.2: Three parts of the thesis

the mechanisms developed for knowledge sharing in artificial agent societies in a system-controlled (*centralised*) environment and its results. Chapter 5 presents the mechanisms developed that help to resist exploitation and lead to betterment of the society in *semi-centralised* agent societies by playing the Prisoner's dilemma game. Chapter 6 presents the mechanisms developed for sharing digital goods (P2P file-sharing) in *closed, decentralised agent societies* and the results. Chapter 7 presents the mechanisms developed for sharing digital goods in *open, decentralised agent societies* and the results.

The third part presents the discussion and conclusion and consists of Chapters 8 and 9. Chapter 8 discusses the contributions, limitations and future work. Chapter 9 concludes the thesis.

Chapter 2

Social control in multi-agent systems

Computer scientists have been inspired by several natural phenomena to design computer systems. Inspired by natural evolution, they have been exploring how evolutionary concepts such as mutation and cross-over can be used to generate new solutions in the areas of genetic algorithms and evolutionary computing (Holland, 1992; Eiben and Smith, 2008). In the field of neural computing (Wasserman, 1989), researchers have been inspired by the neural processes that happen in the human brain and have modeled them as software processes to find solutions to certain types of problems in the areas of image recognition and speech synthesis. In a similar vein, researchers in multi-agent systems have sought inspiration from human societies. Human societies are made up of several autonomous individuals who communicate, collaborate, compete and cooperate with each other. They operate in groups that are cohesive, and the order in these groups is sustained through informal social mechanisms such as trust, reputation and norms (Dellarocas and Klein, 1999). Inspired by processes in human societies, this thesis explores the social mechanisms that can be employed in artificial agent societies to facilitate social order through the separation of groups. Towards this goal, this chapter aims at discussing the background of this thesis. First, it provides an introduction to multi-agent systems (MAS) and how societies are modeled and simulated in the field of MAS in Section 2.1. Second, it provides an introduction to artificial societies, agent-based social simulation (ABSS), the nature of the agent societies and level of control applied on these societies. Third, it introduces the problem of non-cooperation in human and electronic societies in Section 2.3.

2.1 Agents and multi-agent systems

This subsection aims at providing an introduction to software agents and the field of Multi-agent Systems (MAS).

2.1.1 Software agents

An agent that is a part of a MAS is an autonomous software entity which can perform tasks in order to achieve its goal(s) without direct human supervision. A software agent may use Artificial Intelligence (AI) which is an adaptation of human-like intelligence for computational entities. Agents interact with each other to achieve their goals. But still they are *autonomous* entities with the ability to say *no* to requests to perform an action. Software agents have three other main capabilities to meet their delegated objectives. They are

a) *Reactivity*: agents sense the changes in the environment and react accordingly.

b) *Proactiveness*: agents exhibit goal-directed behaviour by carrying out appropriate tasks by initiating those tasks themselves.

c) *Social ability*: agents interact with each other to work together as a team to achieve a shared goal when the goal is not achievable by an individual agent. Hence cooperation is needed in many circumstances to achieve the goal.

According to Wooldridge (2009) an agent is "a computer system that is capable of independent (autonomous) action on behalf of its user or owner (figuring out what needs to be done to satisfy delegated objectives, rather than constantly being told)."

Characteristics of software agents also include learning, negotiating, being adaptive, and being cooperative or competitive (Wooldridge and Jennings, 1995). A software agent can come in many different types such as: interface agent, data mining agent, gaming agent, user agent, buyer agent, monitor agent and mobile agent (Nwana, 1996).

2.1.2 Multi-agent systems (MAS)

A multi-agent system is a system that consists of many autonomous agents interacting with each other to achieve a goal. The research field of multi-agent systems is about three decades old. It originated as a research branch of Distributed Artificial Intelligence (DAI) (Weiss,

1999), which started in the 1980s as a series of distributed artificial intelligence workshops (Davis, 1980, 1982). The focus of DAI was to investigate interaction mechanisms for computing nodes that are distributed across a network. Over the years, researchers realised the need for modeling societies (i.e. a collection or group of individuals as opposed to single agents), since problems of cooperation and coordination can be better addressed through the design of mechanisms for the society as a whole as opposed to just the individual agents. Additionally, earlier linguistic modeling of human communication abilities through the modeling of speech-acts provided the infrastructured modeling paradigm for software agents to exchange messages, negotiate with one other, influence each other, and also resolve conflicts. Thus, MAS also became a computational tool for sociologists to test their theories about human societies (Doran and Palmer, 1995; Helbing, Farkas, and Vicsek, 2000; Macy and Willer, 2002; Pan, Han, Dauber, and Law, 2007).

According to Bradshaw (1997), multi-agent systems are suitable for developing large scale distributed applications to help solving complex problems. Typical examples that employ distributed solutions can be found in domains that require resource allocation and load balancing. MAS are suitable for applications such as air traffic control, robots working together, vehicle monitoring, supply chain operations, and network management (Nguyen-Duc, Briot, Drogoul, and Duong, 2003; hadouaj and Drogoul, 2004; Arkin and Hobbs, 1993; Cakar and Muller-Schloer, 2010; Durfee, 2006; Steeb, McArthur, Cammarata, and Narain, 1986; Lesser and Corkill, 1983; Swaminathan, Smith, and Sadeh, 1997; Chan and Chan, 2010; Smith and Davis, 1981). A real life example for distributed problem solving would be crowd sourcing (Brabham, 2008). A problem is submitted to the online users/groups. They work to find the solution by managing the workload within themselves. Different groups come up with different solutions. Among these solutions the best one will be chosen. Alternatively, a solution can be found by dividing the main problem into several parts and assigning each of them to different groups. At the end, combining the solutions from different groups would produce the final solution for the problem. The main advantage of distributed problem solving is the speed (Smith and Davis, 1981). The solution can be reached faster by making more entities involved in solving the problem and balancing the work load.

According to Wooldridge (2009), a multi-agent system "is one that consists of a number of agents, which interact with one-another. In the most general case, agents will be acting

on behalf of users with different goals and motivations. To successfully interact, they will require the ability to cooperate, coordinate, and negotiate with each other, much as people do."

The next subsection is organised as follows. An introduction to the field of Agent Based Social Simulation (ABSS) is given in Section 2.2. It also discusses two important dimensions that are considered when modeling agent societies. These two dimensions are the nature of the society (e.g. open and closed) and the level of distributed management control on the part of the agents in the society (e.g. centralised and decentralised). These two dimensions are discussed in Sections 2.2.2 and 2.2.3 respectively.

2.2 Artificial societies

An agent society is made up of several agents that interact. These agents may play different roles, and their interactions may be governed by protocols and norms (Artikis and Pitt, 2001). These societies, are often called artificial societies (Gilbert and Conte, 1995). Sociologists are interested in investigating how human societies that are formed with these capabilities perform under certain conditions (Conte, 2001; Gilbert and Troitzsch, 1999; Suleiman, Troitzsch, and Gilbert, 2003). Computer scientists use artificial societies to model and study how electronic societies that are made up of agents and also human users operate under certain conditions (e.g. auctions (Sierra, 2004; Noriega, 1999) and P2P systems (Pouwelse, Garbacki, Epema, and Sips, 2005; Babaoglu, Meling, and Montresor, 2002)). Artificial societies are thus tools not only for understanding human societies but also for investigating mechanisms that can be used in electronic societies. Multi-agent systems also provide sophisticated tools for simulating societies (Davidsson, 2002; Ferber, 1999), which may aid understanding various kinds of social processes in these two types of societies.

Artificial societies are agent-based systems which are also suited for the study of emergent properties of societies of agents (Ferber, 1999; Di Marzo Serugendo, Gleizes, and Karageorgos, 2005; Bak, 1996). Interactions between agents in these artificial societies may lead to new and emergent collective 'macro' behaviour that may result from these 'micro' interactions.

2.2.1 Agent Based Social Simulation (ABSS)

Multi-agent-based social simulations are used for modeling the real world by creating artificial agent societies with many interacting agents. Individuals in human societies are modeled as software agents. As a consequence, intuitions and theories can be tested on this artificial society, in the so called "Artificial Social Laboratory".

Simulation modeling is a tool for understanding, testing hypotheses, and predicting results. This is particularly valuable because not all real world environments provide access to their internal processes and states. Using simulations, their states (e.g. variables) can be controlled and their effects on results can be studied through "what-if" scenarios. Accordingly, an Agent Based Social Simulation (ABSS) is a social modeling approach that employs an agent society to conduct microsimulations which may lead to macro behaviour (emergent behaviour such as self-organisation) through the interactions of the agents. Thus, ABSS enable scientists to study the complex social behaviour.

2.2.2 Nature of societies

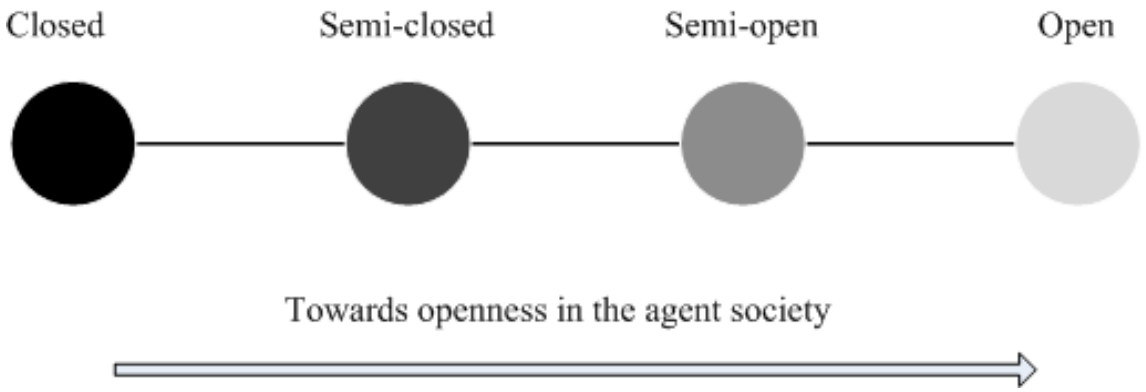

Figure 2.1: Nature of societies

Researchers interested in ABSS have modeled different types of agent societies. The societies can be open, semi-open, semi-closed or closed as depicted in Figure 2.1. These categories are based on the classification by Davidsson (2001) and are discussed below.

- Closed society

A closed society is limited in terms of the number of participants. In a closed society no external agents are allowed entry or participation. Because of this reason, the maintenance of the society is relatively easy (i.e. less effort is spent in maintaining social control). Thus, this type of society supports stability and trustfulness (i.e. cheaters are easily identified[1], hence participants usually tend to be more trustworthy).

- Semi-closed society

  In a semi-closed society an external party can request for an agent to be created which will work on behalf of the user (e.g. booking flight tickets). Here the new agent is of a 'predefined' type. Hence this society avoids the problem with malicious agents, since it has created the predefined agent types or roles. In our view a semi-closed society could also be called an 'open society constrained by roles'.

- Semi-open society

  In a semi-open society an electronic institution operates as a gate keeper. The joining agent promises to follow the constraints specific to the society. This institution may decide whether to accept the agent to the society based on its reputation. (e.g. Second life (Rymaszewski, Au, Wallace, Winters, Ondrejka, Batstone-Cunningham, and Rosedale, 2006)). Once the agent has been admitted, the institution can check whether the new agent follows the rules by monitoring its interactions from time to time, since it could be a malicious agent. In our view a semi-open society could also be called an 'open society based on reputation'.

- Open society

  In an open society any one can join and leave at any time. The best example for this type of society is the World Wide Web (WWW), since it is open and has no restriction on who joins. These type of societies support openness (i.e. any one can join) and flexibility. Since there are no restrictions for joining the society, there are concerns about trustfulness (i.e. bad agents can cause damage). Hence the stability of the society cannot be guaranteed.

[1] Since there is limited number of participants in the society.

The main difference between semi-open and semi-closed societies is that in semi-open societies, it may be difficult to monitor the new agents (due to the need for more resources to monitor interactions). These new agents may execute malicious code. So the stability and trustfulness properties are at stake, while in the semi-closed society there is less of a need for monitoring, since the agents are less likely to be malicious. However, the semi-closed society suffers from the lack of flexibility in the society (i.e. if a new agent is not of a predefined type, it cannot join the society).

Though there is a distinction between semi-open and semi-closed societies, this thesis will make use of just two broadly defined types of societies: closed agent societies and open agent societies.

2.2.3 Level of control in agent societies

An important aspect that has to be considered when investigating the behaviour of agent societies is the level of control of agents in the society. For example an agent can be monitored and controlled heavily by an institution all the time, while an agent in another institution may not be monitored at all.

There are different levels of control which can be applied in the society. The society can be centralised, semi-centralised, or decentralised. Figure 2.2 shows different levels of control in a society. The level of control decreases when we move from left to right (centralised to decentralised).

- Centralised

  Centralised systems are controlled by a central authority. These systems operate in a top-down fashion. Having centralised control is a convenient solution, since it is straightforward to implement a direct control. However, this solution is not scalable and is not suitable for dynamically growing systems which change and expand their state spaces (Minar, 2002; Bias, 2009; Bondi, 2000). There are several disadvantages of central control including single-point-of-failure and performance bottlenecks. Most significantly, centralised systems have scalability issues.

- Semi-centralised

Semi-centralised systems do not have one central authority; instead they have several local controllers (e.g. local controllers in groups). Modern electronic societies tend to be semi-centralised (partially centralised / hybrid systems) or decentralised. Semi-centralised systems also suffer partly from the weakness of the centralised systems.

- Decentralised

  Decentralised systems are distributed, and they have no central authority. These systems operate in a bottom-up fashion. Bottom-up approaches are suitable for developing dynamic systems and are responsive to changing conditions, and are relatively scalable.

It should be noted that these societies operate either under closed- or open-world assumptions. For the above mentioned reasons and also to facilitate scalable and robust societies, it is better to have decentralised control for open and distributed information systems.

This thesis models all three types of controls in societies. The overall goal is to develop techniques that facilitate a progressive move from a centralised to decentralised system control for open agent societies.

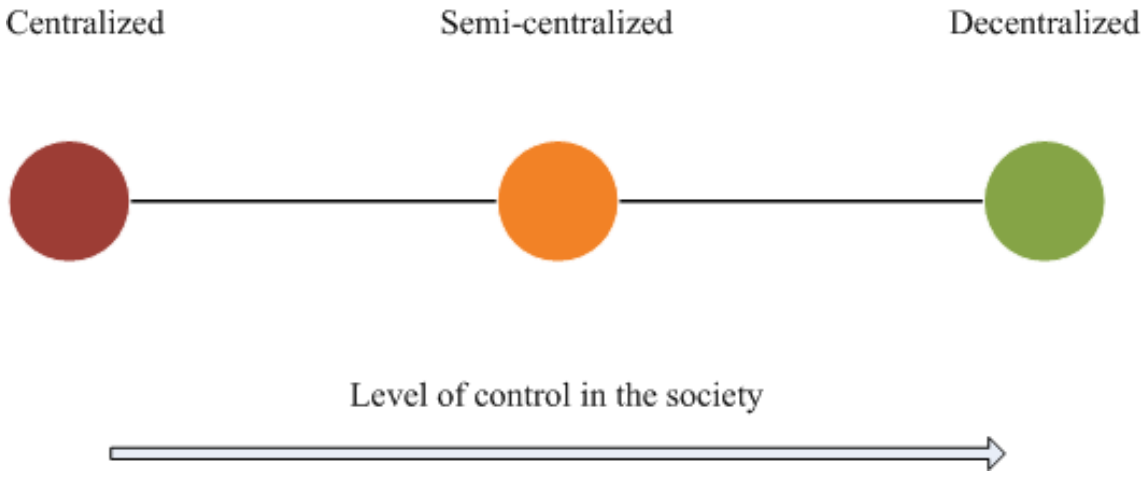

Figure 2.2: Control applied in the society

2.3 Non-cooperation or selfish behaviour in societies

This subsection introduces the problem and organisational metaphor that has motivated this thesis. It discusses the problem of non-cooperation in human and artificial agent societies. Sociologists have been investigating the problem of cooperation for several centuries (Dawes, 1980; Axelrod, 1984; Kollock, 1998). In particular, they have been working towards finding ways to facilitate and encourage social order through informal social mechanisms, such as norms, and more formalised mechanisms, such as rules that are enforced through policing mechanisms. In their well-known paper on civil agent societies, Dellarocas and Klein (1999) argue that civil agent societies provide adequate social mechanisms that can be used for solving problems in artificial agent societies. These mechanisms include interaction protocols and norms.

This subsection aims at discussing the problem of non-cooperation in *human* societies and electronic societies. Section 2.3.1 also discusses research solutions relevant to this problem.

2.3.1 Non-cooperation problem in human societies

For a society to operate effectively, agents within the society must obey certain social rules and norms. However, autonomous agents might not obey these rules or norms. When everyone else cooperates a rogue agent may benefit from non-cooperation. This situation is exemplified in Hardin's famous example of the "Tragedy of the Commons" (Hardin, 1968).

2.3.1.1 Tragedy of the commons

In Hardin's classic paper (1968) "Tragedy of the Commons", he outlines a scenario that depicts the tragedy of the herders in pastures. A common pasture is open to herders, each of whom tries to maintain as many cattle as possible on the commons. A herder reckons that the positive benefits of adding one additional animal will all go to him, alone, whereas the negative effects from overgrazing of that one additional animal will be shared and borne by all the herders. Accordingly, self-interested herders may continue adding one more animal to their herds, even if they know that collectively this is destroying the commons. The

question posed by this scenario is: how can we restrict selfish herders and avoid the tragedy of resource depletion?

Tragedy of the commons is a social dilemma which depicts the public goods problem of the society. The Prisoner's dilemma game models the dilemma between two individuals.

2.3.1.2 Modeling the tragedy of the commons using Prisoner's Dilemma

The Tragedy of the Commons can be modeled and studied using the "Prisoner's Dilemma" game (Axelrod, 1984). The Prisoner's Dilemma was proposed by two mathematicians Merrill Flood and Melvin Dresher in 1950 and was formalised by Albert Tucker (Tucker, 1950). The game depicts a situation where two collaborating criminals are imprisoned and questioned separately. Each criminal may either cooperate with his fellow criminal by refusing to divulge details of the crime or defect by "ratting" on his colleague. It is possible to establish a reward structure such that

- if both criminals cooperate they get a reward, R,
- if they both defect, they are punished (punishment, P),
- if one player defects and the other cooperates, then the defector gets high reward (temptation, T) and the other gets a severe punishment (sucker, S)
- and $T > R > P > S$, and $2R > T + S$

Under these reward conditions, each individual criminal will reason that if the other

- cooperates, he does better by defecting.
- defects, he also does better by defecting.

The payoff matrix for Prisoner's Dilemma is shown in Table 2.1

Criminals 1 / 2	**Cooperate**	**Defect**
Cooperate	R,R	S,T
Defect	T,S	P,P

Table 2.1: Payoff matrix for Prisoner's Dilemma.

Thus the Nash equilibrium situation (John F. Nash, 1950) for this game is for both players to defect, even though they would collectively get a higher reward if they were both to cooperate. The Tragedy of the Commons can be likened to a situation in which the individual herder is playing the Prisoner's Dilemma game against the collection of all the other herders: his selfish interests lead him to defect, even though they are all better off if they cooperate. A solution to this problem is to provide social incentives such that players are encouraged to cooperate and avoid the sub-optimal solution of defection.

2.3.2 Non-cooperation problem in electronic societies

Electronic societies are also plagued by scenarios that can be likened to the Tragedy of the Commons. Freeriding is a well-known example in peer-to-peer (P2P) systems, where an agent does not contribute to the society by sharing its files but downloads the files from other users. P2P communities heavily rely on altruistic sharing of digital goods such as video files, audio files and e-books. If everyone shares, that is good for the societal welfare. However, if everyone is exclusively self-interested, there would be nothing to share in the community. Though neither of these two extreme cases are seen in real life file-sharing communities, the communities are aware that the freeriders should be kept under a close watch in these communities, since the increase in the number of self-interested people refusing to share leads to the degradation of the system.

Following the rapid rise and fall of the popular file-sharing system Napster (1999), P2P systems such as Kazaa (2001) and BitTorrent (2004) have become widely used. However, uncooperative behaviour is frequently observed in these systems. BitTorrent is currently particularly popular, and Hales and Patarin's analysis of BitTorrent's workings (2005) is of interest. With BitTorrent, groups of users "swarms" with an interest in a specific media file coordinate to speed-up the process. A given file is partitioned into pieces, and each peer is responsible for obtaining and sharing with the other peers some of the pieces. Each swarm is managed by a "tracker", which keeps track of the peers interested in a file or group of files. Peers may query the tracker for a random list of other peers in the swarm, and once obtained, the peers can exchange their piece lists so that they may determine which peers may have pieces that they need. Since a peer may not be able to service at once all the peers that need

its piece, it only services up to a limited number of other peers, with remaining peers being left out "choked". Presumably peers will choose to cooperate with those peers which, in turn, have cooperated with it, and so cooperative behaviour is presumed to be induced by an implicit "tit-for-tat" strategy. But Hales and Patarin determined that it would be easily possible to cheat under these arrangements and be a freerider. Interestingly, such cheating is not observed in connection with BitTorrent, although is observed on other systems. They wondered what the reasons for this more cooperative behaviour might be.

Hales and Patarin (2005) have suggested an answer to their own question which concerns the way that file metadata is handled with BitTorrent. BitTorrent does not provide a central distribution for metadata; instead the acquisition of metadata is left to the users. To download a file using BitTorrent, one must supply information which can be found in a special .torrent file, but the user must use his or her own devices, such as user-run Web sites, to find this file. This means that the connectivity of this "network" of interested users is not complete: separate, possibly somewhat isolated, groups of users will form and share the metadata. Although this is sometimes thought to be a weakness of BitTorrent, Hales suggests that this may be an advantage, since separate swarms with their individual trackers can be formed for the same file. This can lead to a swarm selection process, whereby higher performing swarms (with more cooperative members) are selected and poorly performing swarms (with more freeriders) are deselected and eventually die off.

The suggested mechanism at work here is that, by means of probably unintended limitations in terms of metadata access, there is an arrangement in BitTorrent that can lead self-interested peers to generate multiple groups and a group-swarm-selection process that ultimately yields more cooperative (and hence higher overall performing) groups. Thus by having some restrictions in a group, the Tragedy of the Commons can be avoided.

The mechanism based on restriction is one particular approach to address the problem of freeriding. However, in our view, for agent societies, several mechanisms can be proposed and investigated not only to uncover malefactors, but also to guide and sometimes restrict agent behaviour so that a more cooperative environment is fostered. Thus by having several types of mechanisms that impose certain restrictions in the artificial agent societies, the Tragedy of the Commons can be avoided. The restrictions in this thesis are embedded in the social mechanisms that we have investigated. Our focus is not on expelling the "evil" agents

that do not cooperate in a society (because the "evil" ones can return anyway with a new identity), but to substantially distinguish them from the ones that are good through social mechanisms that help in the self-organisation of groups (or the segregation of groups) based on their cooperativeness.

Our own approach is tailored to segregate groups by making use of social mechanisms such as tagging, referral, voting, gossip, and ostracism to facilitate social control in artificial agent societies. It shields cooperators from exploitation by non-cooperators, thereby leads to te betterement of the society. In this kind of configuration and managerial arrangement, good agents segregated into good groups get good service, and the bad ones end up in groups with bad agents where the service is restricted. In this thesis we have considered different types of societies, such as closed and open, and also have considered different types of control applied to societies such as those organised according to centralised, semi-centralised, and decentralised control. Figure 2.3 shows the two important characteristics considered (i.e. type of society and level of control in a society) in the experimental Chapters 4-7.

The next chapter provides an overview of the social mechanisms used in this thesis and also discusses prior work on achieving segregation using these social mechanisms.

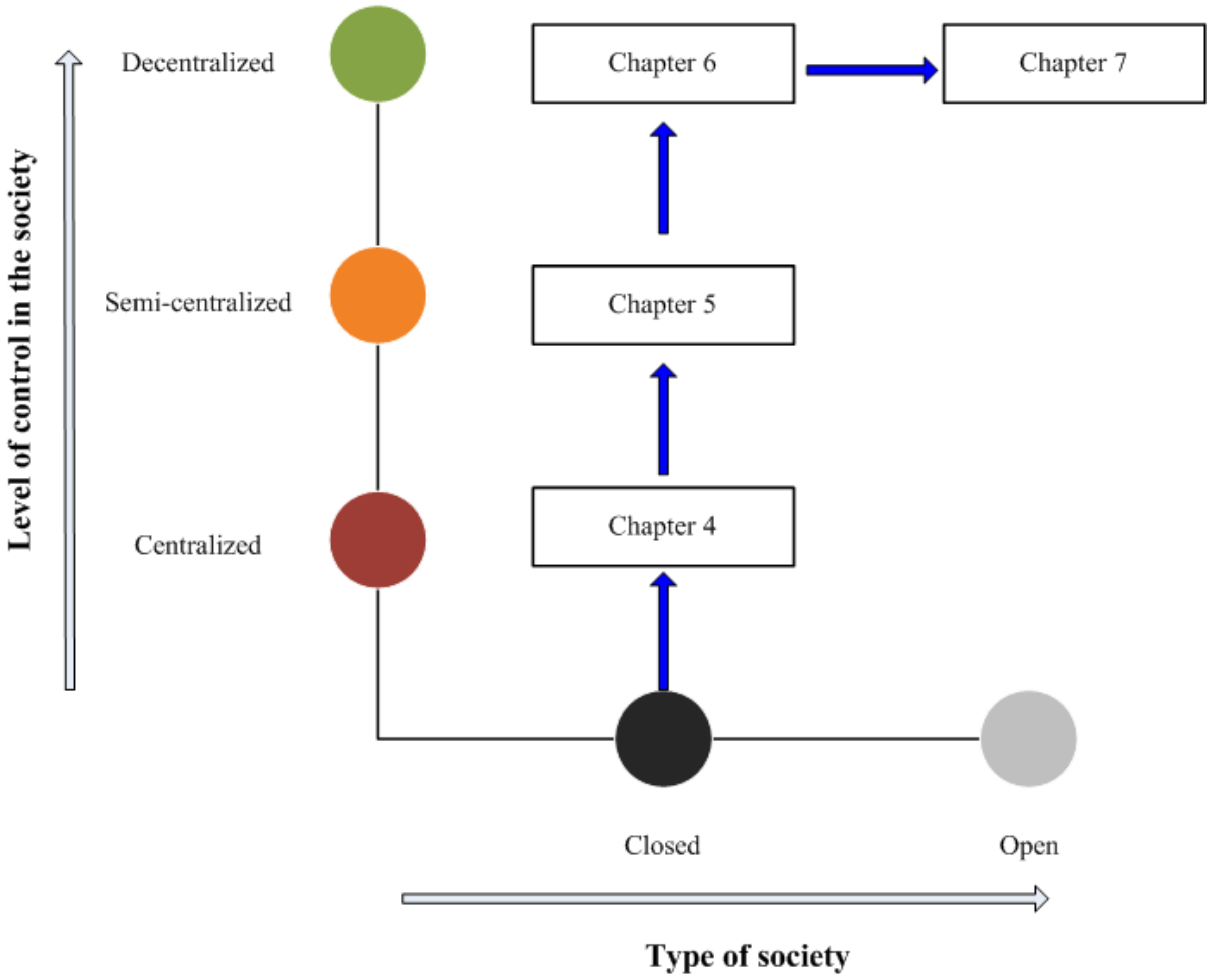

Figure 2.3: Progression of the experimental chapters.

Chapter 3

Facilitating social control in artificial agent societies

> *"We humans do act collaboratively, not out of altruism, but because of social mechanisms that bound us to do so. Human societies have created a myriad of different mechanisms to reduce the inherent complexity and the uncertainty that exist in society. Analogously to human societies, research on multi-agent systems and artificial societies is in its way to create social mechanisms to be applied to computational entities."- (Pujol, 2006)*

This chapter aims at discussing the social mechanisms investigated by researchers in multi-agent systems to facilitate social control in artificial agent societies. Towards this end, Section 3.1 discusses the background on designing agent societies as investigated in the field of multi-agent Systems. In Section 3.2 the three-step process involved in the investigation of artificial agent societies is explained. In Section 3.3 we discuss the segregation of groups within a society. In this context, several social mechanisms that are developed and employed in this thesis are presented in Section 3.4 along with various relevant research works that have investigated social mechanisms for facilitating social control, and Section 3.6 contextualises the contributions of this thesis. Finally, Section 3.7 summarises the chapter.

3.1 Why artificial agent societies?

Social simulation is a research approach which explores the behaviour of societies. Societal behaviour is a complex process. Computer simulations are suitable tools to help explore and understand social processes better, since various 'what-if' scenarios can be explored by changing the parameters of the system.

In the field of multi-agent systems, several researchers have realised the importance of studying artificial agent societies (Gilbert and Conte, 1995; Dellarocas and Klein, 1999; Hales, 2000; Conte, 2001; Davidsson, 2001, 2002; Pujol, 2006). Two important reasons for the interests in these societies include:

a) the ability to investigate social models in this artificial laboratory.

b) the feasibility to apply some well-known social theories to artificial agent societies to improve the societies.

In an early paper, Dellarocas et al. (Dellarocas and Klein, 1999) note that ideas borrowed from human societies can be taken advantage of, by incorporating them into electronic agent societies. Inspired by human societies, the authors argue that there is a need for creating electronic institutions, modeling social contracts, and specifying social rules for interactions between agents in the society.

Soon researchers working along these lines developed electronic institutions (Colombetti, Fornara, and Verdiccchio, 2002; Esteva, Rosell, Rodriguez-Aguilar, and Arcos, 2004; Arcos, Esteva, Noriega, A.Rodríguez-Aguilar, and Sierra, 2005; Boissier, Padget, Dignum, Lindemann, Matson, Ossowski, Sichman, and Vázquez-Salceda, 2006). One of the well-known electronic institution models is described in the work of Sierra and his team (Arcos *et al.*, 2005). They work with tightly controlled systems and have implemented agent-based institutional systems with centrally administered mechanisms that maintain strict control of individual agent interactions within the institutions. The advantage of having a centrally controlled institution is that it is well structured and easy to monitor. The disadvantage is the lack of scalability/flexibility and autonomy of the agents. The agents are heavily controlled in these types of institutions (e.g. a governor checks each message of the agent). Centralised systems can become inefficient over time (due to increase in size), and become costly to maintain, too. They are prone to single-point-of-failure and performance bottlenecks.

While developing a framework for electronic institutions, the infrastructure for facilitating social control in electronic institution is also considered. For example, AMELI (Esteva *et al.*, 2004) is a middleware which prevents agents from violating rules in electronic institutions. This infrastructure enforces institutional rules and controls agent interactions. AMELI's architecture (Esteva *et al.*, 2004) has four types of higher authority agents who act on lower-level interacting agents. The institution manager, transition manager, and scene manager take care of institution execution, transition and scene related functions. The governor agent is an interface that the institution provides which must be used by agents external to the system to communicate with other agents. Thus, the role of the governor agent is to monitor the messages sent by the external agent. Hence every external agent has a governor that is assigned to it. In this system, the communication between the external agents and the agents in the system is vetted by the governor agents, while the other activities in the institution are managed by the corresponding managers. Thus this system is a regimented system where agents are heavily controlled by strict institutional rules and authorities. Though the system allows other agents to join, these are always monitored by the governor agents. The system is thus a semi-closed system that does not let the agents act on their own will. In other words, the agents in this system are not allowed to act without the interference of the monitors. In our own work we prefer to have our system behave desirably and maintain the structure/order without compromising the agent's autonomy. Attaining social control by using social mechanisms is the way to achieve it.

There are research works that have proposed mechanisms for establishing social control in agent societies (Rasmusson and Jansson, 1996; Dellarocas and Klein, 1999; Dellarocas, 2000; Castelfranchi, 2000). An approach that makes use of reputation is a "soft security"-based approach where users monitor themselves and avoid having a centralised control (Rasmusson and Jansson, 1996). If users can establish social control among themselves, then there is no need for any kind of top-level control (global authority). The basic idea of this system is to achieve social control by means of using reputation as part of a security mechanism. Agents themselves take the responsibility of establishing a safe and fair society without involving any external force acting on them. It is thus a self-policing mechanism. The advantage of using a reputation-based social mechanism is that it is distributed and helps to deal with the malicious users present in the society without relying on an authority. The

malicious users can be identified and potentially ignored (e.g. ostracised) by others through social mechanisms. This process helps to prevent users from being harmed by the malicious users. Researchers using the soft security approach have defined social control (Rasmusson and Jansson, 1996) as a "group behaviour that indirectly forces the group members to behave in a particular way". Social control is thus a bottom-up approach implemented in a distributed fashion among users to help achieve cooperation in open societies that are vulnerable to selfish behaviour of malicious users. This work (Rasmusson and Jansson, 1996) notes that it is a good idea to employ soft security to enable secure e-commerce transactions in open environments where no central monitor keeps track of the actions of the individuals.

Based on the fact that human societies are well-structured by social contracts, the framework for electronic societies based on civil agent societies presented by Dellarocas and Klein (1999) has captured a set of processes and elements that human societies possess in order to establish social control. They have proposed the use of social contracts for agent mediated electronic marketplace. They have recommended the use of norms, institutions, and mechanisms for making social contracts which would aid to control the selfish behaviour in the open multi-agent society.

In line with this approach, Dellarocas (2000) has proposed the design of Contractual Agent Societies (CAS), where autonomous agents playing different roles coordinate with each other through social contracts. With these contracts the interactions between agents are clearly defined. This setup helps to achieve coordinated social activity. Social control is established, by which the system shows desirable behaviour by preventing possible deviations.

Due to the open nature of the systems, establishing social order as an emergent behaviour by means of local interactions would be preferable to having the system hardwired in order to achieve a predesigned social order (Castelfranchi, 2000). Since modern open systems have a dynamic nature (i.e. they can change unpredictably due to varying conditions), the social order should also be established dynamically considering the current state of the system. The system needs to be adaptive with built-in adjusting capabilities. The social order can be produced by having some form of social control (Castelfranchi, 2000). There are several research works which make use of a normative approach for social control (Conte, 2001; Grizard, Vercouter, Stratulat, and Muller, 2007; Boella, van der Torre, and Verhagen, 2006; Vazquez-Salceda, Aldewereld, and Dignum, 2004). Social control has been achieved by

having other mechanisms such as sanctions, incentives and reputation in place. Castelfranchi (2000) discusses different forms of social control. In particular he distinguishes centralised social control mechanisms from decentralised mechanisms. In this thesis, we are interested in establishing decentralised social control, since it is a bottom-up approach which can lead to emergent, self-organising behaviour of the system. We intend to develop and employ different types of social mechanisms to achieve this. Note that using norms in an electronic society is one approach for monitoring and controlling the agent behaviour. However, there could be several other approaches that are possible for monitoring and controlling the agents (e.g. referral and gossip mechanisms).

We have developed and employed social mechanisms for agent societies to achieve the desired result of *segregation* of groups of agents with similar behaviour (cooperativeness). We describe these social mechanisms in more detail in Section 3.4.

3.2 Processes used in managing agent societies

In this section we identify three main steps of the process associated with specifying and managing agent societies. These three steps include the following.

- Step 1 - Agent interaction in an agent society
- Step 2 - Monitoring agent interactions
- Step 3 - Controlling agent interactions

In order to elaborate these three steps let us consider an example from Axelrod (1986). His work captures the emergence of cooperation in social dilemmas through norms without having a central control. In his paper (Axelrod, 1986), agents interacted with each other in the form of a simple norms game that was played between agents (Step 1). Step 1 corresponds to the context of agent interaction. In this example the context is the norms game. Agents were supposed to monitor other agents for the violations of certain actions (i.e. violation of norms). This corresponds to Step 2. If violations are detected, those agents are expected to punish the violating agents (i.e. controlling). This corresponds to Step 3. There was also a meta-level punishing (i.e. agents were expected to punish those non-punishers who failed to

punish a violation). Here, a meta-level monitoring and controlling was in action. In general, agent societies employ these three steps when investigating social control mechanisms.

We note that Step 1 in agent interactions can be based on informal norms and/or through formal rules established by the institution. Agent interactions are often modeled in the research literature using a game-theoretic approach (e.g. cooperation or coordination games). This thesis places an emphasis on the Steps 2 and 3 where several reputation-based social mechanisms are employed.

3.2.1 Reputation-based mechanisms for social control

Electronic marketplaces are increasingly being used. Some of the well-known examples include eBay (1995), TradeMe (1999) and Amazon.com (1995). These market places facilitate trading (buying and selling) goods online, and they provide mechanisms that ensure a fair trade where the expectations of buyers and sellers are met. These systems also provide mechanisms for specifying and monitoring obligations (e.g. a buyer should pay within 3 days after winning an auction). These mechanisms ensure secure and safe trading in these electronic market places.

To ensure security and safe trading, online markets have a mechanism of users rating each other based on their interactions. Consider eBay (1995) as an example, which is currently the world's largest electronic market place. In eBay people leave feedback about others whom they have had transactions with. Users could give positive, neutral or negative feedback. The feedback about users is associated with their profile and is visible to everyone. The reputation of a user (both as a seller and a buyer) is calculated by the overall ratings. In this approach the users are expected to report honestly and this approach works well for the current system. The user has no control over his ratings being visible to everyone using the system. Ebay displays the reputation records about all its members. It suspends or removes users violating the rules (Omidyar, 2011). People generally tend to behave honestly and show their good side, since their reputation score is visible publicly. The advantage of this kind of social control is that it enables improvement in electronic societies. The disadvantage is that the user loses control to the central authority which stores the user's information. The user has no control over his/her information being visible to everyone. We believe decentralised

mechanisms where information is distributed across different entities can be adopted in order to achieve the same purpose. For instance, partial information about the reputation of users can be held by distributed peers.

Once the information about agent behaviour has been collected (either through centralised or distributed mechanisms), this information can be used for two purposes in the context of identifying good behaviour and discouraging bad behaviour in the system. Firstly, this information can be used for producing recommendations. For example, an agent can vouch for another agent based on that agent's interactions (i.e. through a referral process). An agent can also identify the best agent or a set of good agents that have cooperated in the past (e.g. in the context of voting in a society for the best agent, agents can nominate the best one based on their past interactions). Secondly, this information can be used for controlling bad agents. For example, the reputation score can be used to restrict resources for bad agents and can even be used to sanction bad agents in a society. Reputation-based mechanisms are shown to solve social dilemmas (Milinski, Semmann, and Krambeck, 2002).

Monitoring agent interactions can be achieved through two approaches. The first approach is to use a centralised reputation system which collects all the information (e.g. eBay, TradeMe, Amazon.com) as explained before. Since there are several disadvantages of this approach, several multi-agent researchers have investigated reputation-based systems for monitoring agent interactions (Yolum and Singh, 2003b, 2005; Candale and Sen, 2005; Gursel, Sen, and Candale, 2009) which use distributed approach. They make use of a decentralised, partial reputation system such as the use of referral or gossip in agent systems. In these systems, each agent has only partial information. These types of systems are more scalable and decentralised. We have adopted a similar approach of using several social mechanisms to establish social control in this work. More details are presented in Section 3.4.

3.3 Separation of groups within an agent society

The previous section discussed how reputation-based systems might be used for monitoring and controlling in an agent society. This section describes how several groups can be formed within one society based on employing some social mechanisms. Having numerous agents in a society can become unmanageable. By having smaller groups of agents in a society, the

groups can be more manageable by reducing the complexity of the individual group. For example, if there are more agents joining an open society (imagine a very large group), it may become unmanageable and may lead to scalability issues. Instead, more and smaller groups could be formed if an excessive number of agents join the society, and that can make things more flexible and scalable. Two mechanisms can be used for grouping purposes:

- Tag mechanism

  A tag mechanism can be used to form groups. It is a simple group-forming mechanism, and it is explained in detail in Section 3.4.4.

- Reputation-based mechanism

  Initially a society may employ a tag-based mechanism for assigning members into groups (for bootstrapping purposes), and later these groups can evolve to be cooperating or non-cooperating groups (based on cooperativeness of the agents) when reputation-based mechanisms are used. This is the scheme used in this thesis.

3.3.1 Group segregation

Schelling's model (Schelling, 1969) was the first which showed segregation of groups as an emergent behaviour based on micro interactions among agents. In this model there are two types of agents, with red and green colors. An agent is satisfied (according to its satisfaction level), if it is in the cluster of similar color agents. Otherwise the agent can move to a random place. This model showed the segregation of similar color clusters forming over time corresponding to the satisfaction level of the agents. The advantages of segregation based models are

a) the model does not need a global view (local view/partial information is enough).

b) the model is scalable for large numbers of agents.

c) the model has the flexibility to cope with random arrival and departure of agents.

These properties of segregation-based models make them suitable for designing distributed dynamic systems in open environments. The advantage of grouping has been shown to be beneficial in achieving the emergence of cooperation in prisoner's dilemma interactions (Hirshleifer and Rasmusen, 1989; Oh, 1999). Martinovic's work has adopted such

a segregation model (Martinovic, Leng, Zdarsky, Mauthe, Steinmetz, and Schmitt, 2006). It is emphasised in this connection that during the segregation it is important to get into a good neighborhood, since such locations are essential for self-protection against exploiters (freeriders). Being in a good neighborhood also helps in obtaining useful information and better service. The freeriders, on the other hand are moved to bad neighborhoods. This is similar to our own work in the sense that cooperators move to good groups and non-cooperators are moved to bad groups.

3.4 Research works using different social mechanisms

In this section we discuss social mechanisms that we have developed in this thesis to facilitate social control in multi-agent systems. These social mechanisms include gossip, ostracism, referral, voting, and tagging. These social mechanisms are described in subsections 3.4.1 to 3.4.4. Each subsection provides a brief introduction to the social mechanism and discusses similar work that has used these mechanisms in agent societies.

3.4.1 Gossip

Gossip is a powerful mechanism in human society for information sharing. Research done by evolutionary biologists suggests that humans have shown more interest in gossip more than in the original information (Sommerfeld, Krambeck, Semmann, and Milinski, 2007), when participants were presented with both types of information (the gossip information and the "real", original information). Based on their research they have noted "gossip has a strong influence... even when participants have access to the original information as well as gossip about the same information" and also that "gossip has a strong manipulative potential".

In a way, a gossip mechanism can be considered to be a 'distributed referral' mechanism (Eugster, Felber, and Le Fessant, 2007). It is similar to having a reputation system, except that the gossip information is distributed. Paine (1967) has explained that people who gossip within their gossip circle feel the fellowship or belongingness to the community. In fact gossip is a property of a group (Paine, 1967) that can be used to provide local social control; it aids in maintaining the social structure. Gossip can also be considered as an indirect

attack on a person where no other easy way of sanctioning is possible. Thus gossip is a mechanism for transmitting public opinion which can lead to some social benefit (Stirling, 1956). Gossiping is also a way of doing social comparison (Suls, 1977; Paolucci, Marsero, and Conte, 2000).

The grooming behaviour in animals is analogous to the gossip behaviour in humans (Dunbar, 1996). Dunbar's work (2004) also acknowledges that gossip is a way of social bonding, and it is a part of social life. An additional benefit of gossip is that it helps to control freeriders. Gossip information about freeriders is important, because people do not want to be exploited by a freerider, and freeriders degrade the societal welfare. The most common way of controlling freeriding is based on memory of past behaviour (Dunbar, 2004). Gossip helps this by being a medium for 'information storage and retrieval' (Roberts, 1964). The work of Conte and Paolucci (2002) considers gossip as a medium to spread and retrieve the reputation about an agent which aids in establishing social control.

De Pinninck et.al.'s work (de Pinninck, Sierra, and Schorlemmer, 2008) has adopted the gossip mechanism to achieve a similar goal as ours. In their work gossip has been used as an identification mechanism to spread and find information about the norm violators. In our work we use gossip for a similar purpose, which is to identify freeriders. In their work (de Pinninck *et al.*, 2008), gossip is shared locally with nearby agents called 'mediators', and it does not need complex computations. The mediator agent contacts the enforcer agent to punish the violator. Gossip has been used in relation to the norm enforcement technique in their work, and gossip has been also used for reputation management in Yu and Singh's work (2000).

In a recent work of Mordacchini et.al. (Mordacchini, Baraglia, Dazzi, and Ricci, 2010), they have proposed an architecture for gossip-based peer-to-peer systems which facilitates information exchange among peers through gossip. Their system groups users with similar interest (or users who are interested in similar contents) by the distributed process of sharing information through gossip. This sort of grouping has an advantage of making recommendations accordingly, since those groups of users share a common interest. It is different from our work, since our work groups agents based on behaviour (cooperativeness). The work of Khan and Tokarchuk (Khan and Tokarchuk, 2009) proposed a group-structured P2P system which also uses gossip to form groups with peers of common interest. Both (Mordacchini

et al., 2010; Khan and Tokarchuk, 2009) have used gossip to form groups based on peers looking for similar content. In our work we, too, use gossip to form groups (based on behaviour), but content searching is not the focus in our work.

The work by Ganesh et.al. (Ganesh, Kermarrec, and Massoulié, 2003) has proposed a similar gossip-based information dissemination for P2P membership. It has used a gossip-based distributed mechanism which does not require a global view, so it uses less memory and leverages scalability and decentralisation. These aspects are similar to the approach used in this thesis, but our focus is not P2P membership. Instead our focus is on forming dynamic groups automatically by enabling self-organisation of agents into groups of different levels of cooperativeness to avoid exploitation by freeriders.

There is other work in agent-based simulation and P2P systems which has used gossip-based protocols (Kempe, Dobra, and Gehrke, 2003; Jelasity, Montresor, and Babaoglu, 2004, 2005; Gorodetsky, Karsaev, Samoylov, and Serebryakov, 2007; Zaharia and Keshav, 2008; Dasgupta, 2003; Boyd, Ghosh, Prabhakar, and Shah, 2006). Gossip-based protocols broadcast information, and they are based on the way information spreads in the form of rumors in human societies. The information is shared between one-to-many individuals. In these protocols the number of participants (with whom the information is shared) increases each time. In this fashion the information is spread to everyone in the society in a short period of time. The disadvantage is that lots of resources are used for this sort of information sharing without realising whether the information would be useful to the recipient. This results in information flooding which overloads the system (i.e. causes congestion and performance bottlenecks) (Li, 2008). Instead, it could be made more simple and useful if the recipient could get the information by requesting, rather than having all types of information being overloaded through a flooding model. Following this request-based approach, the user has an option of choosing the information he/she wants and avoiding the unnecessary information. This approach does not overload the system, and it needs less computation compared to the former approach. Moreover, the user has the control over the information he/she is being exposed to. Our approach uses such a kind of gossip mechanism where the user can specifically request for information that he/she needs.

3.4.2 Ostracism

It has long been the case in human and animal societies that the member of a group who does not abide by rules or norms can be punished by other members of the group (the followers of the rule/norm). One kind of punishment is ostracism, which results in the social exclusion of the punished member. All the other members of the society would stop interacting with the member who is being ostracised and would no longer consider that person to be a part of their group (by ignoring him/her). This kind of behaviour is used as a social punishment mechanism where there is no higher authority or institutional monitor to check deviations and establish punishment. Thus it is a decentralised mechanism as opposed to having a central controlling authority to establish sanctions.

There is work which has used ostracism for improving cooperation where the context of interaction has been modeled using the Iterated Prisoner's Dilemma (IPD) game (Hirshleifer and Rasmusen, 1989; Oh, 1999). Cooperation is achieved by a grouping mechanism. Players who persistently defect are expelled from the group as a way of punishment by other players. Thus group ostracism is used to protect cooperators from the defectors. Ostracism is used to expel defectors from the cooperative groups. By doing this, cooperators would only have other cooperators in their groups and they play only with them. By imparting the fear of punishment, cooperation is enforced. Thus having ostracism as a punishment mechanism has helped to control defectors and promote cooperative behaviour in the IPD game. Thus cooperative groups of similar players are formed and emergence of cooperation is achieved.

Pinninck's work (de Pinninck *et al.*, 2008) has used an ostracism-based mechanism to punish norm violators in an open multi-agent system. Their work makes use of the concept of a normative reputation for each of the agents in the society. Depending upon whether an agent abides by the norm, its reputation is spread through gossip. Agents with a bad normative reputation are ostracised by the members of the society (i.e. agents stop interacting with a norm violator). By ostracising the norm violators, agents achieve better payoffs by interacting only with the normative agents. Thus the norm is enforced, and the norm violators are punished in an open agent society. Similarly, our work also uses ostracism along with gossip to identify freeriders and restrict exploitation in a simulated P2P environment.

In his book on ostracism, Williams (2001) explains different types of ostracism. The

ostracism that we use in this work falls under the type of punitive ostracism described by Williams. Punitive ostracism is where an individual is ostracised by others as a punishment (by avoiding interactions or ignoring). In our own work an agent is ostracised for being non-cooperative (punitive ostracism).

3.4.3 Referral and voting

Referral is a useful mechanism when there is no prior experience or information about a new person or a new service. For example, when someone applies for a job, the company that advertises the job asks for referrals (e.g. letters of recommendations or references) from the present or previous employers of the candidate. Referral is first-hand experience shared by another agent about an agent in question. It differs from gossip, because gossip is second-hand information which is normally about the experience of a third party which is shared between agents. For example, agent A may consider B to be of high repute. It may recommend B to C, so it is referral. C may pass on this information about B to D and E, which is then gossip.

Referral mechanisms are useful in many domains including P2P (Yu and Singh, 2002; Yu, Li, Singh, and Sycara, 2004) and have also been demonstrated by multi-agent system researchers in other contexts, such as selecting a service provider (Yolum and Singh, 2003b,a, 2005; Candale and Sen, 2005; Gursel *et al.*, 2009).

Condale and Sen's work (Candale and Sen, 2005) uses a referral mechanism to choose a service provider. They have considered using referrals not only for picking a service provider but also for choosing an agent who gives good referrals. Choosing a good agent for getting referrals improves the chances of finding a good service provider. Another issue is that the activity of referring to a good provider over time would make the provider more busy, and the service may degrade over time. So they have considered two further aspects, such as when and whom to ask for a referral. They have shown that using referrals appropriately delivers satisfying results. They have also considered negative referrals, where agents may provide false information in referrals just to protect their service providers from service deterioration caused by overloading. In this thesis, we don't consider such misleading behaviour on the part of agents.

In the work of Yu and Singh (2002), they have used referrals for selecting reliable services so that agents could find trustworthy agents through referrals. The frequent interaction and adaptiveness of agents make the referrals more efficient and that leads to the emergence of referral networks. These referral networks are the key components for the emergence of social structures in their system. In their extended work in the e-commerce domain (Yu *et al.*, 2004), they have investigated the link structure of the network and how such systems emerge to become self-organising referral networks.

Voting mechanisms are related to referrals. Once agents have information about others, they can use this information for providing recommendations (i.e. a referral can contribute in choosing the best agent through a voting process. Several agents can also vote on the 'goodness' of a particular agent). Voting is particularly suitable for systems with a top-level authority, because voting requires an authority to tally the votes and enforce the decision obtained through voting in the society. Agents typically vote based on their personal experience, which is plurality voting. The definition for plurality voting, as stated by (Shoham and Leyton-Brown, 2009), is "Each voter casts a single vote. The candidate with the most votes is selected". In plurality voting, each agent votes for one of the candidates (Shoham and Leyton-Brown, 2009). In this voting system the person with the most votes is selected as the winner and there is no requirement that the winner should get majority of votes. Voting is suitable for systems in which there is a higher-level authority who can tally the votes and enforce the decision. The voting may not be suitable for systems without such an authority.

In our work voting is performed to establish the most and least cooperative player of the group. The system uses local group monitors to enforce decision, and it is explained in detail in Chapter 5.

3.4.4 Tagging

In this thesis tag-based mechanisms are used for group formation. This subsection provides background information on tags. Tags have been used in modeling artificial societies since John Holland proposed their usage (Holland, 1993). The tags which are modeled in multi-agent-based simulations simply represent markings that are visible to other agents and are used for grouping purposes. Examples of these tags in nature include birds of the same

feather flocking together and animals that look similar to each other coming together to form a herd. Each animal can be thought of as having the same tag (e.g. same appearance). They interact within the group identified by all the members having the same tag. Thus the tagging mechanism that we use is inspired by nature, and it has been widely used to model grouping behaviour in artificial agent societies. Note that tags in nature are often permanent markers identifying the species. But tags can also be temporary markers that are acquired or lost in connection with a particular social context. It is this more flexible and dynamic aspect of tags that is emphasised in our work[1]. A straightforward way to think of the tags used in artificial societies is to assume that they represent group names for sets of agents: agents having the same tags belong to the same group, and agents of the same group have some preference to interact with others within their group. Thus people are usually friendly to others who are similar to them (belong to the same group of interests, education, ethnicity, profession, culture, personality). They choose their friends, partners based on certain similarities that are assumed to represent compatibility.

Various researchers have characterised what tags are in slightly different terms.

- According to Riolo (Riolo, Cohen, and Axelrod, 2001) a "tag can be a marking, display, or other observable trait. Tag-based donation can lead to the emergence of cooperation among agents".

- According to Hales (Hales and Edmonds, 2003b) "tags - observable social cues or markers attached to agents. These tags are visible (readable) by other agents allowing them to distinguish between agents with different tags".

Tagging offers a simple mechanism that can facilitate cooperative behaviour on the part of selfish individuals. Individuals prefer to interact with other individuals who are observed to be similar to them (because they have the appropriate tag).

Tags offer several advantages. Using tags is relatively simple when compared to other complicated mechanisms that are used to achieve cooperation/altruism. For example, other

[1]There are two ways of assigning tags to an agent in an agent-based simulation system. First, an agent can be assigned to a tag-group at the initialization phase. Second, an observer can tag an agent based on an observed attribute.

known cooperation mechanisms used are direct and indirect reciprocity (Nowak and Sigmund, 1998; Trivers, 1971), kin selection (Hamilton, 1964a,b; Axelrod, Hammond, and Grafen, 2004), and centralised control mechanisms. In the reciprocity mechanism, keeping the memory of past interactions is needed. In kin selection, it is necessary to have a good recognition mechanism to identify the kin. Centralised control systems require a monitor to employ punishments or offer incentives, which represents a performance bottleneck and does not scale as well as decentralised systems. In contrast, a simple tagging mechanism does not involve additional overheads, such as memory storage, maintaining reputation records, and monitoring logs.

In this connection some researchers have shown that tag-based mechanisms have been successful in the evolution of cooperation using the Iterated Prisoner's Dilemma/ Prisoner's Dilemma (IPD/PD) games (Riolo, 1997; Hales, 2000, 2001, 2002; Hales and Edmonds, 2003b; Hales, 2004b). Riolo (Riolo, 1997), has shown how cooperation is achieved by using tags in playing the pair-wise Iterated Prisoner's Dilemma game. Tags achieved cooperative behaviour in that work by biasing the partner selection towards similar tags. This biasing achieved cooperation among agents after enough iterations of the games were played between the interacting agents.

In Hales's work (Hales and Edmonds, 2003b) different types of tags were used to achieve cooperation in different scenarios, such as Prisoner's Dilemma, resource-sharing and load balancing. In a subsequent work Hales (2008) compared the various tag models. In other work by Hales's (Hales, 2004b), a tag-based P2P system was proposed, where tags are markers that are visible to other agents. Agents interact with other agents that have the same tag. The Prisoner's Dilemma game was used to model the interactions among the P2P nodes for sharing files. This work extends his previous work on tags, to networks, where a neighbor list of nodes is considered to be a tag, and the movement of a node from one place in a network to another is modeled as a mutation. His results showed that tags work well for P2P systems in achieving cooperation, scalability and robustness.

In another work by Issac Chao (2007), it is reported that tag recognition does better than multi-agent based learning mechanisms in one-shot PD game. Using experiments conducted using simulations, the performance of tag-based mechanism is compared with other learning mechanisms including evolutionary algorithms. Tag-based mechanisms showed promising

results. Learning mechanisms might fail due to computational requirements (limitations) required in an open agent society where agents join and leave a society at run-time. A simple genetic algorithm applied on the whole population just makes copies of the fittest whereas the tag-based mechanism incorporates the same evolutionary algorithm across different tag-groups, thus enabling the scalability of an open agent society. It should be noted that an evolutionary algorithm performs better when compared to other learning mechanisms, the evolutionary aspect combined with a tag-based system performs the best in facilitating co-operation in an agent society.

All these works use tags as a group identifying mechanism to form teams and to improve cooperative behaviour within the group, which eventually leads to the cooperation of the entire society. In this thesis, tags are used as simple markings to form groups (in Chapter 4) where agents interact based on tag matching. We used tags for forming groups and also 'colored' tags are used for defining a simple meaning, implying the status of the agent (in Chapter 5). In later parts of the thesis, tags are used as bootstrapping mechanisms to form initial groups, and later the groups evolve based on agents' behaviour (in Chapters 6 and 7).

3.5 Altruism

Altruistic behaviour is commonly observed in the animal kingdom, particularly in species which have complex social structures (e.g. vampire bats, insect colonies and some bird species) (Okasha, 2009). Altruism exists in human societies too.

Altruism has been of interest to researchers in the fields of sociology, psychology and computer science (Axelrod *et al.*, 2004; Batson, 1991, 1998; Tankersley, Stowe, and Huettel, 2007; Trivers, 1971; Sober and Wilson, 1998; Rushton and Sorrentino, 1981; Lehmann and Keller, 2006; Németh and Takács, Németh and Takács; Fehr and Rockenbach, 2003; Fehr and Fischbacher, 2003). Multi-Agent Based Simulations (MABS) provide the platform for scientists to experiment with such models (Epstein and Axtell, 1996).

We give money to beggars or charity without expecting reciprocity. We know certainly that they would not be returning the favor to us. But still we help them. People help strangers at random places. Those helpers don't know if the favor will be reciprocated because the chances are very slim, but they still help strangers. Even though we live in a materialistic

world, people do help others altruistically without self-interest. We could see lots of examples when calamities occur. Sometimes people offer help anonymously which means they would receive no benefit for their altruism. By these examples, we see that altruism is built in. Sometimes people show altruistic behaviour towards others, sometimes they do not show.

In brain science research (Tankersley *et al.*, 2007; Rilling, Gutman, Zeh, Pagnoni, Berns, and Kilts, 2002), it has been revealed that people are naturally moderately altruistic. They discovered that altruistic behavior is hard-wired in the human brain and it is pleasurable to humans. Research in bioethics also claims that people who adopt the altruistic decision are likely to be the happy people in their lives (Post and Neimark, 2007).

Whether it be our real life human society or (any form of) the electronic digital society or virtual world, the basic truth is that some sort of altruism prevails in there which keeps things going, making it a better place (still).

3.6 Contextualising the contribution of this thesis

This section contextualises the contributions of this thesis with respect to the three-step process of investigating agent societies (see Section 3.2). The three steps of the process are given in three rows in Figure 3.1.

- Step 1 - Agent interaction

  The context of agent interaction is depicted in the upper row of Figure 3.1. In Chapter 4, agents interact with each other in the context of playing the Knowledge-sharing game. In Chapter 5, the Prisoner's Dilemma game is used to model the agent interactions. In Chapters 7 and 8, the context of interaction between agents is file-sharing in electronic societies. In all the four chapters *tags* have been used for group formation.

- Step 2 - Monitoring interaction

  The monitoring process is depicted in the middle row of Figure 3.1. In Chapter 4, monitoring is handled by the system itself. In Chapter 5, the behaviour of agents is monitored by other agents in a group, and the *referral* mechanism is in place to spread the information. Referral is used along with *voting* to choose agents based on their

behaviour (i.e. best or worst). In Chapters 7 and 8, decentralised systems are modeled. We move away from the global view of information by considering partial view of information and aggregation using our *gossip* mechanism.

- Step 3 - Controlling interaction

  The control process is depicted in the lower row of Figure 3.1. In Chapter 4, system control is used. In Chapter 5, *resource restriction*[2] is used as a controlling mechanism for exploiters and also to restrict access to them. In Chapters 7 and 8, *ostracism* is used as the control mechanism.

In this thesis, we have employed a mix of social mechanisms to accomplish social control. We have used tags for forming groups, referral to get first-hand information, and voting to obtain the public opinion. Gossip is used to share the second-hand information. Resource restriction and ostracism are used as controlling mechanisms to restrict access to exploiters. These mechanisms employed in agent societies help them achieve social control by structuring the society which enables betterment of the society as a whole. Overall, this thesis has adopted a novel approach of establishing social control through several socially-inspired mechanisms in artificial societies.

Employing these mechanisms in agent societies results in improvement of the society by restricting exploitation of agents. In other words uncooperative behaviour in the artificial agent societies is curbed by making use of social mechanisms. This thesis systematically develops and tests computational mechanisms for controlled agent societies and gradually moves to develop more advanced and scalable mechanisms for fully decentralised societies and also from closed to open societies. Our goal has been to develop computational mechanisms to support distributed systems that are more open, scalable, decentralised, adaptive and self-organising without any top level control by employing our agent-based social mechanisms in place.

[2]Explained in Chapter 5

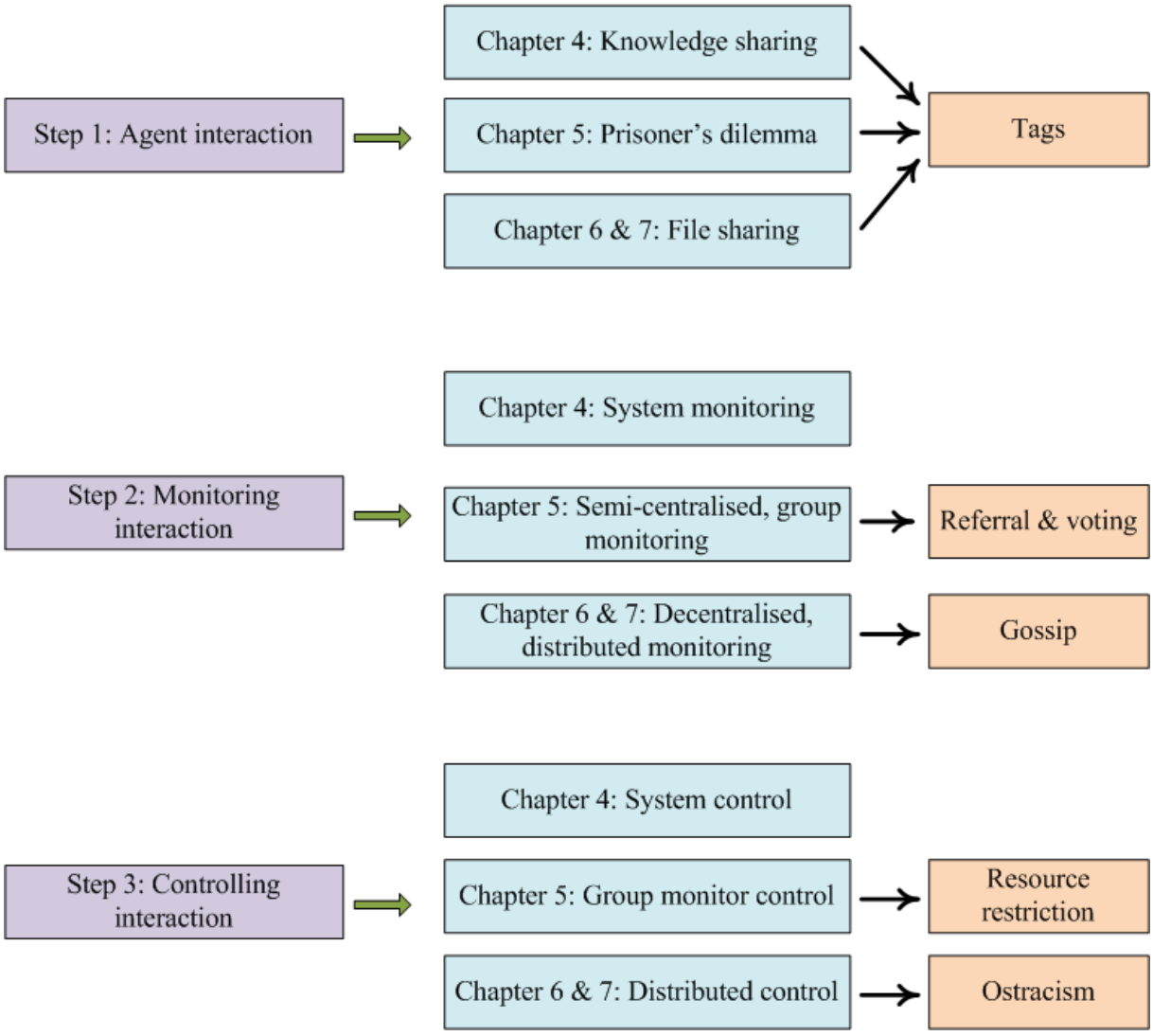

Figure 3.1: Contextualising the experimental chapters.

3.7 Summary

In this chapter we have discussed the social mechanisms to facilitate social control in artificial agent societies. We have presented the three-step process involved in the investigation of artificial agent societies and discussed the social mechanisms employed for the segregation of groups within a society. For each social mechanism various relevant research work was discussed. The next part of the thesis presents our specific computational models and mechanisms and demonstrates the empirical tests that have been performed.

Part II - Socially-Inspired Models

Chapter 4

Knowledge sharing in agent societies

4.1 Introduction

Both human and animal societies have an innate ability to operate in groups. In the animal kingdom, animals often live in groups. Humans too, form societies and often work together as a group: we believe in working together to achieve something as a group which we cannot achieve individually. For example, even primitive hunter-gatherer groups have been shown to have exhibited group tendencies to obtain food (Clutton-Brock and Parker, 1995), and it is common to observe group-supporting behaviour in animal societies (Dugatkin, 1997; Wilkinson, 1984).

For human beings, group mechanisms have provided social machinery that enables co-operation and collaboration. It has been observed in nature that animals that look similar often form a group. Members of a group have an identifier (a tag) that helps them associate with each other. Such tags can be differently interpreted by external observers.

It is frequently observed that when a task becomes too difficult for a single entity to perform, some kind of collaboration or cooperation arises between individuals. For example in the hunter-gatherer societies, it was difficult for an individual to hunt a mammoth, so members hunted as a group. There was an advantage to form their own group, which gave them strength to defend themselves and also provided more hands to work towards obtaining food. Cooperation is the key to the group's achievement.

An important aspect of this collaborative behaviour is the sharing of knowledge or skill

sets among a group. Our concern in this chapter is to experiment with tag-based mechanisms, where groups are formed using tags. Members that belong to a particular group share their skills with other members of their group. In this chapter, we seek to identify those conditions under which a tag-based mechanism produces satisfactory results when there are multiple, competing groups present in the environment. Our experiments in this area were conducted in the context of agents engaging in a knowledge-sharing game.

Consider a scenario where there are different groups that use different techniques for cultivating a crop. The group with the best technique might have a higher yield, hence this group can be considered as outperforming the others. Eventually the other groups will likely follow the technique of the successful group. In other words, the other groups improve (convert to the best group by following their technique). This is a simple example of the conversion process, which can lead to the betterment of a society in many cases.

Another example of conversion is the adaptation of ideas. In the academic research domain, we may be influenced by ideas reported by different research groups and subsequently embrace valid ideas of those groups that enhance our own work.

Thus conversion is a mechanism that has been present in human societies for a long time, such as converting people from a conquered land to adopt new customs, beliefs, skills, and even religion. Traditionally, new members that are being inducted into a group take up the new skills in order to enhance their prospect of success[1]. The strategies employed by the winning group are considered to be the successful ones (at least for the time being). In this work, we have adopted a conversion mechanism in connection with playing a knowledge-sharing game for the betterment of the society.

In equitable societies, the cost of communal services (such as the cost of road works, setting up parks) is shared. But, in some cases, it is best for an individual to bear the costs, rather than dividing the whole cost over the entire society. In this work we demonstrate one such example (in Section 4.3.9) where the whole society is better off when some individuals bear the cost of sharing.

[1] Success is determined by wealth which leads to survival.

4.2 Outline of basic experimental mechanisms

Our concern is to experiment with tag-based mechanisms, where groups are formed using tags. Members that belong to a particular group share their skills with other members of the group. To start with, not all members in the society might be skilled in performing a task, and also not all members that possess the skill might want to share it with their group members let alone other group members. We have experimented with different scenarios to observe whether cooperative (sharing) behaviour can be induced in those setups.

- **Tagless model**

  In Sections 4.3.1 and 4.3.2, we show the baseline experiment when there are no tags in the system.

- **Tag model**

  We show how sharing occurs when there are tags in the system and discuss how tags help in the process of sharing in Sections 4.3.3 and 4.3.4.

- **Tag-and-trait model**

  We describe in Sections 4.3.5 and 4.3.6, the conditions under which tags are suitable to facilitate altruism and when they are less suitable for this purpose. We have also discovered a necessary condition for the society to evolve to "All-sharing".

- **Conversion model**

  We show how knowledge-sharing can be made possible even in the presence of non-sharing agents in the population. This is discussed in Sections 4.3.7 and 4.3.8.

- **Individual versus group cost sharing model**

  Additionally, we have experimented with two cost sharing mechanisms, individual cost-bearing and group cost-bearing. We present the mechanisms in Sections 4.3.9 and 4.3.10.

4.3 Experimental setup

In our own work we have employed a straightforward and practical model for examination: knowledge sharing which deals with sharing knowledge within a society composed of sharers and non-sharers. Note that knowledge-sharing differs from resource-sharing (Hales and Edmonds, 2003b; Holland, 1993; Riolo, Cohen, and Axelrod, 2001), since resources are depleted when shared, which is not the case when knowledge is shared.

The inspiration for our experimental domain model came from the work of Nemeth and Takacs (Németh and Takács, Németh and Takács). In their model agents can teach or learn, or can do both. In their work sharing is based on proximity. They have demonstrated how altruistic teaching led to group formation and settlement. Agents shared their knowledge with their neighbours in their locality. As a result, the accomplishment of altruistic teaching led to an emergent phenomenon of settlement of agents and the accumulation of knowledge.

To model social behaviour in the artificial society, we used an interactive mechanism called the knowledge-sharing game. The operational details of the game are described in Section 4.3.1. It employs a social interaction model, where the sharing of knowledge is most beneficial for the group. Non-sharing is the selfish option, which benefits the individual but not the society as whole. Sharing benefits the society by spreading the knowledge, but it costs the donor who shares but not the recipient who gains the benefit. In this work "sharing the knowledge with other peers who lack knowledge at a cost to itself" is referred to as 'altruism'. The donation (sharing) costs the donor, and the donor gets nothing back as a reward/benefit. Donations reduce the score (wealth) of the donor, which can lead to the decrement of its survival and reproduction chances. The parameters of the experiment are Knowledge (K), Sharing (S), Wealth (W) and Tag (T).

- Knowledge (K bit) could be 0 or 1. If K=1, the agent possesses the knowledge, otherwise it does not.

- Sharing (S bit) could be 0 or 1. If S=1, the agent is willing to share, otherwise it does not.

- Wealth (W) could be 1.0 or below. When the agent initially possesses the knowledge, it has its Wealth set to 1.0. But each time it shares the knowledge, it loses 0.1from its

wealth.

- Tag (T) is a string of binary bits. Agents having the same tag belong to the same group.

In the start of the game, only 20% (randomly assigned) of the population possess the knowledge (K=1), hence they have a wealth score of 1.0 for possessing knowledge, and 50% (randomly assigned) of the population has the tendency to share (S=1). This leaves the agent population with four different types of players.

- **Type K+S+:** agents with knowledge who do share (K=1, S=1)
- **Type K-S+:** agents without knowledge who do share (K=0, S=1)
- **Type K+S-:** agents with knowledge who do not share (K=1, S=0)
- **Type K-S-:** agents without knowledge who do not share (K=0, S=0)

This is the general setup and the initial composition of players in all the experiments unless otherwise articulated. We chose these parameter values since we wanted to keep the number of sharers and non-sharers equal and start with a small number of knowledge bearers (1/5th of the population (20%) and see how this composition (four types) changes over time.

4.3.1 Tagless model

In this game, players are randomly paired, and sharing may or may not occur between them. Sharing happens only when one player of the pair (player1) has the knowledge and the tendency to share (K+S+) and the paired player (player 2) is without knowledge. The player who acquires the knowledge gains a wealth score of 1.0. During reproduction, the player with high wealth gets to reproduce and the player with low wealth dies. Sharing the knowledge does cost the donor (0.1) in terms of its wealth. The receiver gets only the wealth benefit (1.0) with no corresponding cost. 1.0 is the maximum value of wealth that a player can have at any time in this game. Thus if a player receives knowledge, its wealth value can never surpass 1.0, but each time it shares, it loses 0.1 from its wealth score (since it spends some time and energy to share). From the individual agent's perspective, it is better not to

share, so that it can keep its score high and increase its survival chances. But for the society's welfare overall, it is good to share.

The game is played with 100 players over a duration of 1000 iterations. In each iteration, every player gets to play the game once as a donor (player1) and once as a receiver (player2). The players with higher fitness have more chances for being selected for reproduction and they outperform others. This process derives logically from the concepts of survival of the fittest. In our approach reproduction does not apply for all individuals (the whole population) at the same time. Often in evolutionary models, the whole population enters the reproduction phase at the same time, but in nature it does not work that way.

In natural systems, reproduction takes place gradually in the population. In our approach 10% of the population is selected and paired for reproduction in every iteration. In each pair the stronger reproduces, and the weaker dies. If both are of same wealth, one is randomly selected. Thus 5% of the population reproduces (has one offspring), and 5% dies. In this manner, the population maintains a steady state with a value fixed at 100. The offspring agent is a copy of the parent, having the same behaviour of the parent (sharing bit, S), but not the knowledge (K bit). All young ones are born without knowledge and with a wealth of 0. The new agents acquire knowledge when they interact with other agents in the population that (a) have knowledge and (b) the tendency to share their knowledge with others. The overall process is outlined in Algorithm 4.1.

4.3.2 Results for the Tagless model experiment

Figure 4.1 shows the average of 30 complete runs of this experiment. It depicts the overall knowledge (not the wealth) of the population (represented by the K line) and the sharing behaviour (the S line). The experiment starts with 20% of the agents having knowledge (K line) and 50% of them having the sharing tendency (S line). When 50% of the population comprises sharers, the population gains more knowledge. After a number of generations (around 30-40 iterations), almost 90% of the population have acquired the knowledge. The sharing tendency starts going down as the sharers die out, because of their low wealth score due to the cost of donation (0.1). The selfish non-sharers take over, and the population drifts towards non-sharing, since the non-sharers retain the maximum score. After several

Algorithm 4.1: Pseudocode for the Tagless model

```
1  foreach generation do
2      foreach player do
3          Play with a random player;
4          Interact;
5          Share based on the behaviour (S);
6          Collect payoff;
7      end
8      Select 10% of the population;
9      Pair them for comparison of wealth (payoff);
10     foreach pair do
11         Weaker one dies;
12         Stronger one reproduces;
13         Newborn inherits the behaviour (S);
14     end
15 end
```

generations (100 iterations), the population has few with knowledge and very few with the tendency to share. Because most of the population now has the non-sharing behaviour, there is less sharing and hence a decline in knowledge (remember that offspring are born without knowledge). The S line then tends towards 0. When the S line is 0, there is no sharing at all, and the 5% newborn agents that appear after each iteration cannot obtain any knowledge. So in the end almost the entire population has become ignorant and selfish.

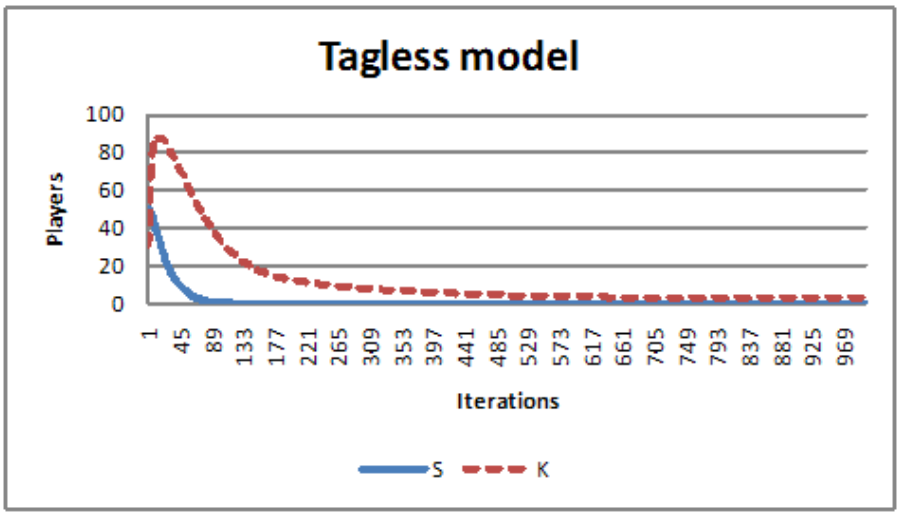

Figure 4.1: The knowledge and sharing level in Tagless model. The knowledge of the population represented by K line and the sharing behaviour by the S line.

4.3.3 Tag model

To improve upon the scenario described in the Tagless model, we introduced a new modeling approach, the Tag model. Tagging has been shown to achieve cooperation in animal societies (Axelrod *et al.*, 2004) and also in artificial agent societies (Hales, 2002; Riolo, 1997). This model is aligned with the idea that in general, most of us do not share information with just anyone, but only with those with whom we feel comfortable.

We again used the basic knowledge-sharing scenario described in Section 4.3.1, except that there is no sharing bit assigned, and instead the players have group tags. The decision to share is based on tag matching. If the tags match, sharing takes place, otherwise it does not. The sharing agent's score decreases by 0.1, every time it shares, and so sharers are more

likely to die than non-sharers. For our tag experiment, we use a string of 3 binary bits as tags (000, 001 etc.). Thus there are 8 different tag groups. Every player is randomly assigned a tag.

Algorithm 4.2: Pseudocode for the Tag model

```
1  foreach generation do
2      foreach player do
3          Play with a random player;
4          if tags match then
5              Interact;
6              Share;
7              Collect payoff;
8          end
9      end
10     Select 10% of the population;
11     Pair them for comparison of wealth (payoff);
12     foreach pair do
13         Weaker one dies;
14         Stronger one reproduces;
15         Newborn inherits the tag (T);
16     end
17 end
```

When two players interact, player1 (if it has knowledge) always shares its knowledge with player2 if they both have the same tag. Players are altruistic towards other players who are like them (based on their tag). At the end of each iteration 10% of the population is picked randomly, paired and compared by score. In each pair the high scorer gets the chance to reproduce, and the low scorer dies, so that 5% of the population reproduces in every generation. In natural evolutionary systems, mutations randomly occur, because of natural errors (which are very infrequent). In our experiments, the offspring agent gets the tag of the parent with a very low mutation probability and has no knowledge or wealth.

This tagging mechanism results in a population of 100 players grouped into 8 tag groups. Group members are always altruistic towards their own fellow group members. Since the sharing is based on tag matching, the population keeps maintaining the knowledge by sharing it with newcomers to their group. The newborns are born with their tags which they inherit from the parent. But they have no knowledge from birth. Whenever they get to interact with other players with the same tag that have the knowledge, they receive the knowledge. The overall process is outlined in Algorithm 4.2.

4.3.4 Result for the Tag model experiment

This setup results in almost 90-100% of the population eventually having knowledge (see Figure 4.2). Even after many generations (10000), the knowledge level is maintained. In Figure 4.2, the results from Tagless model and the Tag model are compared. The graph shows the average of 30 runs. The red K (without tag) line shows the knowledge level that was achieved in the Tagless model. The blue K (with tag) line shows the knowledge achieved in the Tag model, which makes use of tags. The tagging mechanism promotes altruistic behaviour in populations even where the reward for being selfish is more than that of being fair. At the start of the experiment, there were 8 tag groups with varying numbers of members. At the end of 1000 iterations, there was only 1 surviving group that had the entire population: all others had died.

At the outset only 20 players had knowledge. When they start sharing their knowledge within their group, the number of knowledge bearers in the population increased, iteration by iteration. By the end, one group ended up with all the knowledge. (Note that the maximum number of knowledge bearers in the population will be 95, because there are always 5 new agents born without knowledge.)

In this model, even if few players are available in the population, at least the parent of the newborn has the same tag. For the newborns the knowledge will continue to be shared in this setup, which does not occur in the previous setup (Tagless model) where all the sharers may die out.

This result is in accordance with the work reported in (Riolo *et al.*, 2001; Riolo, Cohen, and Axelrod, 2002). They have designed their model in such a way that every group is full of

donors who donate the resources, and there is no one who does not donate within the group. By that the agents of the same tag group always cooperated with each other. In other words, the behaviour is correlated with the tag. It was remarked by Roberts and Sherratt (Roberts and Sherratt, 2002) that the agents were forced to cooperate within their group.

We extend the investigation by employing a mechanism in our next experiment that does not correlate a tag with behaviour.

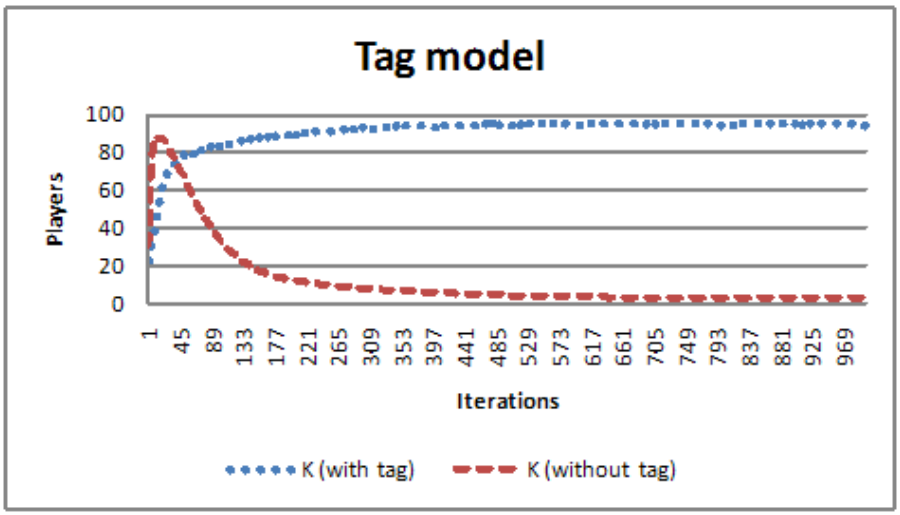

Figure 4.2: Comparison of knowledge in Tag model and Tagless model.

4.3.5 Tag-and-trait model

We believe that having both sharers and non-sharers in a group would be a plausibly realistic approach. In order to test whether the tag system really works for the given domain, the strategy/behaviour should not be correlated with the tag. In our next mechanism, the behaviour of the agent is independent of the tag, which means that even though the tags may match, the agents do not have to cooperate/share. Behaviour is based on the sharing bit (tag and behaviour are not correlated). This is analogous to an agent not sharing a skill with someone in the same group, because it is inherently selfish.

The Tag-and-trait model is a combination of Tagless and Tag models. In the Tagless model, the decision to share is based on the S bit, and interaction is allowed with any random player in the population unconditionally. In the Tag model, the decision to share is based on tag matching, and the interaction is restricted within the group. We experimented with

combining aspects of Tagless and Tag models. In the Tag-and-trait model, the decision to share is based on the S bit, and the interaction is local, based on tags.

We performed the experiment of the Tag-and-trait model in essentially the same fashion as the Tag model, with a few differences. Recall that the sharing decision in the Tag model is just based on tag matching: if the tags matched, then the agents shared. In the Tag-and-trait model, once the tags match, then whether sharing takes place is based on the sharing bit (S). Also, in the Tag model during reproduction, the offspring agents only inherit the tag of their parent. In the Tag-and-trait model, the offspring agent also inherits the sharing tendency (S bit), as well as the parent's tag.

Algorithm 4.3: Pseudocode for the Tag-and-trait model

```
1  foreach generation do
2      foreach player do
3          Play with a random player;
4          if tags match then
5              Interact;
6              Share based on the behaviour (S);
7              Collect payoff;
8          end
9      end
10     Select 10% of the population;
11     Pair them for comparison of wealth (payoff);
12     foreach pair do
13         Weaker one dies;
14         Stronger one reproduces;
15         Newborn inherits the tag (T), behaviour (S);
16     end
17 end
```

The overall process of the Tag-and-trait model is outlined in Algorithm 4.3.

4.3.6 Results for the Tag-and-trait model experiment

There were dramatically different behaviours observed for different stochastic experimental runs of the simulation. One result, call it "All-sharing", shows that 100% sharing can be achieved using tags (Figure 4.3, see the S line), while another result, "No-sharing", shows 0% sharing (Figure 4.4, see the S line). In Figure 4.3, it can be observed that the sharers ultimately increased in numbers for that run, so there was more knowledge-sharing in the population. Almost 95% of the population gained knowledge. Another result for the same experiment is shown in Figure 4.4. From Figure 4.4, it can be observed that the number of sharers eventually declined towards 0, and due to the lack of sharing, the knowledge level decreased.

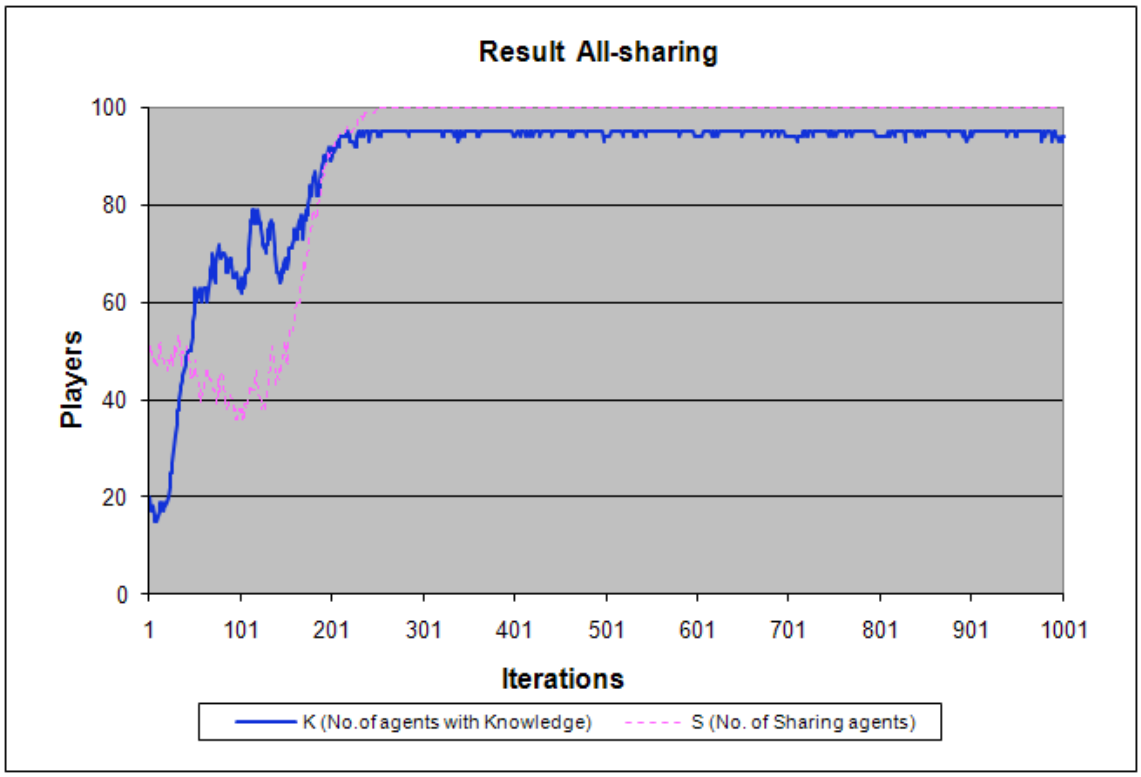

Figure 4.3: The K line shows the knowledge and the S line shows the sharing (All-sharing).

The reason lies in the formation of groups with K and S. The rate at which the number of sharers die out in the population should be less than the number of sharers who are born in the groups. If so, the sharing is supported and produces the "All-sharing" result. If the

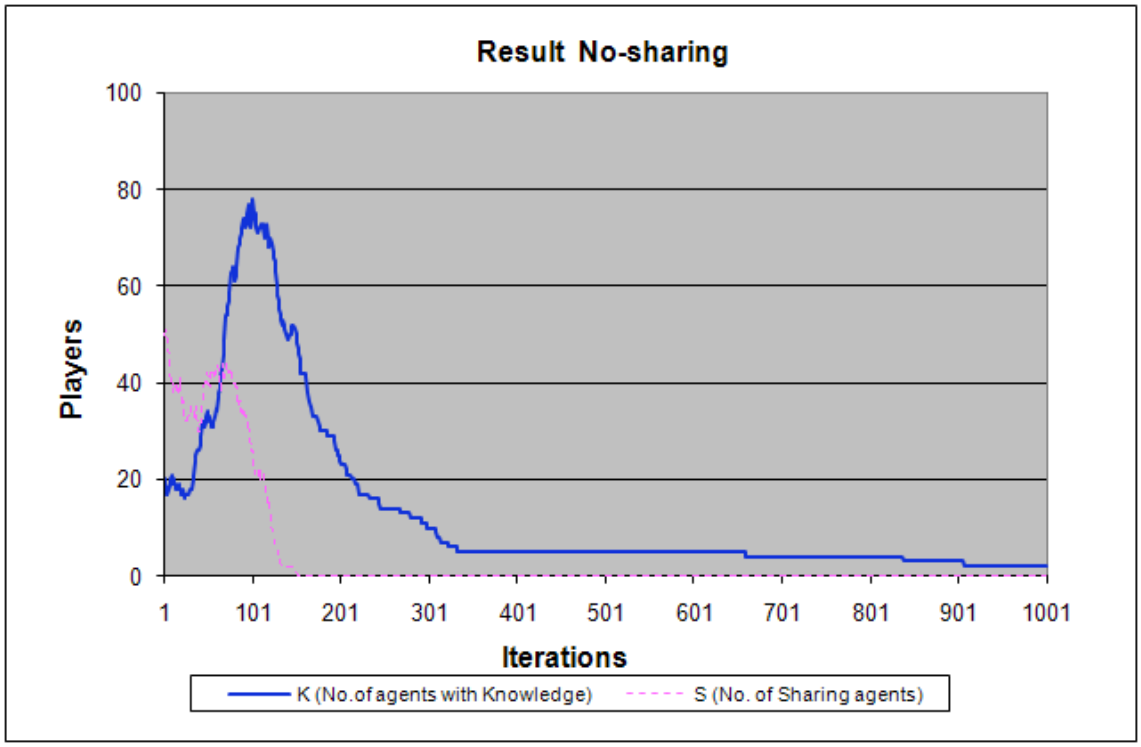

Figure 4.4: The K line shows the knowledge and the S line shows the sharing (No-sharing).

number of sharers that die are more than the number of newborn sharers, the "No-sharing" result is obtained. We note that behaviour of a qualitatively similar nature was observed by McDonald and Sen (2005) in the Prisoner's Dilemma game, who explained that for cooperation to evolve, the number of groups invaded by defectors must be less than the number of new cooperative groups formed.

In the process of investigating the two different behaviours of our results, we observed that there was a key structural indicator associated with those groups that eventually dominate the society when the society has complete sharing and knowledge.

This indicator reflects the crucial condition that identifies when the group has rid itself of all its non-sharers. The group with this condition must also have at least one member of type K+S+. When non sharers (type K+S- and type K-S-) gain the knowledge, their wealth becomes 1, and they maintain this wealth (since they never share). That will make it unfavorable for type K+S+ players to survive, since they gradually lose their wealth by sharing. As a result, type K+S+ players eventually die out, and type K-S- players increase.

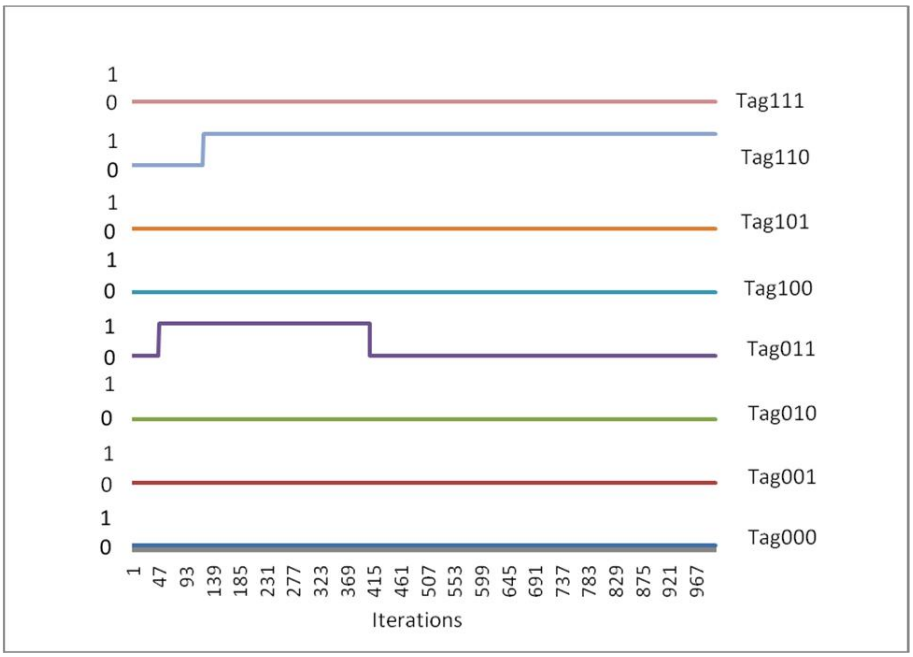

Figure 4.5: Groups satisfying the winning condition.

When a group loses all its type K+S+ players, there is no way for the players in that group to obtain knowledge. The group will end up producing more knowledge-less non-sharers (type K-S-) whose wealth is 0. That will eventually lead to the group's extinction.

As long as there are type K+S+ players in a group, they serve as sources for the entire group to gain knowledge. When type K-S+ players gain the knowledge from type K+S+, they also become type K+S+. This group will survive by producing type K-S+ sharers (no knowledge) who will change as type K+S+ and share more.

Thus a group which gets rid of type K+S- and type K-S- and is composed of only type K+S+ and type K-S+ will survive. Hereafter we refer to this condition as the "winning condition". This winning condition is the necessary condition for a group to win and to achieve the all-sharing result. This winning condition, which is necessary for any group to survive in the long term, can be expressed as follows.

$$\mathrm{COUNT}(\mathrm{K}+\mathrm{S}+)>0 \quad \mathrm{AND} \quad \mathrm{COUNT}(\mathrm{S}-)=0 \tag{4.1}$$

The predicate 'COUNT' should be read as the number of players with the given condition.

The number of type K-S+ agents does not matter, because if there are some type K-S+

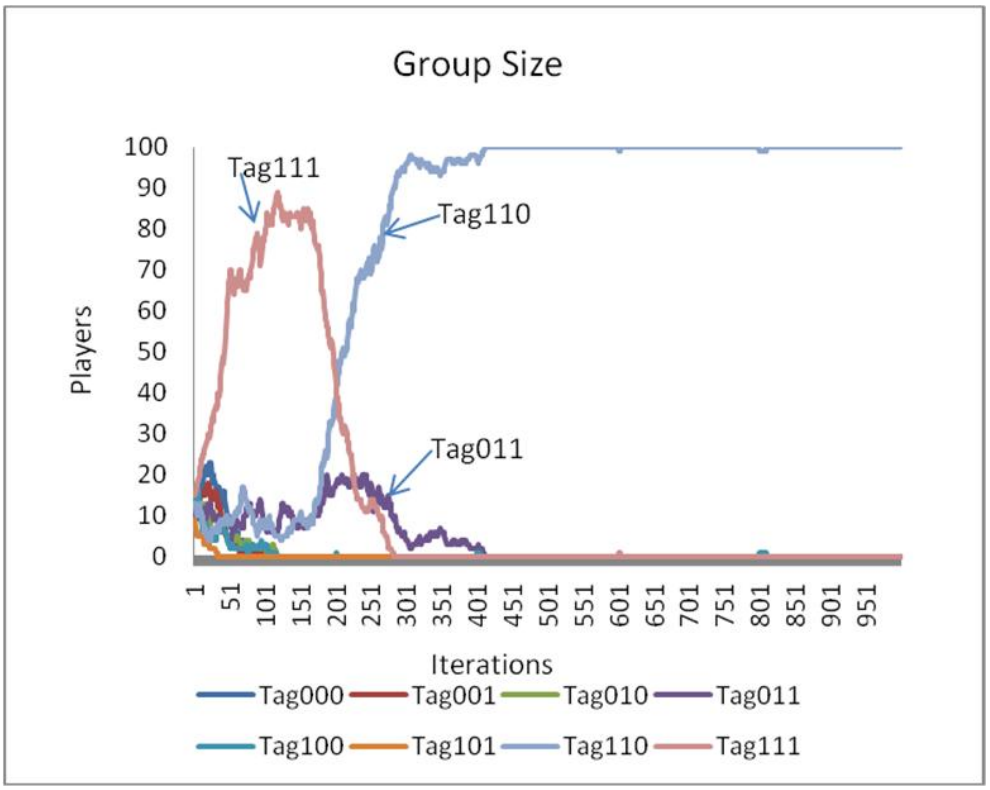

Figure 4.6: Size of tag groups (with 3 bit tags).

agents without any type K+S+ agent in the group, the group is still doomed, since these K-S+ agents cannot receive knowledge.

Results from a sample run are presented in Figures 4.5 and 4.6. Among the 8 groups there could be more than 1 group which could meet the winning condition. Figure 4.5 shows 8 lines corresponding to 8 tag groups. On the y-axis each line has two values, 0 and 1. When a tag group meets the winning condition, the line goes from 0 to 1. Among 8 groups, 2 groups met the winning condition at some point. They are tag groups 011 and 110: tag group 011 met the winning condition in iteration 47 and it sustained the condition for some iterations. At iteration 408, it failed the condition so the line dropped to 0.

But tag group 110 met the condition in the 123rd iteration and became the winner, since it sustained the condition. Even though these two groups had only sharers, when group 011 lost their type K+S+ players, the group could not succeed in the long run.

From Figure 4.6, it can be observed that there is only one winner out of all the groups. The entire population is of one group (group 110), because of the nature of the interaction mechanism to replicate the fitter member and extinguish the weaker member. The group

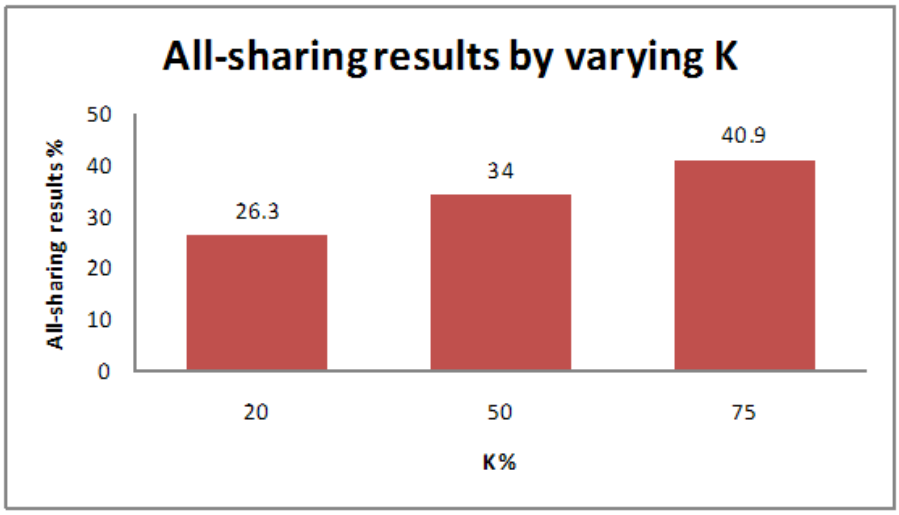

Figure 4.7: Keeping S constant and increasing K.

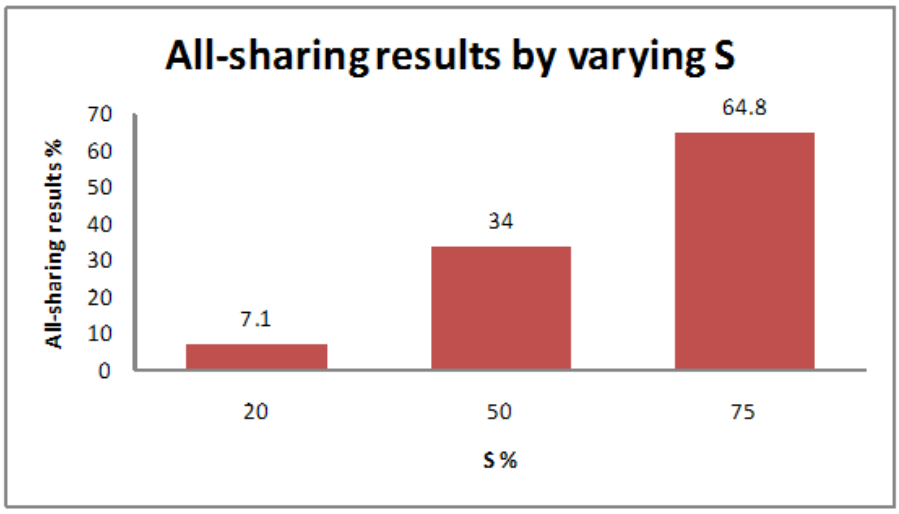

Figure 4.8: Keeping K constant and increasing S.

111, which had almost 80-90% of the population at one point, became extinct later, because it had non-sharers and also lost its type K+S+ players. The group 110, which was initially a small group with the winning condition, eventually took over.

The "All-sharing" result was obtained when there was at least one group that met the winning condition and sustained it. Otherwise we obtained results like "No-sharing", where none of the groups satisfied the condition, and they all ended up with non-sharing, knowledge-less entities. If the agents are either of type K+S+ and type K-S+ at the end of the iterations, the "All-sharing" result is obtained. If they are only of type K+S- and type K-S-, the "No-sharing" result is obtained.

We have experimented with different percentages of agents with K and maintaining the percentage of those with S constant at 50%. We observed that using initial K populations of 20%, 50% and 75% the likelihood of obtaining the "All-sharing" result increased as we incremented the value of K. It is shown in Figure 4.7.

We have experimented with different percentage of agents with S and maintaining the percentage of those with knowledge, K constant at 50%. We observed that using initial S populations of 20%, 50% and 75% the likelihood of obtaining the "All-sharing" result increased as we incremented the value of S. It is shown in Figure 4.8.

Note that these are values averaged over 1000 runs. The percentage of getting "All-sharing" results is an approximation calculated by averaging these results. For this setup, we received 22.3% of "All-sharing" results (1000 runs). It can be observed from the two figures that an increase in either S or K produces better "All-sharing" results. And also from these two figures (Figures 4.7 and 4.8) we can observe that the "All-sharing" result is more sensitive to varying S than K.

4.3.7 Conversion model

We developed another mechanism which works in the same fashion as the Tag-and-trait model, but differs in the reproduction phase. It employs a conversion mechanism instead. The behaviour (S bit) is not inherited by the offspring.

The conversion process at the end of each iteration works in the following way. 10% of the population is picked randomly, paired and compared by wealth score. With every pair

the high scorer in wealth gets the chance to convert the low scorer to its tag group. If both players are of the same wealth in a pair, one of them gets to convert by random selection.

The winning agent converts the losing agent by adding the agent to its group (i.e the low scorer joins the tag group of the high scorer). The converted agent does not have the knowledge (its K bit value is 0) when joining the new group. The converted agent retains its original sharing behaviour (S bit).

The new agents wind up acquiring knowledge whenever they interact with other agents in the population that have knowledge and the same tag and also have the tendency to share their knowledge. By this process, after each iteration 5% of the population gets converted. The population thus has a steady state population with a value fixed of 100. The overall process of the conversion model is described in Algorithm 4.4.

Algorithm 4.4: Pseudocode for the Conversion model

```
1  foreach generation do
2      foreach player do
3          Play with a random player;
4          if tags match then
5              Interact;
6              Share based on the behaviour (S);
7              Collect payoff;
8          end
9      end
10     Select 10% of the population;
11     Pair them for comparison of wealth (payoff);
12     foreach pair do
13         Stronger converts Weaker ;
14         Converted one gets Stronger's tag (T);
15         Converted one retains its behaviour (S);
16     end
17 end
```

4.3.8 Result for the Conversion model experiment

In our results, when we say that 'knowledge is sustained' we mean that more than 80% of the agent population has the knowledge and that knowledge is constantly being transferred by conversions, to newcomers arriving each iteration.

The final population (at the end of the run) belongs almost entirely to a single tag group which is the strongest (has the greatest wealth), and all other agents previously belonging to other groups have been converted to the winning group.

Our results for this configuration (Figures 4.9 and 4.10), show that the knowledge could be sustained in the population even with the presence of selfish agents. This result contrasts significantly with earlier experiments with the Tag-and-trait model where the agent population was able to sustain knowledge only when one of the groups attained the winning condition.

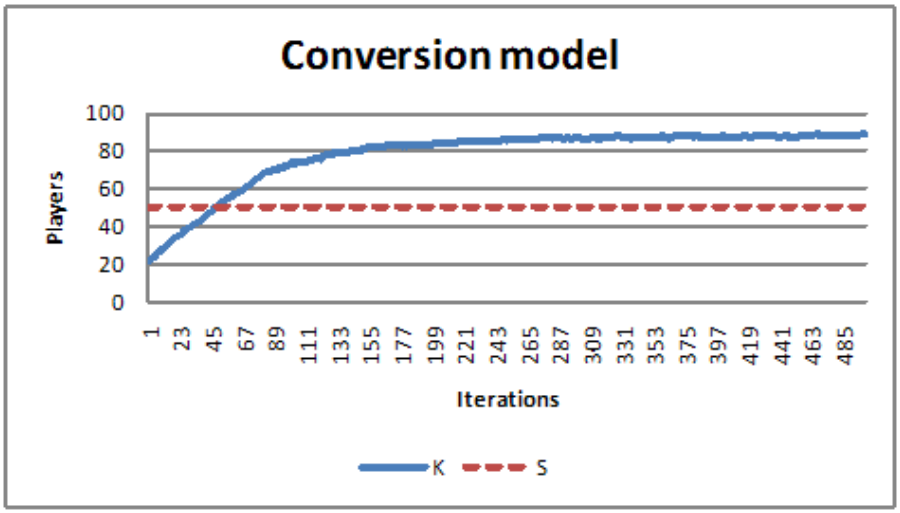

Figure 4.9: The K line shows the knowledge and the S line shows the sharing.

Figure 4.9 shows the average of 30 runs. It shows that the number of sharers is always 50 (see the S line) and the knowledge is sustained (see the K line) in the society. Figure 4.10 shows a sample run which depicts the distribution of the 4 types of players at the end of the run.

The distinguishing feature of the conversion model which brings about distinctly different results when compared with the Tag-and-trait model is in the agent interaction mecha-

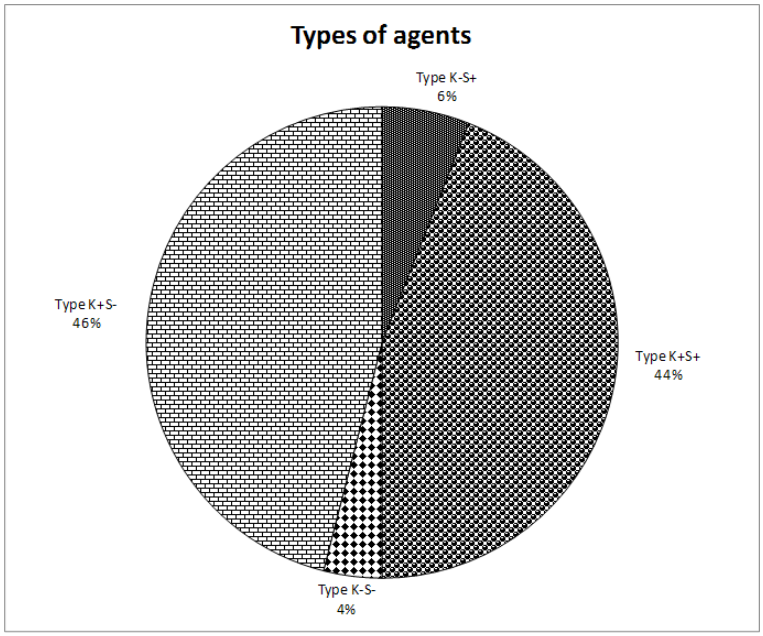

Figure 4.10: Four types of agents at the end of the iterations.

nism. In this configuration converted agents retain their behaviour, but just the tag changes. It means the group tag changes to that of the stronger player, instead of inheriting the behaviour of the dominant player with its tag as in the previous Tag-and-trait model.

In order to elucidate these results, we examine the agent interaction operator in detail. Note that agents possessing the (K+S-) attributes are likely to have a high wealth since they never share, and thereby never incur the 0.1 cost of sharing.

In this conversion model, the newcomer (converted agent) retains its own behaviour and inherits the tag of the dominant mate. If the newcomer is a non-sharer, it comes to the new group as a non-sharer (without knowledge). If it is a sharer, it comes as a sharer (without knowledge). Both cases are advantages in the current mechanism. They are listed below.

- If a sharer comes to the group and receives knowledge from an existing sharer in this group, it starts sharing within the group as well (K-S+ becomes K+S+).

- If a non-sharer comes to the group and receives knowledge from an existing sharer in this group, its wealth becomes 1 (K-S- becomes K+S-). Since it never shares, its wealth is high, and it converts other players and brings new members to this group.

The number of sharers and non-sharers remained the same in this situation, but the

knowledge-sharing was still made possible and was sustained and passed on for future generations.

4.3.9 Experiment on individual cost bearing versus group cost sharing

We have also performed experiments comparing individual cost bearing versus group cost sharing. In the setup explained so far, the sharer always incurs a cost for its donation, which reduces its wealth. So we have examined an alternative mechanism whereby the cost is not incurred individually by the sharer alone, but distributed across the entire group and shared by everyone. Everyone's wealth is reduced by *cost/n* where *n* is the number of members. We experimented with both of these cost-bearing mechanisms (Individual vs. Group cost-sharing) to compare the performances[2].

We tested 2 types of cost-bearing with 2 sets in a population. Each set has 4 groups. They play the knowledge-sharing game within their group. In the previous experiments, the game is played with 8 tag groups and with individual cost sharing. In the current experiment out of 8 groups, 4 play with individual cost sharing and 4 play with group cost sharing. We wanted to see which group type performed better. Except for these two differing cost-bearing mechanisms, with one group "Set 1" subject to individual cost-bearing and "Set 2" subject to group cost-bearing, everything else in the experiment was the same as that described in Section 4.3.7. Algorithm 4.5 shows the pseudocode schematically comparing the two cost sharing approaches.

Algorithm 4.5: Pseudocode for the process of cost sharing

```
1 if group belongs to Set 1 then
2 |   Sharer bears the cost;
3 |   Receiver receives the benefit;
4 end
5 if group belongs to Set 2 then
6 |   Group members bear the cost;
7 |   Receiver receives the benefit;
8 end
```

[2]Performance is measured by wealth.

4.3.10 Result for the individual versus group cost sharing experiment

Our results in this competitive comparison indicated that groups from Set 1 (individual cost-bearing) won most of the times. This is because when the cost is borne just by a single sharer during a game, only one agent's wealth is reduced; hence its survival chance is low. It could be converted to another group if it gets picked against a wealthier player.

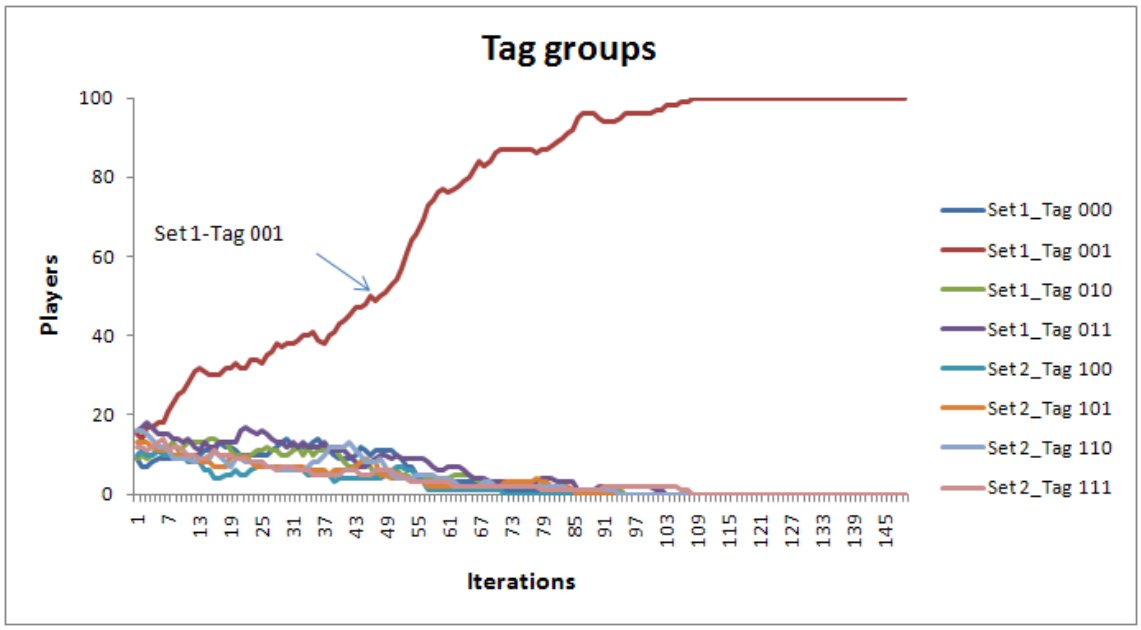

Figure 4.11: Tag groups from 2 sets.

In the contrasting case where the cost is shared by everyone in the group, everyone loses a little of their individual wealth, every time there is a sharing in the group. It makes the whole group weaker and more of the members are prone to be converted when playing against wealthier players.

In this experiment we observed that the individual cost-bearing was more effective in terms of knowledge sharing. This was due to the fact that only a few agents lose their wealth by bearing the cost and the remaining group members who are not sharing the cost are relatively stronger, so they can convert weaker players from other groups. Thus the winner is always a group from Set 1. A sample result is shown in Figure 4.11. Thus out of 8 groups from 2 sets, the tag group 001 from set 1 became the winner which ended up with all the players getting converted to its group and sustained the knowledge in the population.

Group cost sharing under these particular conditions weakened the groups. Individual

cost-bearing did not. Thus in this model it was good for the society to have some people who sacrifice for the well-being of the group, instead of everyone in the society contributing to cost sharing.

4.4 Discussion

Mutation is usually a rare, one-off event that happens in the evolutionary cycle of an organism if the environment triggers it. Mutation may sometimes lead to behavioural trait more suited to the survivability of the organism in its environment. Mutation may have either positive or negative effects depending on external factors as well. So we have used a low probability of mutation in our experiments. In this work mutation did not play any significant role, because it always disadvantaged the mutant. During interaction, the knowledge-sharing happens if the tags match. If the mutant is the only one with a particular tag in the society, then there is no matching partner for him to gain knowledge. If no knowledge then no wealth. Hence the mutant cannot survive without wealth and will be replaced by stronger ones. Even if there is another matching tag partner for the mutant, the mutant will get knowledge only if the matching partner has knowledge and shares it. We note that this will not affect the overall behaviour of the system. The winner in this system will still be the group which is stronger than all others.

In this work we used only tag mutation. The system behaviour could vary if we introduce mutation for the sharing bit, but in our work, behaviour (S bit) is just a single bit entity. We will consider mutation for behaviour in future work. Further in this domain, we intend to investigate the linkage between tag length and the population size. From previous literature (McDonald and Sen, 2005) we know that for cooperation to evolve in larger populations longer tag bits are needed. It will be interesting to find what tag length would be enough for a given population to converge towards altruism in a particular system model (domain). In this work the knowledge is modeled as a single entity. We are also interested in experimenting with an agent population having multiple types of knowledge. It would be interesting to see in this context how effectively different types of knowledge or skills can be shared in an agent society.

4.5 Summary

The main differences in these four different experiments and their results in terms of sharing the knowledge are shown in Table 4.1. The experiment on the cost-sharing model is not listed in Table 4.1, because it had the same setup as the conversion model. In Table 4.1, *tag inheritance* indicates whether tags are present/inherited in the system. *Sharing bit present* indicates whether the S bit is present. *Behaviour inheritance* indicates whether the S bit is inherited. *Knowledge preserved* indicates whether the knowledge is being shared and passed to the next generations.

Experiment	Tag inheritance	Sharing bit present	Behaviour inheritance	Knowledge preserved
Tagless model	No	Yes	Yes	No
Tag model	Yes	No	Yes	Yes
Tag-and-trait model	Yes	Yes	Yes	Yes & No
Conversion model	Yes	Yes	No	Yes

Table 4.1: The main differences in the four sharing systems.

In this chapter we have shown how altruism based on tags can be used in different systems of independent agents. In the context of the knowledge-sharing game, we have shown that tagging can help sustain the knowledge possessed within the society.

Through our experiments, we have shown that sharing does not happen without tags in the Tagless model. We have also shown in Tag model, that the use of tags can do better in improving sharing than when the sharing decision is just based on tag matching, when only group tags are inherited from the parents. It could still fail to preserve knowledge if the sharing includes the tendency (trait), the sharing behaviour (S bit) when inheritance involves both the group tag and the sharing behaviour (S bit) as demonstrated in Tag-and-trait model.

In the Conversion model, we have presented our results about how a society could share and sustain knowledge even in the presence of selfish agents that are present in equal proportions. We have also shown that bearing the cost individually can be a better option than bearing the cost across the whole group under certain experimental conditions.

Overall, the focus of this chapter was to demonstrate how knowledge is shared among

individual agents in a system controlled environment (*centralised* society). Though central monitoring and controlling can be beneficial in facilitating social control, these systems are limited in scalability and flexibility. Thus the focus of the next chapter is to investigate a *semi-centralised* approach for facilitating social control.

Chapter 5

Self-organisation in semi-centralised systems

5.1 Introduction

When autonomous agent communication is employed in artificial agent societies, the communicating agents need to share a common ontology with respect to the terms and relations that are contained in the bodies of messages. Approaches are needed to ensure that individual agents do not act in ways that negatively affect the system goals. A structure observed in natural ecosystems that would evidently be useful for addressing these issues is that of a society or group of agents that interact according to an agreed-upon set of policies or norms of the group (Aldewereld, Vázquez-Salceda, Dignum, and Meyer, 2006). Research in the area of open autonomous agent norms and institutions has focused on ontological considerations with respect to institutional commitments (Singh, 1999; Colombetti *et al.*, 2002; Oliveira, Purvis, Cranefield, and Nowostawski, 2004) and sanctions that may be imposed when commitments are not upheld. Sierra and his team (Arcos *et al.*, 2005), working with more tightly controlled systems, have implemented agent-based institutional systems with *centrally administered mechanisms* that maintain strict control of individual agent interactions within the institutions. In this chapter, we describe our own *semi-centralised* approach which differs from the above.

In our approach, we are concerned not only with agents that break institutional laws, but

also those that merely fail to contribute to the common good. We employ simple tags because of the complexity and intricacies associated with using detailed ontologies for specifying institutional goals and rules. We have developed mechanisms inspired by social concepts such as voting and referral and also use monitoring agents (semi-centralised control) locally at the group level. The objective of this work is to avoid cooperative sharers being exploited by uncooperative freeriders and improve societal performance by restricting freeriding.

By nature's design different kinds of people exist in a society. Every society has cooperative and uncooperative members. In the real world it is not possible to get rid of all uncooperative members from the society. However, by imposing strict control it is possible to exclude people with certain behaviour (uncooperative) from the society, but this is not what we want to achieve here, because measures of strict control are not scalable for larger societies. Our goal is to restrict the performance of uncooperative members and prevent the cooperative members from being exploited rather than excluding exploiters out of the society. The uncooperative members not only take advantage of cooperative members, but also cause damage to the common good. Special mechanisms need to be designed and deployed to control the behaviour of such groups in electronic societies. To avoid exploitation, we can separate good and bad within the society without excluding them.

This work uses simple tags for the self-organisation of cooperative and uncooperative groups by employing monitoring agents for each group. It also includes the voting mechanism on top of the referral mechanism. Additionally, we show that the resource restriction for uncooperative groups improves the societal benefit. In this chapter, we investigate these mechanisms in reducing the performance (wealth) of uncooperative members and increasing the overall societal benefit.

This chapter is organised as follows. In Section 5.2 we present our experiments which make use of monitoring agents, tags, voting, referral and resource restriction mechanisms. In Section 5.3, we summarise this chapter.

5.2 System design and experiments

In our system, an artificial agent society is partitioned into groups, with a special monitor agent for each group. Each monitor agent computes an overall performance measure of the

group that it maintains and supervises group access and membership. The context of agent interaction is the Prisoners Dilemma game which is explained in Section 2.3.1.1. The payoff values are shown in Table 5.1.

Players 1 / 2	**Cooperate**	**Defect**
Cooperate	2,2	0,3
Defect	3,0	1,1

Table 5.1: Payoff values for Prisoner's Dilemma.

5.2.1 System 1 using tags

In the first set of experiments, an artificial agent simulation environment was set up with a society of 100 agents divided into 5 groups of 20 agents each. Individual game-playing agents were programmed to have an internal parameter which determined their tendency to cooperate or defect, *degree of cooperation, (dc)* that was randomly initialised to have a value between 0 and 1. At the outset of the game, each of the five groups was populated with a random collection of players having various tendencies to cooperate or defect.

In each group, each agent played a round of games (Prisoner's Dilemma) with other agents in its group, after which the group monitor would conduct a voting among the group members to determine which ones are the most cooperative and least cooperative members of the group. When a group member pairs off with a playing partner, it plays 5 games with that player, so each member plays 95 (19 other agents*5) games in a round. Players get scores for their played strategies as shown in the Table 5.1.

Based on the 5 games a player plays with other members, it can compute a cooperation score for each player it played with. But members only know about the games they themselves played, and they know nothing about how much other agents may have cooperated with one another. The *performance* of an individual agent is measured by its individual score and is denoted as p. At the end of each round, the voting is performed. Voting is the process of ranking group members based on their degree of cooperativeness (dc) with other members it played with for that round . The score of an individual member p is different from its own degree of cooperation denoted as dc. For instance, in the Prisoner's Dilemma game,

an uncooperative group member may defect its fellow members and achieve a high score p (as per Prisoner's Dilemma payoff matrix in Table 5.1), even though being considered to be least cooperative by its fellow group members (having a low value of dc).

An agent is considered to be cooperative if dc is grater than or equal to a cooperation threshold value ct. In our case we set ct=0.5. An agent was considered to be cooperative if it cooperated at least 50% of the time (which corresponded to having a degree of cooperation (dc =0.5). For instance if an agent has dc=0.4, it cooperates 4 times out of 10. Since its dc value is less than ct, it is considered to be uncooperative.

An agent's vote is based on its individual playing experiences with all the other agents in the society. Players vote for best and worst cooperators based on their experience. When all the votes based on individual member experiences are tallied by the group monitor at the end of the round, there will be a ranking of group members based on their degree of cooperativeness (dc) for that round. Group members vote for their best and worst cooperators. Thus after tallying the votes, the monitor will know its most cooperative (highest dc) and least cooperative (lowest dc) member for that round. The monitor uses this information to get rid of the least cooperative member and promote the most cooperative member to other groups. It can be assumed as getting a ticket from the group monitor to leave the group.

In addition, for each round, the five groups rank themselves in terms of their overall performance (op), which is the sum of the individual scores (p) of all of its members in the group. The performance score of a player p is the sum of all the scores it scored in that round.

To determine movement between groups, the procedure given below is followed:

- The highest ranked group in terms of op kicks out its least cooperative member.
- The 2nd, 3rd, and 4th groups in terms of op also kick out their least cooperative members, but also promote their most cooperative members for movement to a new group.
- The lowest ranked group in terms of op promotes its most cooperative member for movement to a new group.

There are, thus, eight agents that have been placed into a separated pool for moving to another group: four promoted from the 2nd, 3rd, 4th, and 5th ranked groups and four kicked

Algorithm 5.1: Pseudocode for System 1 (using tags)

```
1  begin
2      initialisation;
3      bootstrap agents in groups;
4      foreach round do
5          foreach group do
6              foreach agent do
7                  play five games with every other agent in the group;
8                  rank them based on their cooperativeness;
9              end
10             performance of the group is calculated;
11             voting is performed;
12             group monitor collects votes and ranks players;
13         end
14         Groups are sorted based on performance;
15         while elimination from group do
16             best cooperative player is taken out from the worst performing group and
               given blue tag;
17             worst cooperative player is taken out from the best performing group and
               given red tag;
18             worst cooperative and best cooperative players are taken from medium
               performing groups and given red tags and blue tags respectively ;
19         end
20         players who were out of their groups are in a temporary pool;
21         group monitor chooses the agents from the pool;
22         while selection from pool do
23             The high performing groups get blue tagged players;
24             The low performing groups get red tagged players;
25             The medium performing groups get blue tagged players and red tagged
               players;
26         end
27     end
28 end
```

out of the 1st, 2nd, 3rd, and 4th ranked groups. Now tagging is employed. The four promoted agents are given blue tags, signifying promotion, and the four kicked-out agents are given red tags, signifying demotion. This procedure is illustrated in Figure 5.1.

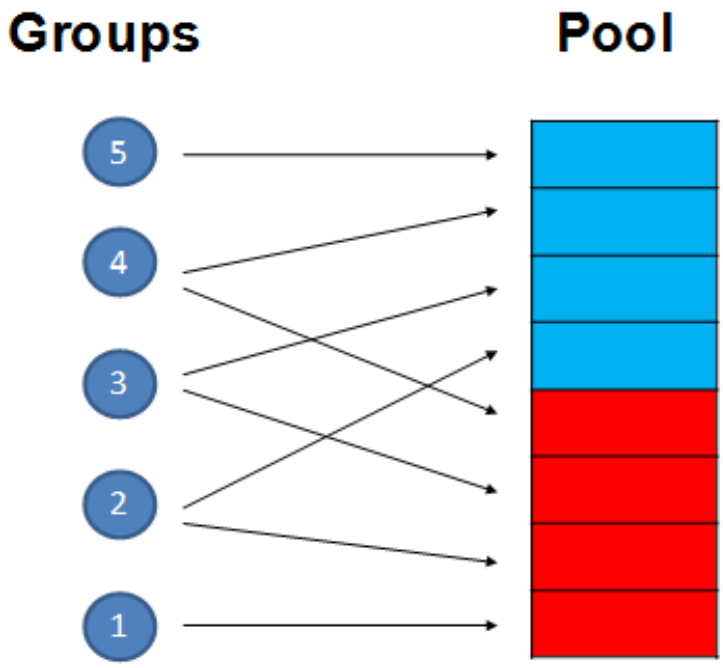

Figure 5.1: Elimination from group.

Commonly in tag-related works (Hales and Edmonds, 2003a,b; Hales, 2004a,b; Riolo, 1997; Riolo *et al.*, 2001) tags serve the purpose of showing the identity of the agent and specify which group the agent belongs to. Here our purpose of using tags is just to represent the status of an agent which is currently in the pool. The status could be high (promoted) represented by blue tags or low (demoted) represented by red tags.

The monitor agent chooses players with blue tags in preference to players with red tags without knowing the performance scores of the players in the pool. The monitor agent takes players with blue tags if they are still available when it comes to its turn to choose. The monitors of the groups pick the players from the pool starting from the highest ranked group.

- At this stage, the highest ranked group gets one agent among the pool members in order to replace the member that has been kicked out. (1st group gets 1 blue tagged agent).

- Then the 2nd, 3rd, and 4th ranked groups get two agents to replace the two agents that they have lost. (2nd group gets 2 blue tagged agents, 3rd group gets a blue tagged and a red tagged agent and the 4th group gets 2 red tagged agents).
- Finally, the lowest ranked group winds up with the remaining agent of the pool. (5th group gets a red tagged agent).

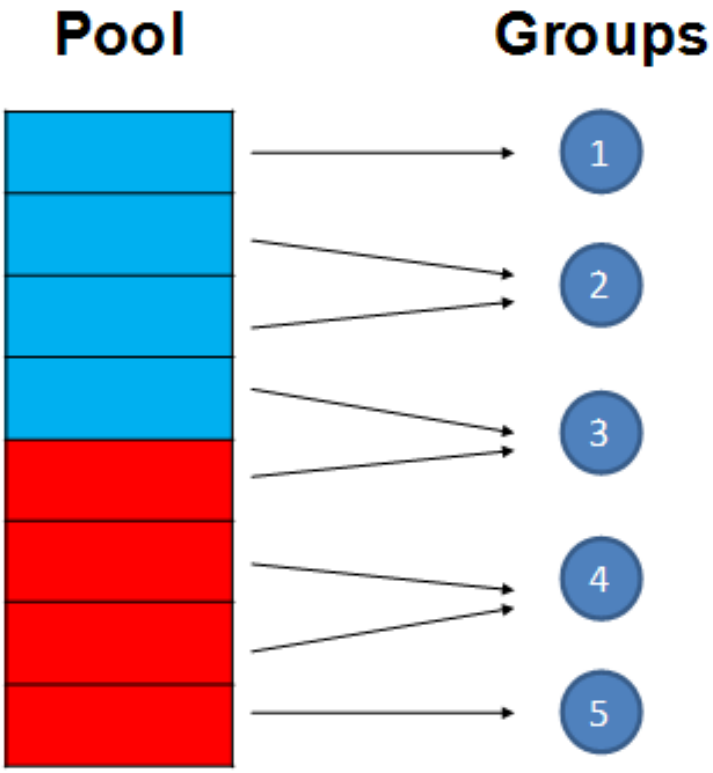

Figure 5.2: Selection from pool.

Thus groups always have the same number of members. The procedure for selection from the pool is illustrated in Figure 5.2. With the new groups now constituted, another round of play is commenced. The pseudocode is given in Algorithm 5.1.

The goal of the experiments is to see how well this mechanism established and maintained groups that separated and protected 'good-guy' cooperative agents from 'bad-guy' defecting agents.

From Figure 5.3, we can see the separation of groups in terms of number of cooperators (from iteration 10 onwards). Two groups with full of cooperators and two groups with almost

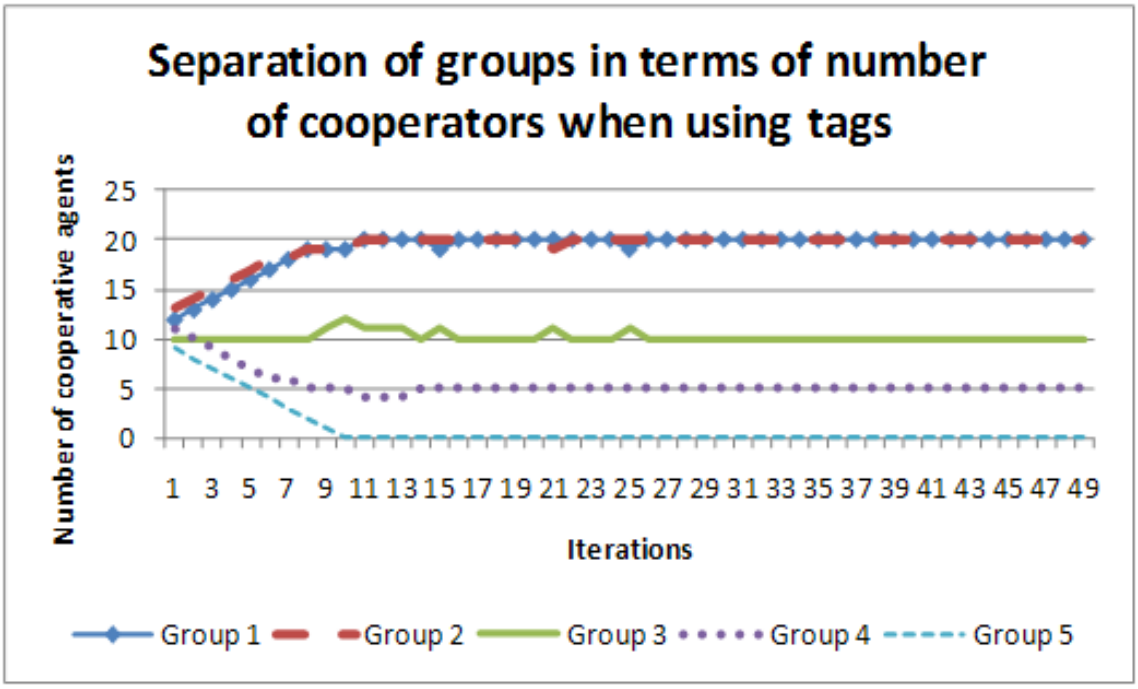

Figure 5.3: Prisoner's Dilemma with tagging.

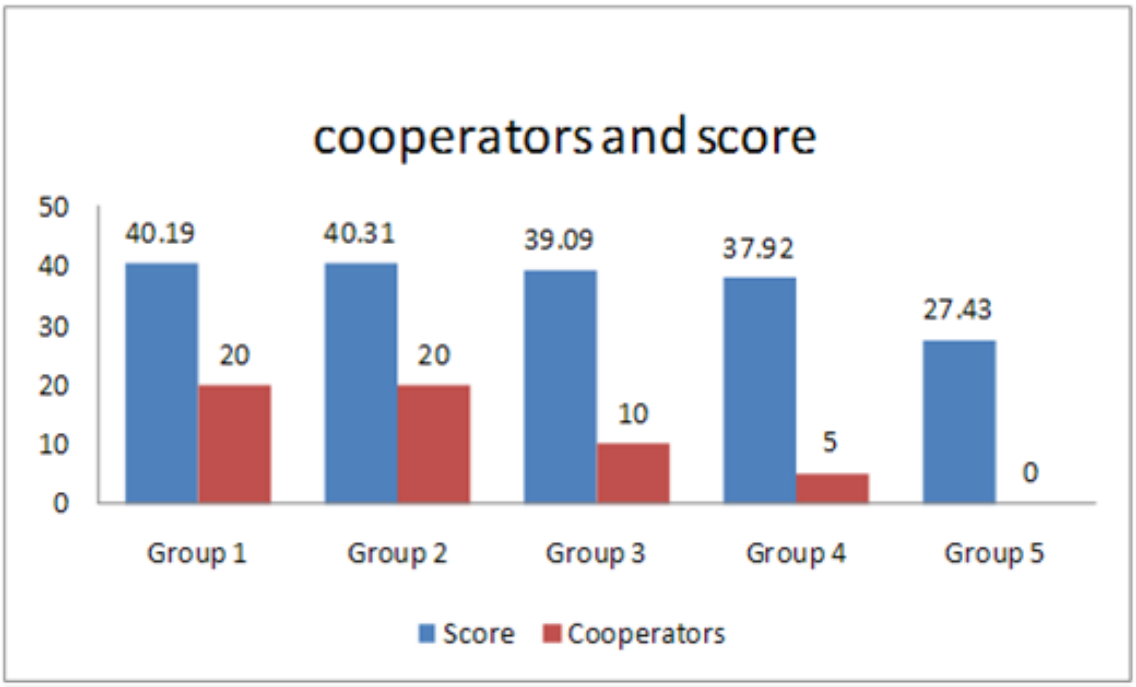

Figure 5.4: Cooperativeness and score of groups.

all defectors and a middle group with both cooperators and noncooperators. This graph shows the clear separation of these groups.

Figure 5.4 shows the groups in descending order (from left to right) based on the number of cooperators. This figure shows the association between the number of cooperators in the group and the average score of the group. The groups with more cooperators had higher scores. The group without any cooperators (Group 5) had the lowest score compared to all other groups (Groups 1 to 4). The differences in the average group scores between groups 1 to 4 are small. The difference in the average group score between groups 4 and 5 is high (a difference of 10.49).

A paired t-test was conducted to prove the separation of groups based on cooperativeness. The standard deviation of number of cooperators in groups at the start and end of each run was calculated for 30 runs. With 95% confidence level we can say that there is a significant difference in the groups at the start and end, t(29)=36.81, two-tail p = $6.72*10^{-26}$.

5.2.2 System 2 without using tags

A second experiment was performed, again with agents playing the Prisoner's Dilemma game, but with a different selection process this time. Although the player pool after each round was set up as before, with four players having been kicked-out of their groups for low cooperation and four players promoted into the pool for high-cooperation, on this occasion, the blue and red tags (indicating whether a player had been promoted or kicked-out of its group) were not used. Instead, players are placed in the pool based on the player's individual performance scores. The monitors of the groups pick the players from the pool starting from the highest ranked group. They chose players from the top of the pool. The highest performing group chose the player from the pool with the highest individual performance score (top one from the pool), and the subsequently ranked groups chose the remaining highest-scoring player available when their turns came up.

The changed process is outlined in Algorithm 5.2. In Algorithm 5.1 the lines between 22 and 26 are replaced by Algorithm 5.2 in this experiment.

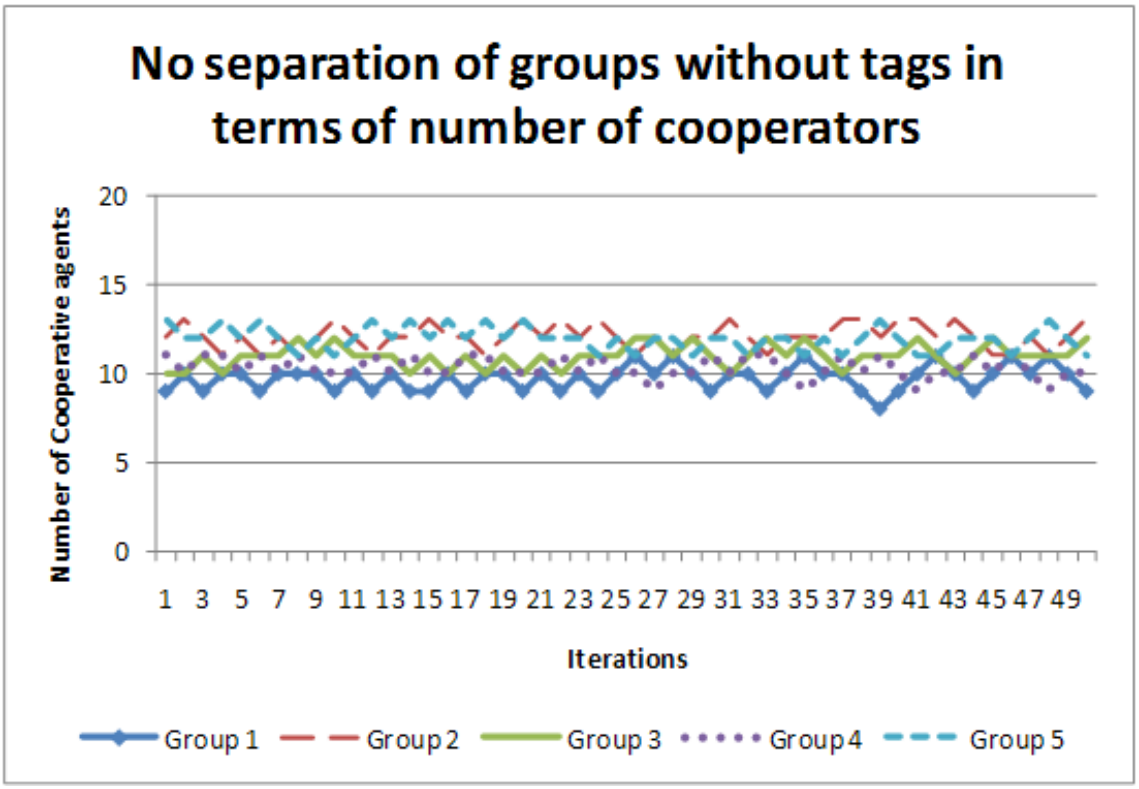

Figure 5.5: Prisoner's Dilemma without tagging.

Algorithm 5.2: Pseudocode for System 2 (without using tags)

```
1 begin
2     while selection from pool do
3         highest performing group chooses the player with highest individual
          performance score;
4         subsequently ranked groups choose the remaining highest-scoring player
          available;
5     end
6 end
```

Under these circumstances, when the group monitor agents chose high-performing agents from the pool, there was no separation into cooperative and noncooperative groups because the best performers (i.e. agents with high scores) are usually the noncooperators. This result is shown in Figure 5.5.

Thus when monitor agents select new group members based on their individual perfor-

mance, rather than by cooperation (signified by tags), there is no observed separation of groups.

5.2.3 System 3 using referral combined with System 1

Referrals have been shown to provide valuable information about an agent's behaviour (Candale and Sen, 2005; Yu and Singh, 2002; Yu *et al.*, 2004; Yolum and Singh, 2003b,a). The goal of this experiment is to increase the performance (score) of cooperators by using referrals along with separating groups.

In this experimental setup, an artificial agent simulation environment has been set with a society of 100 agents divided into 5 groups of 20 agents each. In each group, each agent played 10 games with 19 other agents in its group. Among these 10 games, we call the first 5 games as the first half and the next 5 games as the second half. In the first half, agents play using their degree of cooperation *dc*, randomly assigned to them. Every agent keeps the history of the first half which can be referenced during the second half, so they know who cooperated or defected in the past 5 games. But they know nothing about how much the other agents might have cooperated with each other. In the second half each agent asks for a referral about the opponent from its best 5 cooperators it has played with in the first half. Among five of them, if at least 3 of them say that the opponent is a cooperator, the agent will cooperate, otherwise it defects. So each agent plays 190 (19 *10) games in a round. Then the group monitor will conduct a voting among the group members to determine which are the most cooperative and least cooperative members of each group. It promotes the most cooperative member and demotes the least cooperative member (similar fashion to System 1). With the newly created groups another round of play is initiated. Remember the agents use the assigned constant *dc* value to select the strategy in the first half of every game. In the second half, they use referral scheme to select the strategy.

Thus in this experiment we have studied the effect of adding referrals to the tag based system described in Section 5.2.1 of Algorithm 5.1. The difference between Algorithm 5.1 and this experimental setup is in lines 6 to 9. Those lines are replaced by the pseudocode given in Algorithm 5.3.

From this experiment we are interested in seeing how well the performance of noncoop-

erators can be restricted by using referrals. We also seek to examine whether societal benefit can accrue by using referrals and letting the agents change their playing strategies based on the opponent's past behaviour (from the referral information).

Algorithm 5.3: Pseudocode for System 3 (using referral combined with System 1)

```
1  begin
2      foreach agent do
3          while firsthalf do
4              play with every other agent in the group;
5              keep the history about the other agents;
6          end
7          while secondhalf do
8              before playing, the agent asks for referral about the playing partner;
9              if partner is a cooperator then
10                 cooperate;
11             else
12                 defect;
13             end
14         end
15         rank the partners based on their cooperativeness;
16     end
17 end
```

The game was played over 100 rounds; initially every group had roughly an equal number of cooperators and defectors. As the play progressed, the groups started separating and we could clearly distinguish different groups (similar to Figure 5.3) at the end with different degrees of cooperativeness. There were:

- two groups that had almost all cooperators
- two other groups that had mostly defectors
- a middle group that had about half cooperators and half defectors.

With the separation of groups (based on their cooperativeness) we compared the scores in the first-half of the game with the second-half. Our general observation when using the referral scheme is that the groups that have more cooperators in the population gain higher scores than what they scored in the first-half. The most uncooperative groups (full of defectors) score less than what they obtained in the first half. In such groups, by using referrals, every agent comes to know that its potential partner is a defector, so they all defect and get a low score. Because of this, their group score is also low. The performance of a group which has an equal number of cooperators and defectors is determined by its overall tendency to cooperate/defect (which is based on the individual cooperativeness of all the members of a group). If the tendency to cooperate is higher than defection, the group increases its score over its first-half score. This is also the same for the groups which have very small difference in the number of cooperators and defectors. For instance, a group with 8 defectors and 12 cooperators can get a lesser score in the second half, if the group's overall tendency to cooperate (adding individual's cooperativeness together) is less than to defect.

Groups (ranked by Overall Performance)	1	2	3	4	5
Cooperators : Defectors	18:2	16:4	10:10	4:16	0:20
Score in First half	3397	3293	2961	2688	2247
Score in Second half	3531	3374	3147	2636	2174

Table 5.2: Different groups and their performance.

Table 5.2 shows the values from a sample run. From Table 5.2 we can see the number of cooperators in each group and the group's score in the first and second halves. The first two groups had mostly cooperators, the middle group had both cooperators and defectors in equal numbers, the fourth group had only 4 cooperators and the fifth group was full of defectors. For the first 3 groups the score increased in the second half. For the remaining 2 groups the score decreased since many agents in the group were defectors.

In the second half a cooperator cooperates only with other cooperators. Thus, the cooperator avoids bad transactions with defectors. This shows the referral scheme works well enough to improve the scores of cooperators, which is beneficial for the society.

5.2.4 System 4 using resource restriction combined with System 1

For agent societies to operate effectively, all agents within the society must play by its rules. When some of them do not abide by the rules and exploit the common resource, they must be restricted from doing so. A related issue is that of exploitation of common pool resources. To deal with this problem, the resource access for the exploiters must be limited for the welfare of the entire society. The experimental setup described in this section describes the resource restriction approach in which the 'number of games allowed to play' is considered as a resource. For the noncooperators the number of games is restricted based on their behaviour.

We conducted two types of experiments, one with resource restriction and the other without resource restriction. We compared the performance of the society based on average scores.

5.2.4.1 Experiment without resource restriction

The first experiment conducted was without resource restriction. It was conducted with the same environment as System 1 described in Section 5.2.1. There were 100 agents divided into five groups each having 20 agents. They played 100 rounds. Each agent played five games in a round with other agents in its group. After 15 rounds, the groups started separating similar to the result shown in Figure 5.3. The five groups separate with two groups full of mostly cooperators, two groups full of mostly defectors, and a middle group having half of both. The *total score* of the system was calculated by adding together the *overall performance*, *op*, for each group. The *average score* for each group is calculated by dividing the *overall performance*, *op* by number of games (five). The sum of these *average scores* of the five groups is calculated and we call it the *average without resource restriction*.

5.2.4.2 Experiment with resource restriction

The number of games played is treated as the resource. We limit the resource (number of games) for the groups that are not performing well. The limitation for using the resource varies between groups according to the group's cooperativeness. Groups are ranked based on their cooperativeness. The 1st ranked group is allowed to play 5 games as usual, since it is the best performer. The 2nd ranked group is restricted to play 4 games, the 3rd ranked group

to play 3 games, the 4th ranked group to play 2 games and the 5th ranked group to play just 1 game, since it is the worst performer. The *total score* of the system is calculated by adding together the *op* for each group. The *average score* for each group is calculated by dividing the *op* by the number of games (1 to 5 based on the group's rank). The sum of these *average scores* of the five groups is calculated and we call this the *average with resource restriction*.

Algorithm 5.4: Pseudocode for System 4 (using resource restriction combined with System 1)

```
1 begin
2     foreach group do
3         5 games for first ranked group;
4         4 games for second ranked group;
5         3 games for third ranked group;
6         2 games for fourth ranked group;
7         1 game for fifth ranked group;
8     end
9 end
```

The pseudocode of resource restriction for groups is given in Algorithm 5.4. The lines given in Algorithm 5.4 should be inserted between lines 26 and 27 of Algorithm 5.1 to illustrate the overall procedure.

We compared the scores of both (with and without resource restriction), the scores with resource restriction is higher because of limiting the exploitation by having restriction on games played (for noncooperators). The comparison of these two results (with and without resource restriction) are presented as boxplots in Figure 5.6. The depicted values are minimum (min), quartile1(Q1), maximum (max), median and quartile3(Q3). These results show that restricting resources for exploiters is beneficial for the society.

A paired t-test was conducted to determine if the resource restriction was effective in improving the societal performance by comparing the *average scores* (with and without resource restriction). The paired t-test was performed with null hypothesis over 30 sample runs. The mean societal performance (M=51.7, SD =33.96, N= 30) was significantly greater

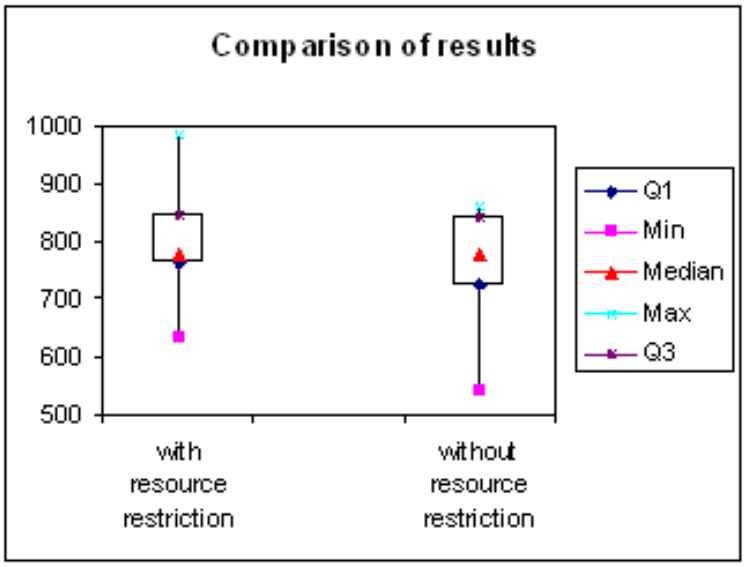

Figure 5.6: Comparison of results in boxplot.

than zero, t(29)=8.33, one-tail $p = 1.7 * 10^{-9}$, providing evidence that the resource restriction is effective to increase societal performance with a 95% confidence interval.

5.3 Summary

In this chapter, we have described mechanisms that help to restrict exploitation and improve societal performance in a multi agent society that has both cooperative and uncooperative agents. These mechanisms use monitoring agents locally within the group, which makes our system semi-centralised. Our first mechanism used tags and voting, and the system showed self-organisation. When tags were not used, there was no self-organisation observed.

Some of our experiments used referrals along with tags. The previous history from the first half was used to make decisions in the second half. This has improved the results of the second half. We have also investigated resource restriction mechanisms and have demonstrated that group level resource restriction can be used to improve the overall societal performance. The referral mechanism is used to restrict the individual noncooperators. The resource restriction mechanism is used to restrict groups with uncooperative behaviour. We have demonstrated that our proposed mechanisms help in improving the societal benefit and restricting exploitation. Our experiments encourage the forming of groups with differing

degrees of cooperation among its members. This can make it easier to deal with a variety of agent behaviours and still maintain an agent society that is open to new membership. Agents with lower reputations can be restricted to operate within groups that may have less access to system resources. As agents "prove themselves" and improve their reputations, they may be gradually given access to more desirable groups within the society.

Our overall conclusion, then, is that with the agent-based mechanisms that employ tags, monitor agents, and some constraints on accessing other groups and together with other social mechanisms, it is possible to foster more cooperative group behaviour within highly interconnected agent societies. But still there is a need to move from semi-centralised to decentralised mechanisms for establishing social control in distributed societies. This forms the focus of the next chapter.

Chapter 6

Self-organisation in decentralised closed systems

6.1 Introduction

One of the most persistent problems in P2P networks is freeriding (Ramaswamy and Liu, 2003; Krishnan, Smith, Tang, and Telang, 2004; Feldman and Chuang, 2005). There are published examples of centralised approaches in facilitating cooperation that employ centralised regulations to control freeriders (Esteva *et al.*, 2004). These researchers have used monitoring agents or governor agents to control agent behaviour. Even though centralised systems have several advantages, such as direct control and access, they have several limitations as well. They suffer from bottlenecks when the number of agents increases in the system. They are computationally expensive, because of the cost associated with avoiding performance bottlenecks, and they are prone to a single-point-of-failure.

With the increase in processing power and storage capacity of low-cost, lightweight computing devices such as smart phones, the arena of computing is becoming much more distributed. The clients of file-sharing systems are not only personal computers but also smaller devices, such as smart phones (Moya, 2010). There is a need for decentralised solutions to deal with the freeriders in these distributed societies.

In this chapter we develop social mechanisms for self-organisation of agents in a peer-to-peer like environment and present our simulation results. We investigate how the self-

organisation of distributed groups based on performance can be achieved. Such a system would self-organise to protect cooperators from being exploited by the non-cooperators. It would also restrict the non-cooperators from taking advantage of cooperators by restricting their access to better groups, where the quality of service or performance is higher. By doing so the performance of the whole system can be improved, because resources can be distributed in greater proportion to the better performing groups. Otherwise, it will be difficult to shield the cooperators from the defectors who hardly or never share their resources.

To that extent, this chapter proposes a decentralised solution that makes use of social mechanisms such as tags, gossip and ostracism. The inspiration to use social mechanisms for our work comes from the human societies, which have evolved over time to work effectively in groups (if need arises). For human beings, group mechanisms provide social machinery that supports cooperation and collaboration. Social control can be employed through leadership mechanisms. For example, the leader can impose rules on his followers. The disadvantage of such an approach is that it is centralised. On the other hand, it is known that social control can also be achieved by decentralised approaches.

A gossip-based mechanism can be used to achieve social control, as it serves as a distributed referral mechanism where information about a person is spread informally among the agents. Another social mechanism that can be employed to deal with freeriders is ostracism. Members that do not adhere to the values, expectations, or norms of the groups can be sanctioned by other agents by their refusal to interact with those agents.

In this chapter we demonstrate how these social mechanisms can be developed and employed for agents in a closed and decentralised society which has several groups.

Similar to the objective of the previous chapter, our aim here is to restrict exploitation or in other words restrict uncooperative behaviour by separating groups based on performance. Thus we investigate decentralised mechanisms to facilitate self-organisation of groups.

6.2 Proposed mechanisms for self-organisation of groups

The five proposed mechanisms are:

- Rank-based grouping mechanism (Section 6.4)

- Dynamic grouping mechanism (Section 6.5)
- Random hopping mechanism (Section 6.6)
- Individual group history mechanism (Section 6.7)
- Sharing group history mechanism (Section 6.8)

We have developed them one after another to overcome the drawback of the previous mechanism. Our motivation in all these mechanisms are to achieve self-organisation of groups (separation based on cooperativeness). These mechanisms are explained in detail in their separate sections which also explains why they were selected or preferred over the previous mechanism. Common steps in the proposed mechanisms are

- **Gossip interaction**

  This step deals with how agents gossip and how they make use of the gossip information.

- **Leaving a group**

  In this step agents may leave a group. There are certain conditions under which agents leave a group.

- **Choosing a group to join**

  In this step agents decide which group to join next.

- **Entry criteria**

  This step is to set the entry criteria (eligibility condition) to join a group.

All these steps are explained in detail in their separate sections within the proposed mechanism. These proposed mechanisms and their differences are tabulated in Table 6.1.

6.3 Agent attributes in our experimental setup

For this experimental model we have used agents which have fixed, randomly assigned attribute values which represent how they behave.

- **Cooperativeness**

  This attribute specifies how cooperative an agent is. An agent has a randomly assigned cooperation value between 0 and 10 that represents how much it cooperates (shares), with 0 representing an agent that never cooperates and 10 representing an agent that cooperates every time. This value is known as the cooperativeness of the agent.

- **Gossip blackboard length**

  Each agent has a gossip blackboard of certain length to store the gossip messages from other agents of its group. This blackboard will be used by an agent to post the gossip information provided by other agents. For example agent A posts the gossip it heard from agents B and C on their interactions with agents D and E respectively. These are individual blackboards for agents, but can be referred by other agents.

- **Cost and benefit for sharing**

  Agents share files. Whenever a file is shared, the receiving agent receives 1 as benefit while the sharing agent loses 0.1 as cost (cost is associated with sharing since the sharing agent loses time and bandwidth for sharing).

- **Tag groups**

  In the initial setup agents are put into random groups. Each group is represented by a tag (badge). Agents within a group have the same tag. They interact within their group, and they can also move to other groups under certain conditions. In such cases they join the other, jumped-to group, and the tag changes accordingly.

Mechanisms	Gossip interaction	Leaving a group	Choosing a group to join	Entry criteria	Separation of groups observed
Rank-based grouping	to calculate the estimated co-operativeness	•after playing a certain number of games •group filtering	group's calculated cooperativeness and group order (Table 6.3)	group order entry value (Table 6.3)	Yes
Dynamic group-ing	•to calculate the estimated cooperativeness •to avoid interaction with worst player	•tolerance level •rejection limit	group's calculated cooperativeness	based on formula (Equation 6.4)	Yes
Random hop-ping	•to calculate the estimated cooperativeness •to avoid interaction with worst player	•tolerance level •rejection limit	random order	based on formula (Equation 6.4)	No
Individual Group history	•to calculate the estimated cooperativeness •to avoid interaction with worst player	•tolerance level •rejection limit	individual experience of previous group(s)	based on formula (Equation 6.4)	No
Sharing group history	•to calculate the estimated cooperativeness •to avoid interaction with worst player	•tolerance level •no increase in wealth	based on members' shared experi-ence of previous groups	based on formula (Equation 6.4)	Yes

Table 6.1: Developed mechanisms for self-organised group composition to reduce exploitation.

6.4 Rank-based grouping mechanism

In our simulation model agents are engaged in the sharing of digital goods in a simulated P2P environment of an artificial agent society. In the initial setup 100 *(Number of agents)* agents are randomly divided into 5 *(Number of groups)* groups, 20 each. Each agent is initialised with a random cooperative value between 0-10. Each agent has a blackboard with a size of 10 *(Gossip blackboard size)*, which is a blackboard for storing gossip information. After reaching the limit, it rolls over based on First-In-First-Out (FIFO) algorithm. This setup updates old gossip information as new gossip information comes in. These blackboards are individual blackboards for agents. We do not model them as common blackboards because if a common blackboard fails, then that would affect a group of agents (Balaji and Srinivasan, 2010). The interaction between the agents is in the context of sharing files.

The experimental parameters are listed in Table 6.2.

Parameters	**Values**
Number of agents	100
Number of groups	5
Number of iterations	1000
Cost for donation	-0.1
Benefit for receiving	1
Number of iterations before hopping	100
Number of gossip requests	5
Minimum number of games in a group before hopping	10
Gossip blackboard size	10

Table 6.2: Experimental parameters for rank-based grouping mechanism.

6.4.1 Gossip interaction

An agent can make a request for a file to its fellow group agent. Whether the agent gets the requested file or not, it can gossip about the outcome to another agent in its group. In the gossip mechanism, there is no lying. It is assumed that the agents report honestly, since this

happens within the group. In this fashion, every transaction is reported (gossiped about) to one of the other agents in the group. Thus the total system has some partial information about every agent, maintained in a distributed way. The first one-tenth of the iterations were played in this manner to build up a distributed gossip repository among agents. For simplicity, the operation of how agents gossip within the group is outlined in Algorithm 6.1. Consider agents A, B and C.

Algorithm 6.1: Pseudocode for gossip.

```
  /* A gossips about B to C                                    */
1 begin
2     A requests for file to B ;
3     if B shares then
4         A gossips positively about B to C;
5     else
6         A gossips negatively about B to C;
7     end
8 end
```

6.4.2 Leaving a group

After 100 iterations *(Number of iterations before hopping)*, agents are able to hop groups if they have played 10 games *(Minimum number of games in a group before hopping)* in their group. We call the agent that tries to hop to a new group a 'hopping peer'. It is assumed that an agent may hop because either it is not satisfied with the performance of the group or it wants to explore other options (i.e. moves to other groups hoping for better personal utility).

6.4.3 Choosing a group to join

The hopping peer makes a request to a random agent of the new group to be permitted entry to its group. We call this agent from the new group a 'new-group peer'. For example, consider agents X and Y. X is the hopping peer, and Y is the new-group peer. In order to allow X into its group, Y asks 5 *(Number of gossip requests)* random agents in X's group

for gossip about X. The value of cooperativeness of X is calculated by this collected gossip information. This would be a value between 0 and 10. This value is called the *'estimated cooperativeness'* for X. The value is calculated by considering the worst-case scenario, and can be calculated as below.

$$Estimated\ cooperativeness = \frac{number\ of\ times\ cooperated}{1 + number\ of\ times\ played} * 10 \tag{6.1}$$

For example, if an agent has cooperated 3 out of 4 times then its estimated cooperativeness will be 6 based on the above formula. We add 1 to the number of times played because we want to calculate by assuming the agent defects in the next play (worst-case scenario). This way the agent which played just one game and cooperated in that game and an agent which played 2 games and cooperated in both the games would not be assessed equally (even though they both cooperated 100%).

An agent which cooperated 1/1 will get a score of 5.0 by Equation 6.1.

$$\frac{1}{1+1} * 10 = 5.0 \tag{6.2}$$

An agent which cooperated 2/2 will get a score of 6.6 by Equation 6.1.

$$\frac{2}{1+2} * 10 = 6.6 \tag{6.3}$$

Again Y asks 5 agents in its own group, to report their cooperation history and it calculates their cooperativeness together. This is called Y's *'group's calculated cooperativeness'*. If X's estimated cooperativeness is higher than Y's group's calculated cooperativeness value, then X is allowed entry to the new group, otherwise it is rejected. If rejected, X can apply to other groups in the same manner (including its previous group). There would be one group in the system which accepts all the agents regardless of their cooperativeness value, which is

the lowest performing group. By this process, an agent will get into a group only if its cooperativeness is eligible to get into that group. The process of joining a new group is outlined in Algorithm 6.2.

Algorithm 6.2: Pseudocode of the process of an agent joining another group.

```
/* X asks Y if it can join Y's group                                    */
1  begin
2      Y requests for gossip information about X from 5 other players in X's group;
3      Y computes X's estimated cooperativeness based on the gossip information
       obtained;
4      Y requests 5 other players in its own group to report their cooperation history for
       computing the entry value (group's calculated cooperativeness);
5      if X's estimated cooperativeness >Y's group's calculated cooperativeness then
6          X gains entry;
7      else
8          X does not gain entry;
9      end
10 end
```

The process that has been described so far would only allow or deny entry to agents of other groups based on their estimated cooperativeness. But it will not get rid of low performing players from their current group. For that to happen, the system needs a sophisticated mechanism which could expel poor performing agents.

Agents can estimate the cooperativeness or performance[1] of their own groups. It is calculated by using all the available gossip in the group. The cooperativeness of every agent in the group is calculated and the average is known as *'group's calculated cooperativeness'*. But the agents have no idea about the performance of other groups. How good your group is compared to other groups can be determined only if you know about the performance of other groups. For setting an entry value for the group, agents in a particular group ask agents in other groups about their group performance. This process is explained in Section 6.4.4.

[1]The performance of the group depends on the cooperativeness of its members.

6.4.4 Entry criteria

In order to achieve the desired maintenance (cooperation standard) of a group's population, in every group, a random agent is selected on each iteration after the 100th iteration (i.e. after one-tenth of the iterations), for a fitness check to decide whether an in-group agent is up to the standards of the group or not. We call this agent the 'chosen peer'. The chosen peer contacts randomly one of the agents in its group for the checking. The agent who checks is called the 'checking peer'. Now the checking peer must decide whether to let the chosen peer stay in the group, based on the group's performance. The group's rank can be determined only after knowing the performance of other groups. An agent has no idea about other groups' performance. So the checking peer asks another random agent in each of the other groups in order to obtain an estimated value of those other groups' cooperativeness which is a calculated value based on the gossip information available in the group. The random agent in that group collects the gossip information from everyone in its group and calculates the cooperativeness of every agent in the group through the gossip. The average of these cooperative values of all the group members is considered to be the *'group's calculated cooperativeness'*. After getting the information from the agents in other groups about their group's calculated cooperativeness, it ranks its own group (as shown in Table 6.3) based on the cooperativeness and chooses the entry value for its group. The highest performing group sets the highest value as its entry value, and so on. Agents who want to remain in this group should have a cooperative value (calculated through gossip) above the group's entry value in order to stay in this group. Otherwise, the agent is not allowed to stay in the group. This is a kind of 'ostracism'. Group order and their entry values are shown in Table 6.3[2]. The checking peer also checks for gossip information about the past behaviour of the chosen peer in the chosen peer's group. Now if the chosen peer's estimated cooperativeness is higher than the entry value for the group, then it allows the agent to stay, otherwise it is rejected/ostracised. In a similar fashion, a chosen agent is randomly selected and checked in every iteration (after the 100th iteration). This filtering process is outlined in Algorithm 6.3.

[2]We could choose the range of separation by setting the entry value. For example, each group could have 20% of members of the whole population. The top group can have the top 20% of the cooperators, the next top group can have the next 20% of the cooperators and so on.

This filtering process expels the low performers from the group. But the lowest performing group does not have any filtering; hence it will accept anyone (usually the poor performers).

Groups Order by performance	1	2	3	4	5
Group Entry Value	>8	>6	>4	>2	0-10

Table 6.3: Group order and entry value.

Algorithm 6.3: Pseudocode of the filtering process.

```
1  foreach chosen peer contacts checking peer do
2      checking peer gets chosen peer's estimated cooperativeness;
3      checking peer requests a random agent in each of the other groups for their
       group's calculated cooperativeness;
4      checking peer receives other groups' calculated cooperativeness;
5      checking peer sets the entry value for its group (based on Table 6.3);
6      if chosen peer's estimated cooperativeness > entry value then
7          chosen peer stays;
8      else
9          chosen peer leaves;
10     end
11 end
```

The whole process is repeated for several iterations. At the end of all the iterations the separation of groups based on cooperativeness is achieved. The whole process is outlined in Algorithm 6.4.

6.4.5 Results of experiments

It can be observed from Figure 6.1 when we performed an empirical test using simulation, that different ranges of cooperative peers were located in different groups. Initially all the groups started with roughly similar cooperativeness. They ended up showing different ranges of cooperativeness as shown in Table 6.4 and Figure 6.1.

Table 6.4 shows the initial and final cooperativeness of groups which is also depicted in Figure 6.1.

Algorithm 6.4: Pseudocode for rank-based grouping mechanism.

```
initialisation;
bootstrap agents in groups;
foreach iteration do
    select random number of agents;
    foreach selected agent do
        play with another agent in the group;
        collect payoff;
        gossip ;                                          // Algorithm 6.1
    end
    if iteration ≤ 100 & selected agent has played a minimum number of games then
        allow selected agent to leave and join another group ; // Algorithm 6.2
    end
    if iteration > 100 then
        Collect other group's calculated cooperativeness;
        Rank the groups based on Table 6.3;
        foreach group do
            check a random agent for group standards ;    // Algorithm 6.3
            if group standards met then
                agent stays in the group;
            else
                agent leaves and joins another group ;    // Algorithm 6.2
            end
        end
    end
end
```

Groups	1	2	3	4	5
Initial cooperation value	5.8	5.4	5.3	4.5	4.3
Final cooperation value	9	7.4	6.6	5.8	1.8

Table 6.4: Initial and final cooperation value of groups.

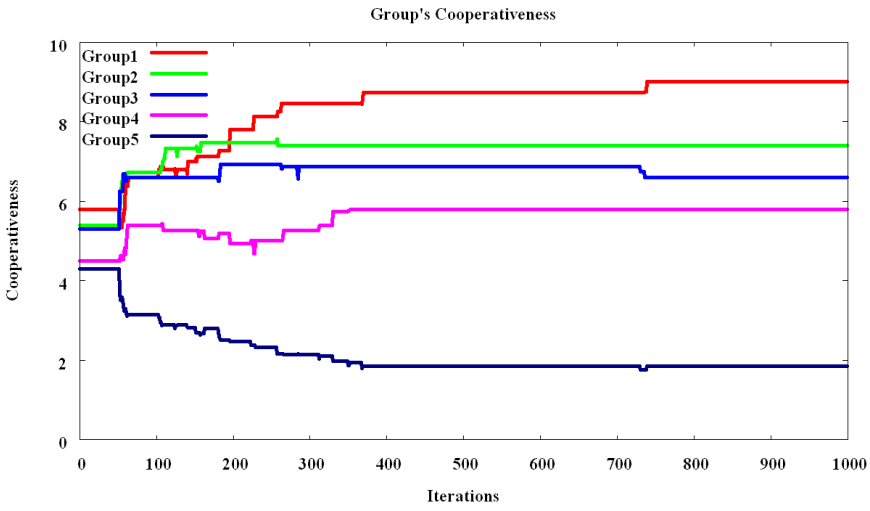

Figure 6.1: Self-organisation of groups using rank-based grouping mechanism.

This system filters out the worst peers well and restricts them from accessing contents from groups of cooperators. This also helps to protect cooperators from not being exploited by freeriders. The best agents get to access or enter into any group and finally end up in the best group.

Using the gossip mechanism, agents share their interaction experiences with some other agents. Unlike reputation mechanisms, where agents keep track of all the reputation information of other agents, only some individuals have this gossip information. This mechanism allows for the partial-view of truth about the world states which is a realistic and scalable feature of agent societies. Even if a peer leaves a group, not a lot of information is lost, but only a very small proportion.

In this experimental setup, the agents get the information about the performance of other groups. Relying on that information, they rank the groups and set their group's entry value as shown in Table 6.3. Even though this mechanism facilitates a separation of groups based on

their cooperativeness, the mechanism for determining a group's entry value based on Table 6.3 can be viewed as a constraint specified *a priori* (i.e. at design time). This is suited for systems where the designer wants to decide the range of separation based on particular values (similar to the ones shown in Table 6.3). If the number of groups change at runtime the values for the range of separation can not be changed accordingly. However these values for separation range could be calculated during run time based on a group's current state which is explained in the next experiment in Section 6.5

6.5 Dynamic grouping mechanism

In order to address a potential weakness of the previous mechanism (Section 6.4), we investigate a mechanism based on dynamic group forming which does not follow *a priori* values. Instead the range of separation is derived by considering the current state of the group. This experiment has some changes from the previous experiment. For one thing, this experiment has 5000 iterations. In addition, agents in this experiment have two more attributes, called 'tolerance level' and 'rejection limit'.

The tolerance level is a value between 1 and 10, which characterises how much non-cooperation the agent can tolerate before it decides to leave the group. A value of 1 identifies the least tolerant agent, and 10 identifies the most tolerant agent. An agent with a tolerance value of 1 leaves the group after experiencing defection once and an agent with a tolerance value of 10 leaves the group only after 10 defections. Each time the agent experiences a defection it increases its tolerance count. The agent decides to leave the group when its tolerance count reaches its tolerance level.

The rejection limit is an attribute of an agent which represents how many *rejections* (rejections are described below) the agent can face before it decides to leave for another group. Every time it is rejected from the play, it increases its rejection count. The agent decides to leave the group when its rejection count reaches its rejection limit.

6.5.1 Gossip interaction

Gossip interaction takes place like the previous mechanism explained in Section 6.4.1. But the agents use this gossip information to avoid playing with the worst player of their group.

When a player requests a file, the giving player can check with five other random agents (asking them what they know from the gossip information they have received) whether this asking-player (taking-player) is the worst cooperator of their group. The worst player is the one who has been uncooperative the most often in its group (according to the available gossip information). If the taking-player is the worst player, the giving player refuses to interact with (i.e. rejects) the taking-player. Otherwise this giving player interacts (sharing a file or not based on its own cooperativeness). The operation of how peers use gossip is outlined in Algorithm 6.5, where D and E are the players in the group. Assume here that E is the taking-player, D is the giving-player, and D checks with any 5 players in the group in order to see whether E is the worst player in their group.

Algorithm 6.5: Pseudocode to avoid the worst player.

```
1 begin
2     D makes a request to 5 other players for gossip about E ;
3     D receives gossip;
4     if E is the worst player then
5         D refuses to play with E;
6     else
7         D plays with E ;
8     end
9 end
```

When only a few agents (less than 5) have gossip about a taking-player, then only the available information is taken into consideration. Sometimes it can be the case that none of the players has gossip information about the taking-player. In such a case the taking-player is considered not to be the worst player, a privilege similar to what happens when a new player joins a group.

6.5.2 Leaving a group

A player can leave a group because of two reasons. First if an agent's tolerance level is exceeded, which means the agent is in a group where others do not cooperate as much as the agent's expectation. After a number of defections from the group members, the agent may decide to leave that group. The agent asks other groups about their cooperativeness and tries to enter into the group whose cooperativeness is better than its current group. If the better groups don't allow the agent in, then since it does not want to move to lower groups, it stays in its current group and its tolerance count is reset to 0.

Second, an agent can move to another group based on its rejection limit. Agents can refuse to interact (i.e. share resources) with an agent if that agent is identified as the "worst", (i.e. the least cooperative agent) in the group. Instead of picking an agent randomly and checking it for cooperativeness (as mentioned in the previous mechanism in Section 6.4.4), the agents in this setup stop playing with a worst player (as shown in Algorithm 6.5) hence after being rejected for a certain number of times (the rejection limit), the worst agent chooses to leave that group and moves to another group.

6.5.3 Choosing a group to join

The leaving agent asks other groups about their group's calculated cooperativeness and tries starting from best to next best and so on, seeking to find a group which allows it in. If none of the groups allow it, then it stays in its current group.

The criteria for a group accepting or rejecting an agent depends on the situation of that group. In this setup a group's entry value is calculated based on the group's standards which are the group's current size and cooperativeness. Based on these two factors the entry value is determined and the agents seeking entry are assessed (remember in the previous mechanism the entry level was set to a particular value based on group rank using Table 6.3).

Determining a group's entry value is described in the next section (Section 6.5.4).

6.5.4 Entry criteria

The hopping peer asks any randomly chosen agent in the group to which it seeks permission to enter. We call this permission-granting agent in the group to which entry is sought, the 'checking peer'. The hopping peer will gain permission to enter the group whenever its cooperativeness is greater than or equal to the group's entry value calculated by the following formula:

$$\mathsf{EV} = \mathsf{C} - (\mathsf{C1}/(\mathsf{SL} - \mathsf{S})^{\mathsf{C2}}) + \mathsf{C3}^{(\mathsf{S}-\mathsf{SU})} \tag{6.4}$$

The group Entry Value (*EV*) is calculated considering the given group's calculated Cooperativeness (*C*) and its group Size (*S*). *C* is the group's calculated Cooperativeness through the gossip information and *S* is the size of the group. *C1*, *C2*, *C3* are constants whose values in our experiments were 25, 2, 10, respectively. These constants were adjusted to make the *EV* expression appropriate for two "boundary values", the upper size limit of a group (*SU*) and the lower size limit of a group (*SL*). It is inappropriate or inefficient for groups of players or traders to become too big or too small. If a group becomes too big, then it becomes unmanageable. If a group becomes too small, below a certain value, then the group is not considered to be an active group. There should be at least a certain number (minimum value) of players in it to be considered as a team (e.g. a sports team may have a certain number of players as a minimum for the existence of a team/group. Thus a volleyball team must have at least 6 players. If there are fewer than 6 players, then it is not considered to be a team/group). In our experiments, *SU* was set to be 25, and *SL* was set to be 10. That means that if the size of the group is 10 or below, the entry qualification value will be set to a low value, making entry into the group very easy to obtain. If the size is 25 or above the entry qualification value is set to a high value and that would make it difficult for any but the most cooperative agents to join. Any values of the *EV* expression that fall below 0 are set to 0, and entry values above 10 are set to 10. Thus a group's entry value is always between 0 and 10.

Figure 6.2 shows the range of entry values for groups with different group sizes and cooperativeness. A simple example illustrates the use of this formula. Consider that a group's calculated cooperativeness (*C*) is 6. When the group size (*S*) is 14 the group Entry Value (*EV*) is 4.43. When the group size (*S*) is 25 the Group Entry Value (*EV*) is 6.88. This can

be identified in Figure 6.2 by examining the line Avg6 for size 14 and for size 25. It can be inferred that if the group size becomes large (closer to *SU*), the group is strict about who it lets in and if the group size is small (closer to *SL*) it lets almost anyone into the group. When the group size is 10, the entry value is 0.

In our system, the checking peer needs to get an estimate of the cooperativeness of the hopping peer (the agent seeking entry). So the checking peer asks 5 randomly chosen players from the hopping peer's group about the hopping peer's cooperation. It is thus asking for gossip information from the hopping peer's group in order to assess the request of the hopping peer to join its group.

Consider a case where G and F are in different groups. G is the checking peer, and F is the hopping peer that wants to enter G's group. F asks G for entry, then G asks 5 other randomly chosen players in F's group for gossip information about F's cooperativeness. If F's estimated cooperativeness calculated through this gossip information is greater than or equal to the entry value (EV) of G's group, G allows entry for F; otherwise it denies. If it is denied entry to a group, F will try to enter into other groups. This process is outlined in Algorithm 6.6. The hopping peer will ultimately get into a group where its cooperativeness makes it eligible to enter. If no such group is available, the hopping peer stays in its current group.

Algorithm 6.6: Pseudocode for using gossip information to hop between groups.

```
/* Agent F seeks entry into the group of agent G        */
1  begin
2      G requests 5 other players in F's group for gossip about F ;
3      G receives gossip about F and calculates F's estimated cooperativeness;
4      G calculates entry value based on Formula 6.4;
5      if F's estimated cooperativeness ≥ entry value then
6          F gets entry;
7      else
8          F does not get entry ;
9      end
10 end
```

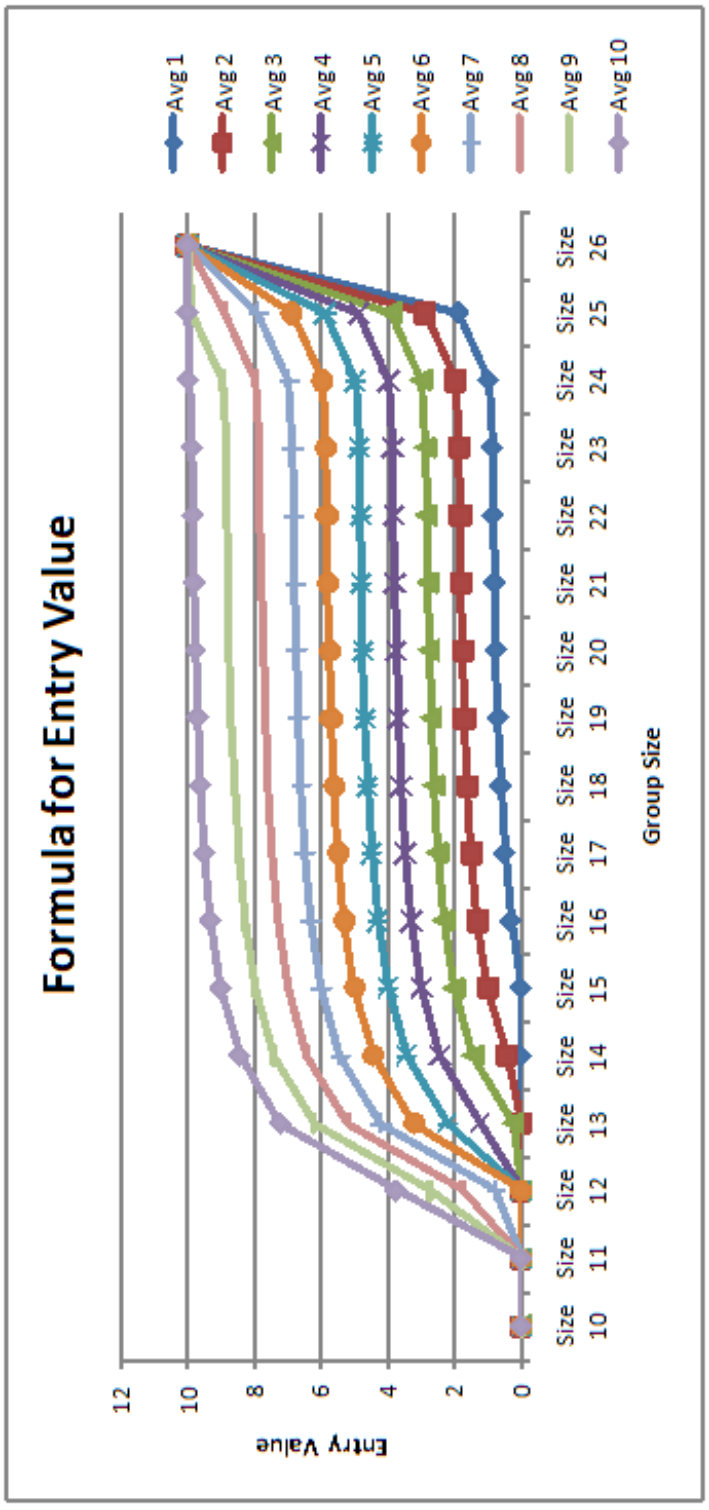

Figure 6.2: Formula for entry value.

6.5.5 Results of experiments

The entire process explained so far is repeated for 5000 iterations, and over time, the agents collect into groups where there are agents with similar cooperativeness to them. Because of this process some groups will emerge as elite groups with many cooperators, and other groups will have less cooperative players. As a consequence, this mechanism achieves a separation of groups based on performance. Figure 6.3 shows the self-organisation of groups based on their cooperativeness.

The main differences between the previous mechanism and the current mechanism are that: (a) this mechanism uses gossip information to avoid interacting with the worst agent, and (b) an agent decides to leave the group based on its tolerance level and rejection limit, and (c) in this mechanism the entry value is determined by group size and cooperativeness of a group at run time (whereas it was determined at design time in the previous mechanism). According to the number of groups, the specific values for group entry need to be specified at design time in the previously described mechanism, but for the current mechanism no such requirements are necessary.

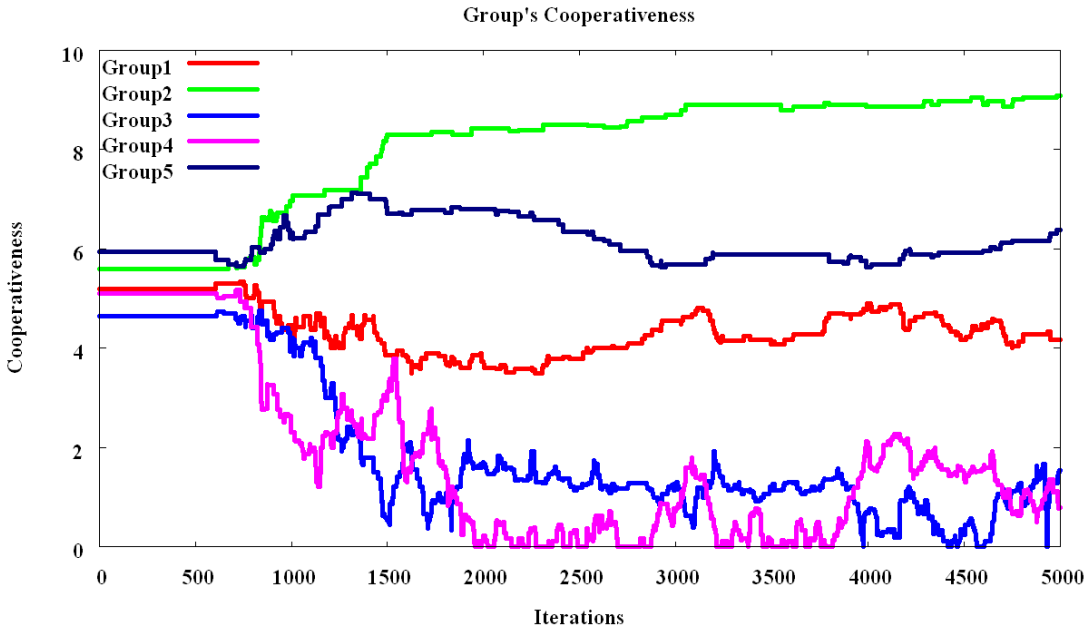

Figure 6.3: Self-organisation of groups using dynamic grouping mechanism.

6.6 Random hopping mechanism

We conducted this experiment to investigate whether random hopping of agents would lead to some kind of behaviour pattern in the system. This experiment has the same experimental setup as the previous experiment, except for a few changes. The agents do not have information about the other groups' performance (group's calculated cooperativeness) and so they decide to hop to another random group. After playing a certain number of games in a group (10), agents try to hop to another random group. The entry criterion is the same as the previous mechanism as explained in Section 6.5.4. This mechanism did not result in the separation of groups, since the agents are hopping randomly and they never settle down in a group. Due to this there is no separation of groups based on cooperativeness like the previous mechanisms.

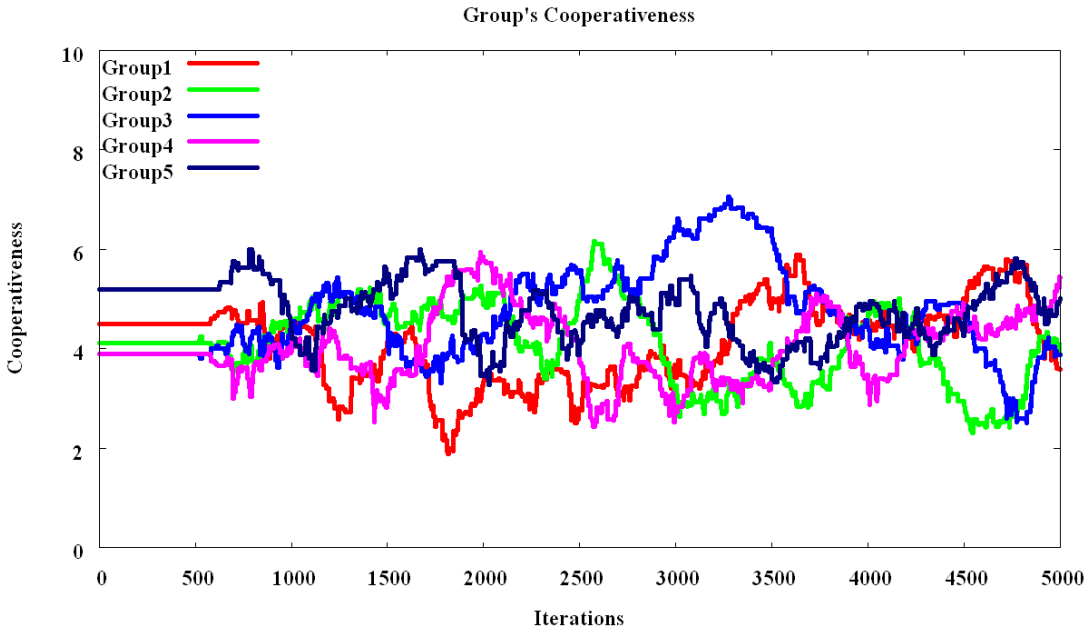

Figure 6.4: No self-organisation of groups based on cooperativeness using random hopping mechanism.

From Figure 6.4, we can see that there is not much difference between groups to distinguish them as most and least cooperative groups. The groups did not change much from their initial setup in terms of cooperativeness. This can be understood when the difference (in spread of groups) is observed in the graph of the previous experiment (Figure 6.3).

6.7 Individual group history mechanism

In order to overcome the random hopping issue, we developed a mechanism in which agents can keep the memory of their previous groups. Each agent has a memory of a certain number of previous groups to which it belonged. In all the three mechanisms we have explained so far, an agent has no memory of the previous groups it has belonged to. In this mechanism, agents keep the memory about previous groups' cooperativeness.

The information about other groups that an agent has been to previously, might be helpful to make decisions. By keeping the history of the performance of previous groups an agent bases its decision solely on its own experience with other groups. We believe this could potentially lead to self-organisation. We investigated the role of keeping the history of previously visited groups in this experiment. We have conducted two experiments, one with memory of just the previous group only and another with memory of all previous groups an agent has been a member of.

6.7.1 Memory of last group only

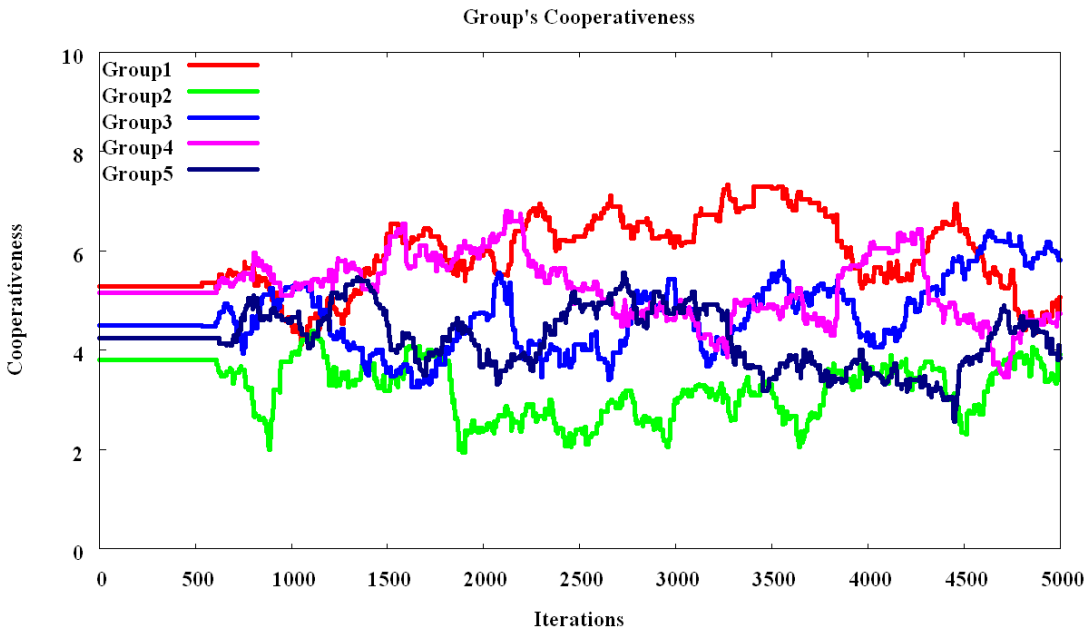

Figure 6.5: Group history mechanism with memory of last group only.

In this experiment agents have just one memory slot to store the information about its

previous group. After a certain number of games in a group, the agent compares its current group's cooperativeness and the previous group's cooperativeness. If the previous one was better, it tries to hop back to the previous group. Otherwise it stays in the current group. This experiment did not show significant self-organisation in terms of separation into groups of different cooperation values (see Figure 6.5), since the agents keep on hopping back and forth between previous and current groups.

6.7.2 Memory of all previous groups

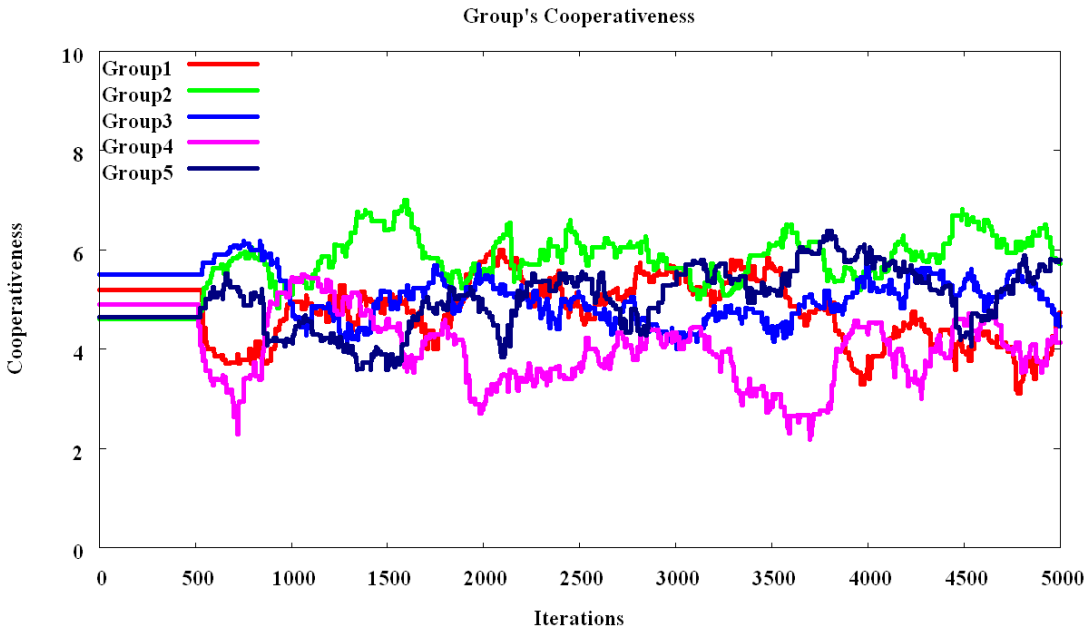

Figure 6.6: Group history mechanism with memory of all previous groups.

Instead of keeping just one memory the agents in this setup have memory of all their previous groups. The agent compares its current group's cooperativeness with the other groups in its memory. If its current group is not the best, then it tries to move to the best group in its memory. If not allowed, then it tries the next best group and so on. If it is not allowed in any of the better groups, then it stays in the current group.

For example, the group with best cooperativeness at iteration 1000 may later (say at the 3000th iteration) be the worst group and the agent does not know that. The agent left that group at iteration 1000, but later the group's cooperativeness has been changed because of

arrival and departure of agents over time.

Figure 6.6 shows there was no evident self-organisation of groups based on cooperativeness. Thus this mechanism also did not achieve significant self-organisation of groups based on cooperativeness since the agents keep on moving to different groups. The main reason for this is the agents not having the latest information about other groups. This is because the group's size and cooperativeness change from time to time. But agents have the memory from their previous experience which may not be recent. So this mechanism did not achieve self-organisation.

6.8 Sharing group history mechanism

As the previous mechanism (Section 6.7) did not lead to self-organisation because of agents hopping to other groups based on their own memory of cooperativeness, we investigated a mechanism where agents share the information and follow the recent information available. We call this mechanism the "sharing group history" mechanism. This experiment differs from the previous experiment in several respects, which are described in the appropriate sections below.

6.8.1 Gossip interaction

The gossip interaction takes place in the same manner as described in connection with Algorithm 6.1. Every transaction is reported (gossiped about) to one of the other agents in the group. Thus the overall system has some partial information about the cooperativeness of each agent, maintained in a distributed way. The first 500 iterations (1/10th out of 5000) are played in this manner to build up a distributed gossip repository among the players.

After 500 iterations, the agents begin using the received gossip information to decide whether or not to play with a taking-player. If the requesting agent is the worst cooperator in the group, the player refuses to play with him (in the same manner as described in connection with Algorithm 6.5).

6.8.2 Leaving a group

A player can leave a group if its tolerance level is surpassed or when its wealth (score) has not increased (remember every time the agent receives a file, it adds up 1 *(benefit for receiving)* to the score). If the agent's tolerance limit is reached, the agent will decide to leave that group and move to another group.

In addition, when other agents reject to play with the worst agent, and it is regularly rejected from play, then, of course, that agent's score will not increase. If, over a given period of play opportunities (e.g. 15 iterations), an agent's wealth (score) has not increased, then it will choose to leave that group and move to another group. Since the other players in its current group are not playing with it, it will be better off (i.e. able to increase its wealth) by moving to another group.

6.8.3 Choosing a group to join

The hopping peer then collects information about other groups from their group members. Then it decides from which group to request admission. Every agent has a memory record of its most recent groups (in our experiments the memory capacity was set to 4 past experiences).

For example, assume agent E has been in 3 other groups before, as shown in Table 6.5. The first row of Table 6.5 shows that E has left group 1 at the 560th iteration, and the cooperation value of that group was 4.5 at that time. E left group 3 at the 700th iteration when that group's cooperativeness was 6 and group 2 at the 1200th iteration when that group's cooperativeness was 6.4. Since the composition of groups invariably change over time, the cooperativeness of any group will change as time progresses (but can canverge as shown in the previous experiment). So it is likely that the recent information will be more accurate reflection of the cooperativeness of those groups. Since all agents have a memory of their previous groups, the hopping peer can collect this information from all its group members and calculates the latest information about other groups. In particular, the agents get to see which agent has moved into this group recently from other groups. Taking into consideration the most recent information available, the agent decides where to move. For example assuming the current iteration is 1400, the latest information collected from the group members is

given in Table 6.6.

Group No	Iteration No	Cooperativeness
1	560	4.5
3	700	6.0
2	1200	6.4

Table 6.5: Previous group history.

Group No	Iteration No	Cooperativeness
5	1330	8.1
3	1170	7.5
2	1200	6.4
1	1199	3.8

Table 6.6: Latest available information.

Assume here that agent L intends leaving group 4, and group 4's cooperativeness is 6.6 (group's calculated cooperativeness) at that moment. From the latest information agent L knows about other groups and their cooperation values. For agent L, groups 5 and 3 are better, since the cooperation value in those groups appear to be higher than L's current group. Groups 2 and 1 are lower-ranked groups. So agent L chooses to move to the groups in the order of their ranking.

If L is intolerant of its current group (which means it is not happy about the cooperativeness of its current group), it will try to enter into the best group that it can find. This is the case of an agent being "too good" (more cooperative) for its current group and wanting to move to a more cooperative group. But if the better groups on its list don't allow entry into their groups, then the intolerant agent L may determine that there is no group available that is better than its current group, and it will remain in its current group. In this case its tolerance count is reset to 0.

On the other hand, an agent may not be good enough for its current group - it is being shunned by the other members for being the worst member of its group. Because of play rejections, its wealth will not improve, and it will want to leave and find some other group

in which it can find players to play with. If the better groups do not allow entry, the agent will go to lower and lower groups, since it is better off moving to any new group rather than staying in the current group where it is known as the worst player and therefore shunned.

Algorithm 6.7: Pseudocode for sharing group history mechanism.

```
1  begin
2      initialisation;
3      bootstrap agents in groups;
4      foreach iteration do
5          select random number of agents;
6          if iteration is less than 500 then
7              foreach selected agent do
8                  play with another agent in the group;
9                  collect payoff;
10                 gossip;                                  // Algorithm 6.1
11             end
12         else
13             foreach selected agent do
14                 play with another agent based on gossip;  // Algorithm 6.5
15                 collect payoff;
16                 gossip;                                  // Algorithm 6.1
17                 if selected agent's tolerance level is met then
18                     collects recent information about other groups from group
                       members;
19                     selected agent leaves the group;
20                     joins another group under certain condition; // Algorithm6.6
21                 end
22                 if selected agent's wealth has not increased then
23                     collects recent information about other groups from group
                       members;
24                     selected agent leaves the group;
25                     joins another group under certain condition; // Algorithm6.6
26                 end
27             end
28         end
29     end
30 end
```

6.8.4 Entry criteria

How a player gets entry to another group (entry criteria) is as explained in Section 6.5.4.

The entire process is repeated for many iterations in this configuration and the separation of groups is observed. The overall process of this experiment is outlined in Algorithm 6.7.

6.8.5 Results of experiments

We present our results in Figure 6.7. Initially all the five groups were randomly seeded and started with roughly similar average cooperativeness values among their members. They ended up showing a separation among the groups based on their cooperativeness. Group 4 contained mostly the best cooperators, groups 2 and 3 had mostly non-cooperators, and groups 1 and 5 had moderate ones.

The snapshots of composition of groups at the start and end of the experiment can be seen in Figures 6.8 and 6.9 respectively. This partitioning was achieved using social mechanisms without central control. The larger green circles represent groups. Agents are color coded based on their cooperativeness which are represented by smaller circles inside the larger green circle. The range of cooperativeness values and their color code are: 0-2 is Red, 3-4 is Orange, 5-6 is Yellow, 7-8 is Green and 9-10 is Blue. In Figure 6.8 we can see that there were mixed colors in all five groups (at the start). In Figure 6.9 we can see that the agents have self-organised themselves into different groups based on their cooperativeness (showing predominantly different colors at the end). A demo video can be seen on UniTube (see (Savarimuthu, 2010a) for the link).

A paired-samples t-test was conducted to compare the separation of groups based on cooperativeness (standard deviation of cooperativeness of groups) at the start and end of the runs. The paired t-test was performed with null hypothesis (for 30 sample runs) that there is no significant difference between the standard deviations at the start and the end of the experiments. The standard deviation of the groups' cooperativeness at the start and end of the run were measured. There was a significant difference in the values at the start (M=0.71, SD=0.25) and at the end (M=3.27, SD=0.31) conditions. The average difference between

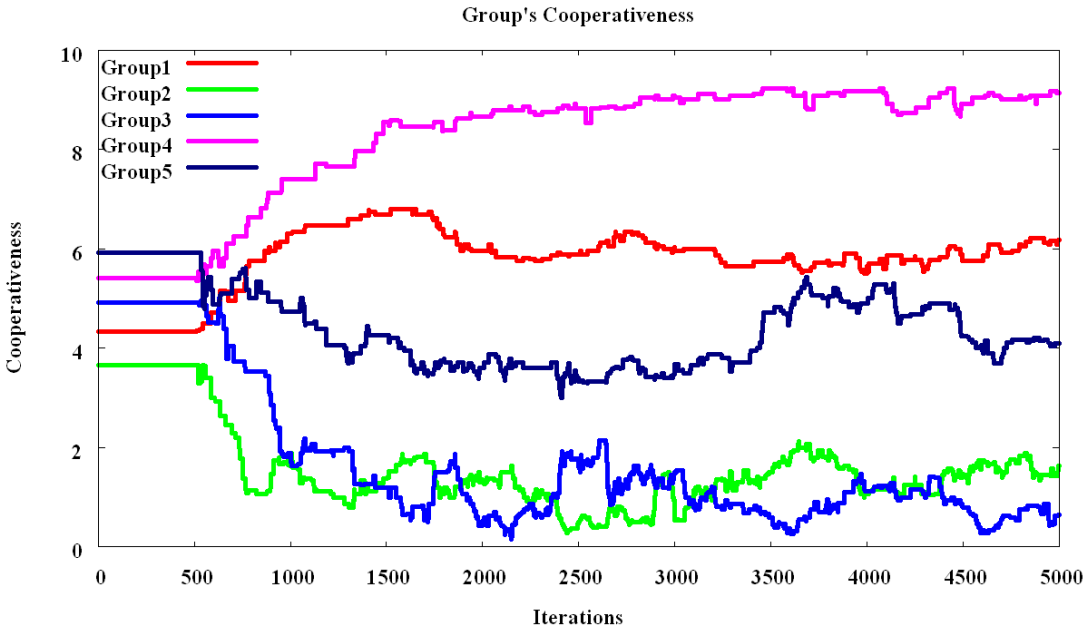

Figure 6.7: Self-organisation of groups using sharing group history mechanism.

the mean values (M=2.55, SD=0.07, N=30) was significantly greater than zero, t=33.77, one-tail p=($3.89*10^{-25}$), providing evidence that our mechanism is effective in producing the separation of groups based on cooperativeness with a 95% confidence interval.

We also experimented by varying the number of agents that are contacted to collect gossip. The number of agents from which gossip was provided varied. We experimented with 2, 5 and 15. From the 30 sample runs collected, the means of the standard deviations of groups at the end of the experiments, for gossip sizes 2, 5 and 15 were 2.80, 3.22 and 3.23 respectively. We noticed significant differences when we compared gossip sizes 2 with 5. Collecting gossip from 5 agents resulted in better separation than from 2 agents. But when we compared 5 with 15 there was not much difference in the separation. Collecting gossip from 15 agents (or less, if there are less than 15 agents in that particular group) has slightly improved the separation, but the difference was very small. This suggests that partial information is sufficient to establish group separation, and thus one can avoid additional computation, because asking 15 agents instead of 5 will require much more message passing and computing in an agent society.

Agent Color					
Cooperation Value	10-9	8-7	6-5	4-3	2-0

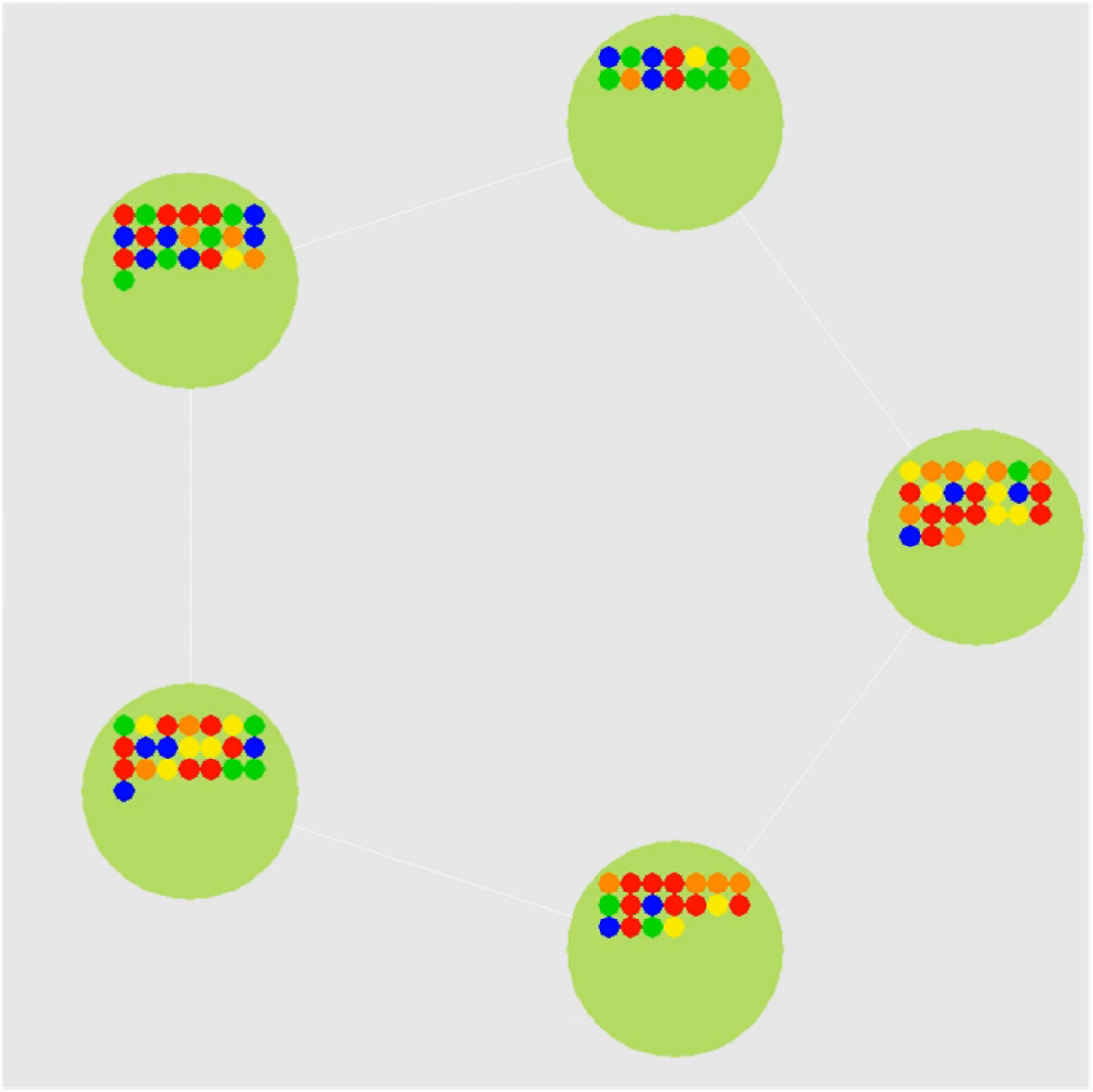

Figure 6.8: Composition of groups with agents at the start.

Agent Color					
Cooperation Value	10-9	8-7	6-5	4-3	2-0

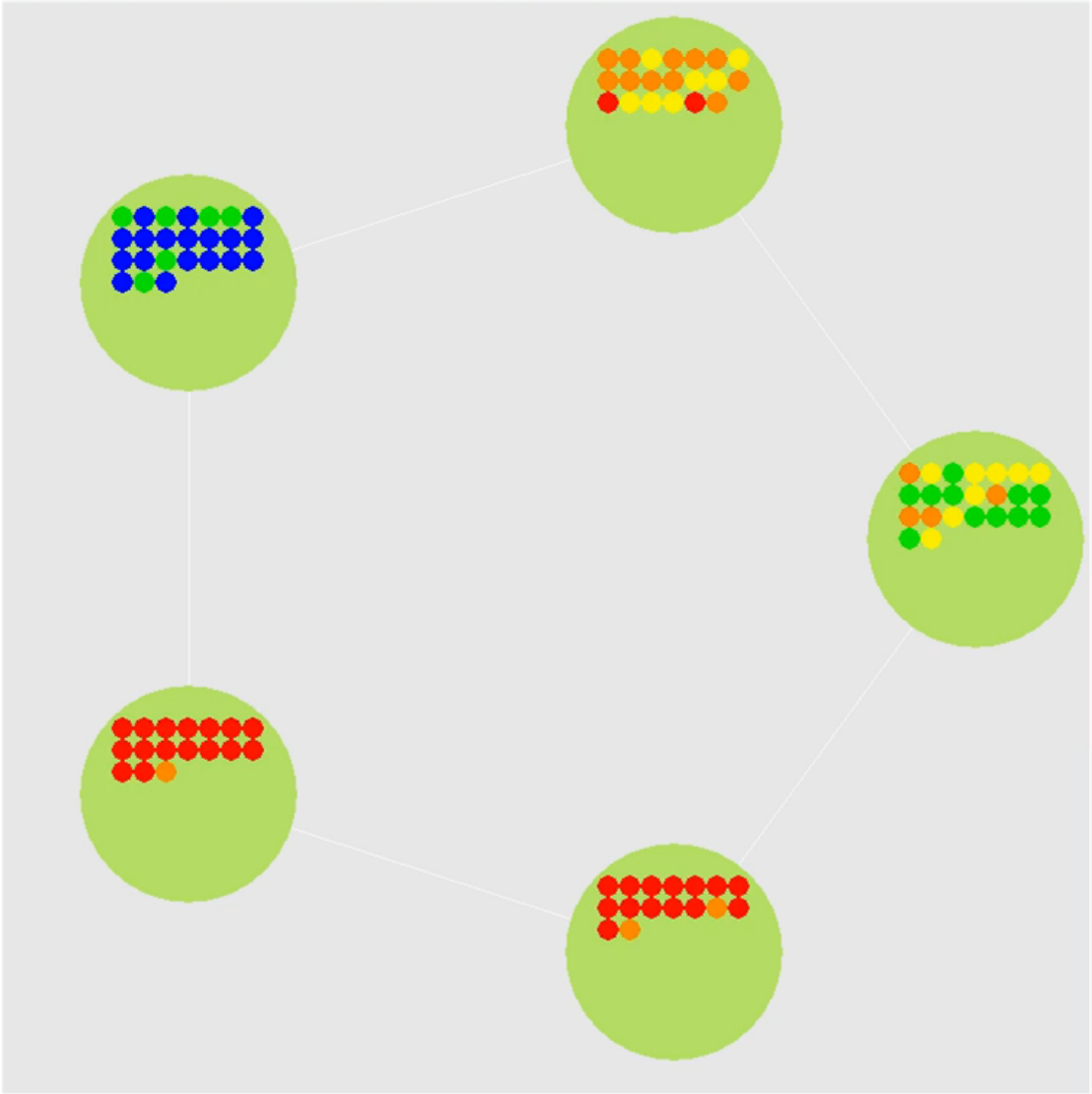

Figure 6.9: Composition of groups with agents at the end.

6.9 Summary

In this chapter we have explored five mechanisms which are suitable for closed societies and which could potentially lead to self-organisation or separation of groups based on degrees of cooperativeness.

- The rank-based grouping mechanism achieves self-organisation and it is suitable for systems in which the performance of the other groups are directly revealed to the agents in the society.
- The dynamic grouping mechanism also achieves self-organisation and it has been enhanced for systems to set entry values dynamically considering the current system state.
- The random hopping mechanism does not lead to separation of groups because of random hopping and agents not settling.
- The group history mechanism also does not lead to separation because of lack of latest information about other groups.
- The sharing group history mechanism leads to self-organisation or separation of groups based on cooperativeness. Agents share their information about other groups, hence the latest information is available.

This system is suitable for sharing digital goods in P2P systems, where the aim is to restrict freeriding. It has produced self-organisation, or the so called self-balancing of P2P systems, in a distributed and dynamic manner in closed societies. This system setup takes advantage of social mechanisms, such as tagging to form groups, gossip to pass information, and ostracism to shun bad behaviour. As a result, it shows the self-organisation of groups based on behaviour (cooperativeness) in closed societies, without any control at the top level.

One of the requirements of modern P2P systems is that they are open which allow agents to join and leave distributed societies. This issue forms the focus of the next chapter.

Chapter 7

Self-organisation in decentralised open systems

7.1 Introduction

The openness of the Internet allows users to dynamically join and leave the system at any point of time. So, a solution to the freeriding problem should take into account the open, dynamic, and distributed nature of modern software systems. P2P systems are increasingly attractive due to continually improving bandwidth and processing capabilities of distributed and mobile platforms.

The aim of the work discussed in this chapter was to develop a self-organising open and dynamic system, where new agents may come into the society and also agents may leave the society at any time. A truly open and dynamic system will allow the formation of new groups and dismantling of existing groups according to the population size. Our aim was to achieve that in a decentralised manner without explicit control at the top level. Forming groups using tags is helpful, since it is scalable and robust (Hales, 2004b). For higher numbers of peers, more tag groups can be formed, and that process will scale well for any number of peers (i.e. if there are more peers, more groups will be formed, and if there are fewer peers, fewer groups will be formed). Now, in the new arrangement, agents are set to have *lifespans*, which determines how long the agents remain in the society and when they leave (i.e. "die"). At any time a new agent could join the society and an existing agent could leave when its lifespan

is over. Thus we added openness to the society (agents can arrive and leave).

7.2 Open system

New unknown peers are allowed to join the society by gaining entry into the lowest ranked group. They can build their way up to higher groups, based on how well they perform in the eyes of their peers. This experiment has the same setup as the previous experiment in the previous chapter (Section 6.8). The additional agent attibute is *lifespan*.

Since the number of agents in the society at any time is dynamic, the system adapts itself to form new groups if more agents join. It also dismantles groups if there are fewer agents in the society (less than the lower size limit of a group).

The motivation for splitting and dismantling comes from real life societies. For example, when the size of a group becomes too large, it becomes unmanageable. Larger hunter-gatherer groups split because of reasons such as seasonal change or inequality in resource sharing (e.g. when food was not shared equally/due to disagreements) (Lee, 1972).

In our approach, a group splits into two if the size of the group reaches a certain limit (40 in our case). Based on the local gossip information in the splitting group, the top cooperators (first half) form one group and the rest (second half) form the other group. If the size of the group decreases and goes below a certain limit (5 in our case) then the group is dismantled. The remaining agents in the dismantled group go to the lowest scoring remaining group. This is similar to a society where it can be functional only if the society has a certain size. For example in hunter-gatherer societies, in order to hunt larger prey, a group has to have a minimum size. Otherwise, the prey cannot be hunted. The same holds in the context of playing a sport. For example, a team playing volleyball has to have six players. Otherwise, the team cannot exist.

It should be noted that the splitting and dismantling functionalities account for the scalability of the system and its robustness.

7.2.1 Results of experiments

A sample experimental run that demonstrates the self-organisation of groups in the open society is shown in Figure 7.1. A sample run for 5000 iterations is presented. Out of 5 initial groups, Group 4 dismantled. Group 3, which is the most cooperative group, split into two most cooperative groups, Group 6 and 7. Group 5 also split into two Groups 8 and 9, of which group 9 was the lowest group. Note that the new groups formed by splitting have some difference in their cooperativeness, since the most cooperative ones from the splitting group team up to form one group, and the rest form the other group. This is an ongoing process, because further groups will be formed and dismantled based on the arrival and leaving rates of the agents.

To test the scalability, we ran the experiment for 10000 iterations by having the initial population set to be 100 in 5 initial groups and also having hundreds of agents enter the scene randomly over subsequent periods. Agents also leave the society when their lifespan is over. This experiment has scaled well by forming 23 groups over 10000 iterations, in which 11 of them dismantled or split and the others were operational groups by the end of the iterations. It shows the system can scale well for a larger number of agents just by forming or dismantling groups dynamically. In open societies like these, agents cannot have a global view of all the groups. They have a limited view, which means they only know about the groups where the agents themselves and its group members have been before.

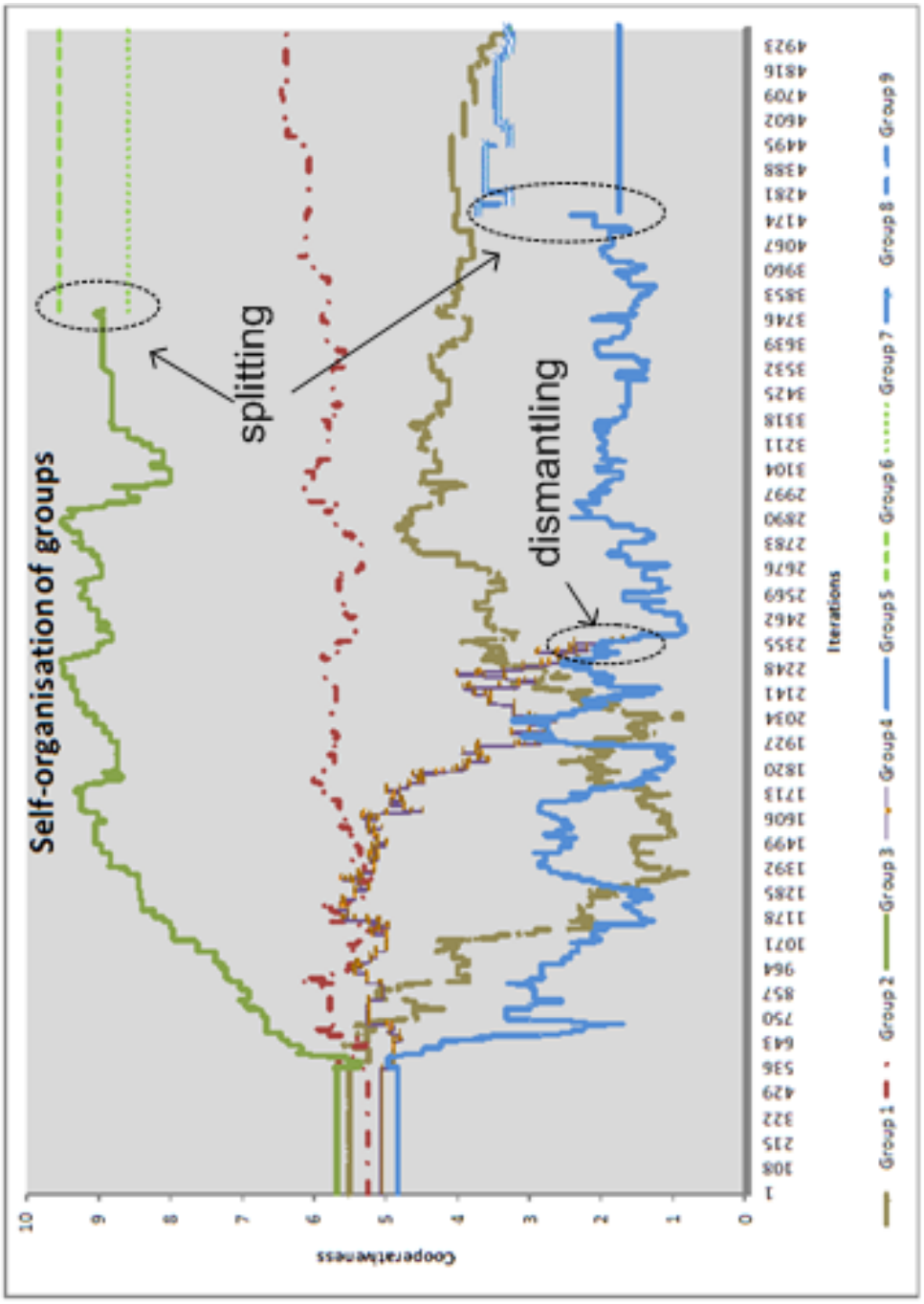

Figure 7.1: Self-organisation of groups based on cooperativeness in open society.

7.3 Enhanced open system

The experimental setup for the enhanced open systems was similar to the previous mechanism (Section 7.2), except for a few differences. The differences are explained in Section 7.4. The experimental parameters are listed in Table 7.1.

Experimental parameters	**Values**
Number of agents to start with	100
Number of groups to start with	5
Number of iterations	500
Agent's cooperative value	0-10 (random)
Agent's tolerance value	1-10 (random)
Agent's rejection limit	10
Agent's gossip blackboard length	10
Agent's group memory limit	5
Agent's lifespan	Varies
Number of gossip requests	5
Group's size for dismantling	5
Group's size for splitting	40
Cost for sharing	-0.1
Benefit for receiving	1

Table 7.1: Experimental parameters for open society.

The group splitting and dismantling processes are also the same as the previous experiment (Section 7.2). A group splits into two if the size of the group reaches a certain limit (40) and the group dismantles if the size of the group decreases and goes below a certain limit (5). In this experimental setup, the remaining agents in the group (from the dismantling group) go to random groups where they could enter.

7.3.1 Experiment 1 - Self-organisation in an open society

We have conducted experiments on an open system by varying the arrival and departure rate of the agents. Initially we started with 100 agents in 5 groups. After that agents could join (new arrivals) or leave (if lifespan is over) the society.

The two graphs together in Figure 7.2 show the dynamic behaviour of the system (the formation of new groups and dismantling of old groups). In Figure 7.2 there are two graphs which share the same x-axis. The x-axis shows the number of iterations. In the top graph the y-axis shows the cooperativeness of groups. Each diamond shown in the graph represents the cooperativeness of a particular group. For a given iteration number in the x-axis, the y-axis shows the cooperativeness of all the groups that were present in that iteration. For example, in iteration 100, there were 6 groups (represented by diamonds), with different levels of cooperativeness. The graph given in the bottom of Figure 7.2 shows the total number of agents (agents alive) in the society for a given iteration. For example, in iteration 100 there were 130 agents in the society.

It can be observed that at the start there were five groups and the groups had an average cooperativeness value of 5 approximately. As the number of agents increased, new groups were formed (see iteration 100). As the number of agents decreased (iteration 200), the number of groups decreased. The separation between the good (cooperative) groups and the bad (uncooperative) groups is distinct. When the total number of agents was about 40 in iteration 300, there were fewer groups. Note that the cooperativeness of these groups was about 5 at that point. As the number of agents in the societies then increased, there were more groups and the separation between the good and the bad groups is evident. We note this process can be appreciated better by viewing the video on UniTube (see (Savarimuthu, 2010d) for the link).

Figure 7.3 shows the graphical output of the result shown in Figure 7.2. The top left window shows the state of the system at iteration 1, the top right shows the snapshot of the system at iteration 100, bottom left shows iteration 400 and the bottom right shows iteration 500. The larger green circles represent groups. Agents are color coded based on their cooperativeness which are represented inside the larger green circle. The range of cooperativeness values and their color code are: 0-2 is Red, 3-4 is Orange, 5-6 is Yellow, 7-8

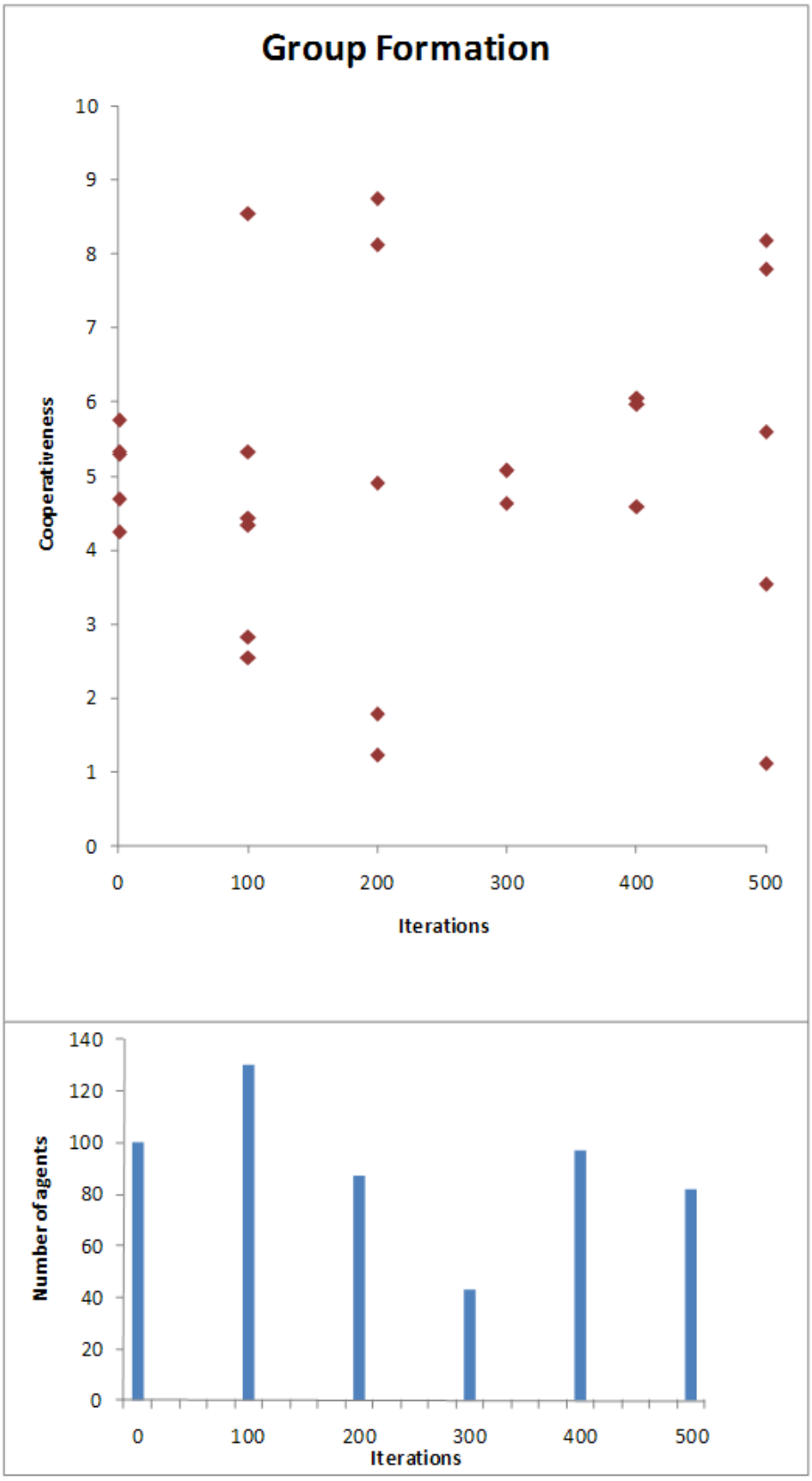

Figure 7.2: Self-organisation of an open system when agents' arrival and departure rates are dynamic.

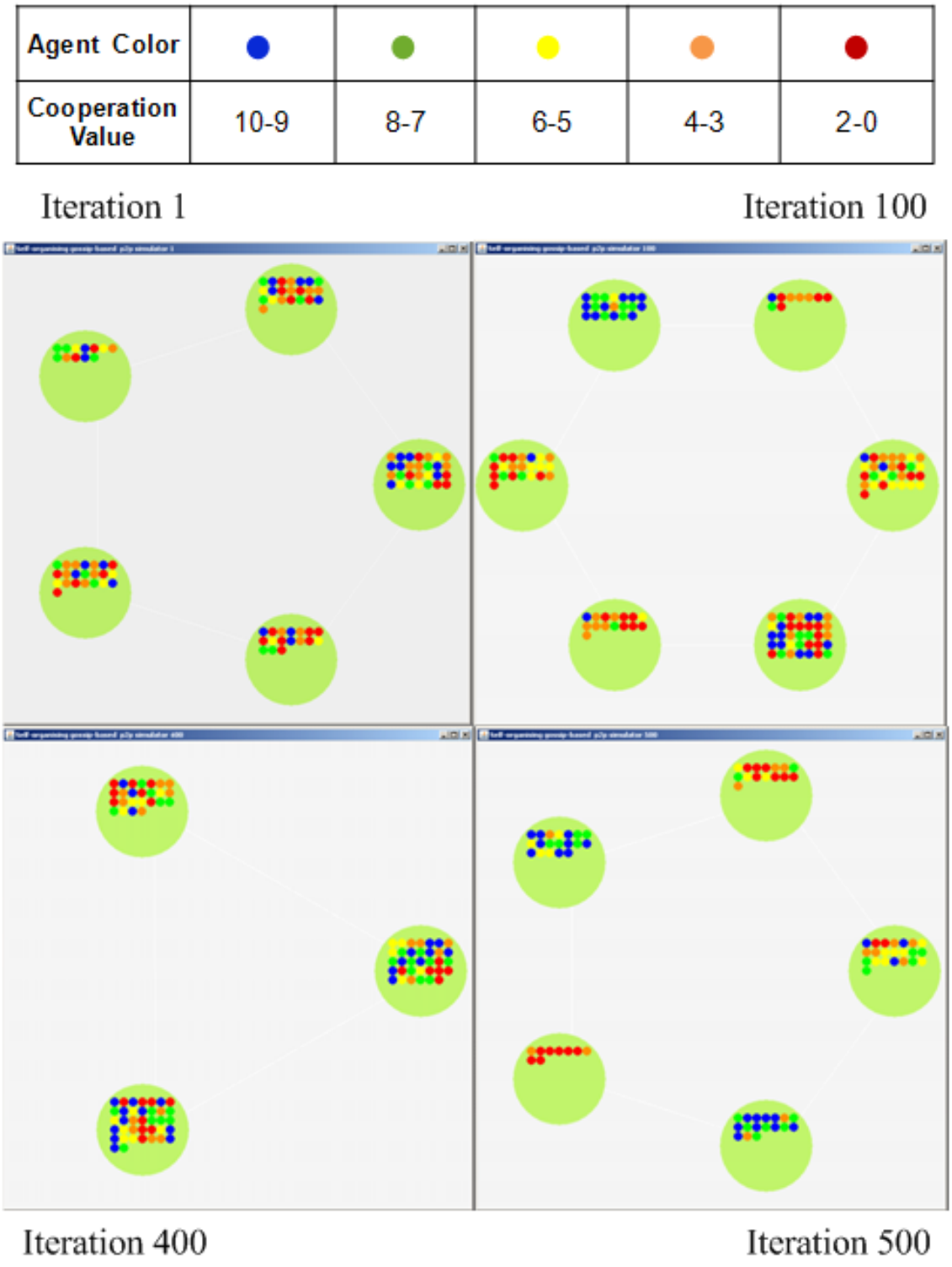

Figure 7.3: Snapshot of the groups at different iterations.

is Green and 9-10 is Blue. For example, in iteration 500 (bottom-right one of Figure 7.3), there are five groups. There are two good groups with a majority of blue and green agents. There are two other groups with mostly red agents and another group has agents with mixed colors (agents with different cooperation values).

There are two kinds of behaviour that can be observed in the system. Firstly, the system dynamically enlarges or shrinks by creating more groups or dismantling groups based on the number of agents in the system. Secondly, it also forms groups based on cooperativeness. Cooperators move towards other cooperators, and non-cooperators end up with other non-cooperators. The agents self-organise into groups that have different ranges of cooperativeness. Thus this system restricts the non-cooperators taking advantage of cooperators by restricting their access to better groups.

7.3.2 Experiment 2 - Arrival rate greater than departure rate

We conducted further experiments by setting the arrival rate greater than the departure rate. An example experimental run of this arrangement is shown in Figure 7.4. It can be observed that when the number of agents increased, the system was able to dynamically create more groups, and also these groups are separated based on the cooperativeness of the agents. This demonstrates the scalability of the system.

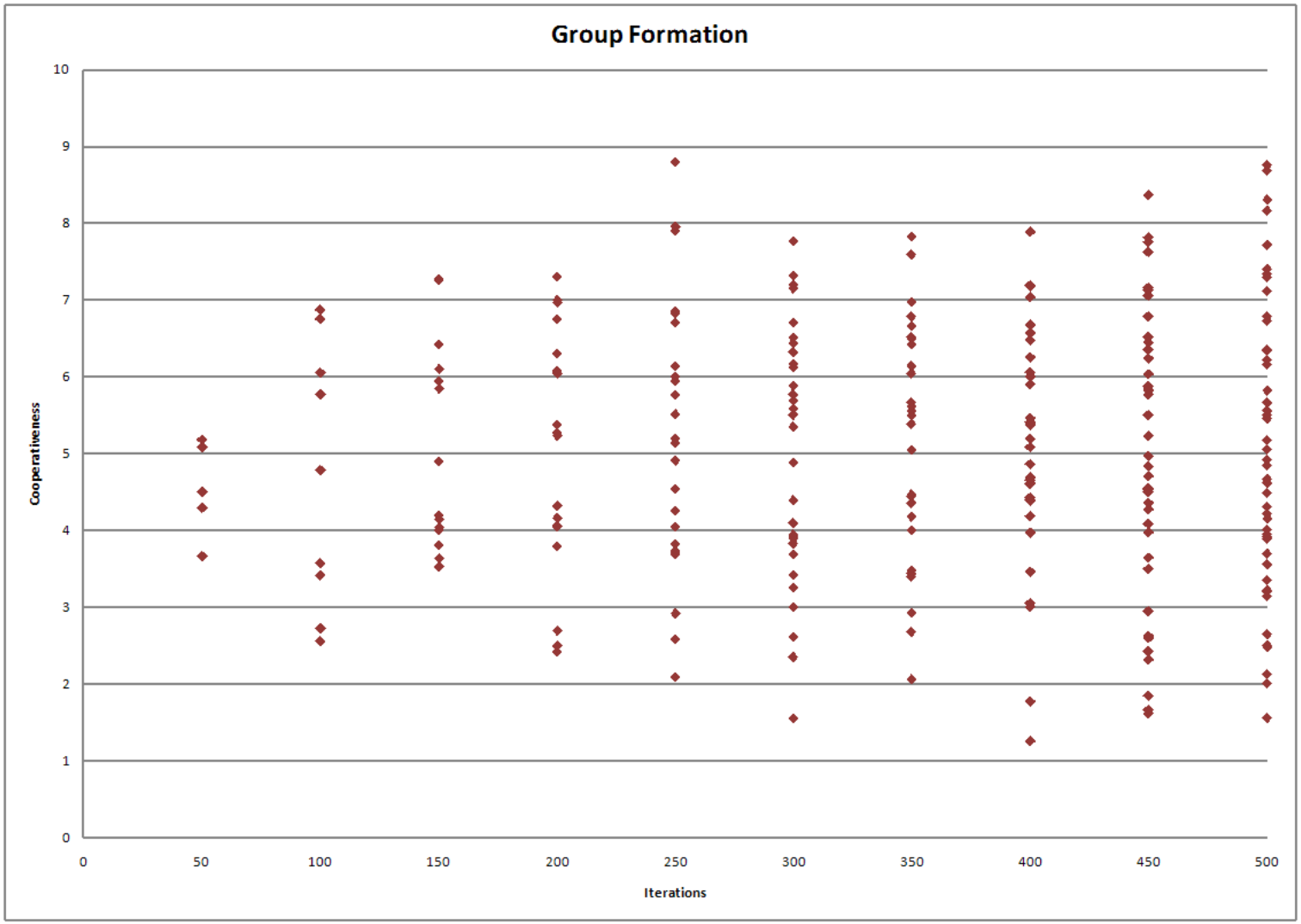

Figure 7.4: Self-organisation of an open system when agents' arrival rate is increased.

7.3.3 Experiment 3 - Arrival rate equal to departure rate

When the number of newcomers is roughly the same as the number of leaving agents in the system, the system will have same number of agents and the number of groups remains the same. However, new agents who join the society may have different levels of cooperativeness. Because of this the composition of groups and the cooperativeness of groups change over time. Figure 7.5 shows the cooperativeness of five different groups over 500 iterations. The cooperativeness of these groups varies depending upon the net effect of the cooperativeness of the agents that are present in the society. A new agent whose cooperativeness value of nine joining a group whose average cooperativeness value is five will increase the group's average. In the same way, a bad agent leaving a good group will increase the group's cooperativeness average. Figure 7.5 shows that there were only five groups all the time and how these five groups change over time based on the number of agents (composition of the group) and the cooperativeness of agents present in the system over time.

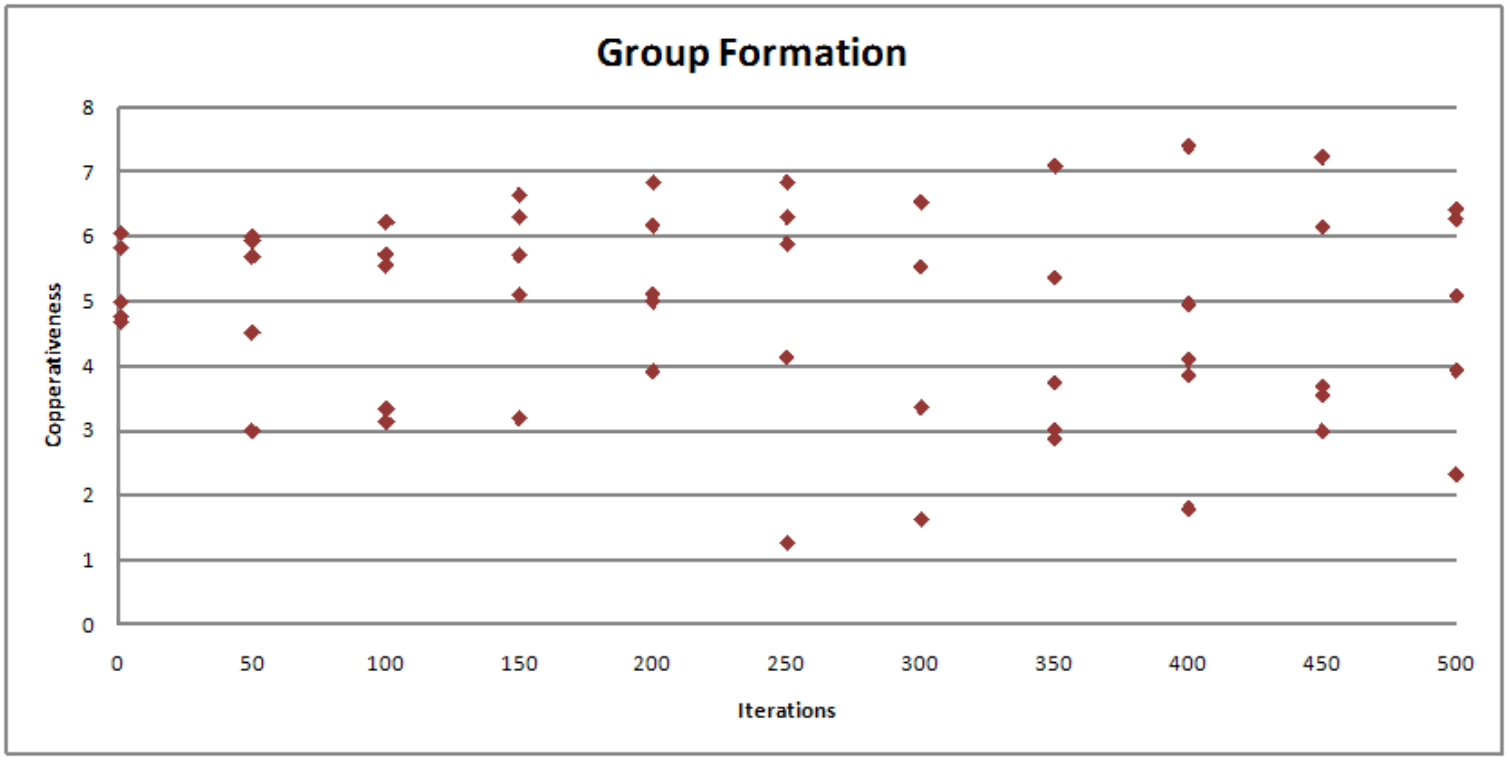

Figure 7.5: Self-organisation of an open system when the arrival rate is equal to the departure rate of the agents.

7.3.4 Experiment 4 - Varying lifespans of agents

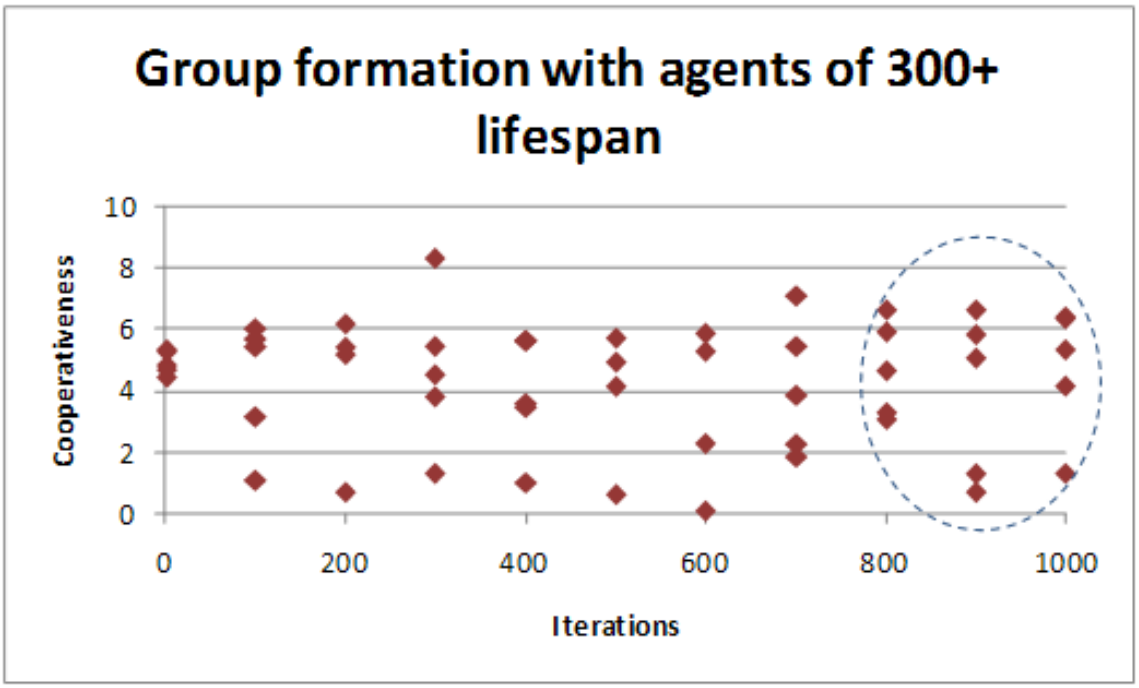

Figure 7.6: Group formation with minimum TTL = 300.

We varied the lifespan of the agents in order to investigate the impact of the lifespan of agents on the system's behaviour. We conducted two experiments by varying the lifespan. The lifespan of an agent is governed by the lifespan parameter. The minimum lifespan in one of the experiments was set to 300 and the other was set to 500. Figures 7.6 and 7.7 show the cooperativeness values of the groups for these two values of minimum lifespan for 1000 iterations. From these results it can be observed that having longer lifetime (agents being in the society for longer period of time) helps to achieve better segregation of groups. This is because when the lifespan is longer, agents have a longer period to gather and use gossip. Furthermore, when agents live for a shorter period of time, the system has a comparatively shorter period of time to segregate into groups than the system where the agents live longer. This can be observed by comparing the results for iterations 400 and 500. The separation of groups is better when *minimum lifespan*=500. The same can also be observed in the circled regions of these two figures. The videos of these simulations can be seen in these links on UniTube (see (Savarimuthu, 2010b,c)).

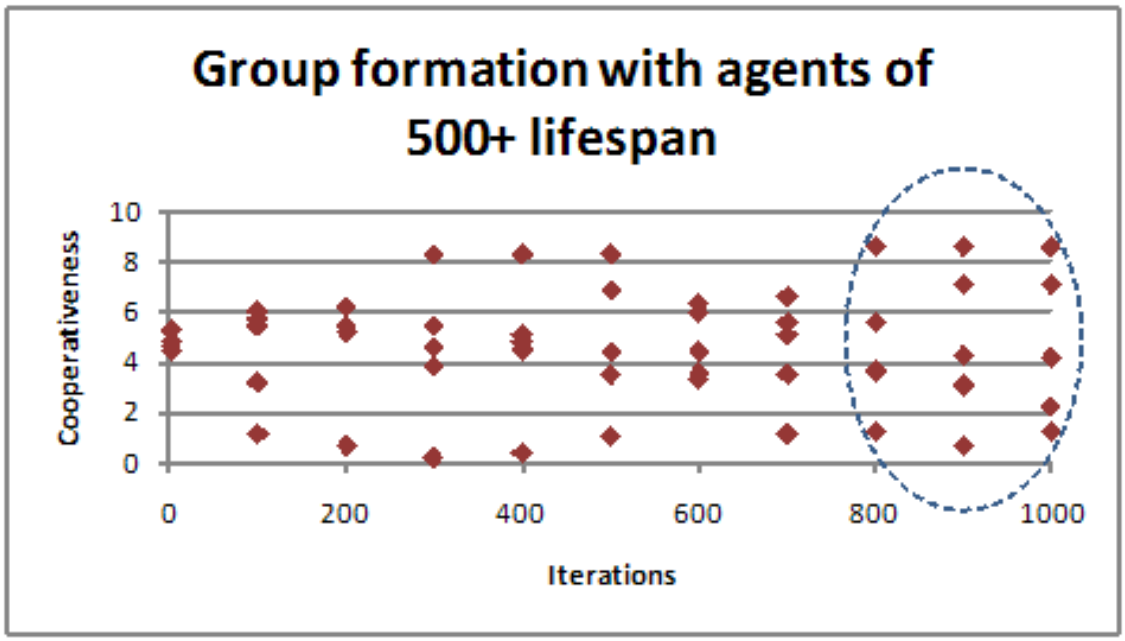

Figure 7.7: Group formation with minimum TTL = 500.

7.4 Differences

This mechanism presented in this chapter shows the self-organisation of groups using similar social mechanisms used in the previous chapter (Chapter 6). In this chapter, for enhanced open systems (Section 7.3), the mechanism presented in Section 7.2 (which has similar setup as the sharing group history mechanism (Section 6.8) with added openness) has been improved upon. The differences between this and the previous system (described in Section 7.2) are as follows.

- In the previous mechanism, the game was played for certain iterations (1/10th) and the gossip information was stored. Later the agents use the stored gossip information when they play. In the current setup the agents start using the gossip right from the start. If there is no information the agent is considered as a new player and allowed to play. As they play the gossip is also stored and used.

- In the previous mechanism wealth (score) has been taken into account. If the wealth of an agent has not increased over a certain period of iterations, then the agent decides to move. In the current setup, instead of wealth, if the rejection limit is met then the agent decides to move. We found that using a rejection limit works better for group

separation than basing the decision on wealth, since it is likely that the wealth will increase for a certain number of iterations (because the agents play with bad agents if the gossip information was not available, hence the wealth of the bad agents might increase).

- In the previous mechanism the remaining agents in a dismantling group go to the lowest performing group. In the current setup, they can apply to other groups and go to the group that accepts them. If they are not allowed then they keep trying to get entry into one of the groups.

- In the previous mechanism new players are introduced into the lowest group in the society and they are expected to build their way up to the higher groups based on their behaviour (cooperativeness). For that it was necessary to keep track of the lowest group of the system all the time, which is not a preferred practice if we want to achieve a decentralised open environment. In the current setup new agents go to random groups in the society. Because they are new, they have no past behaviour to track and they are allowed in to any group to which they seek entry. Eventually they will end up in an appropriate group based on their behaviour.

7.5 Summary

This chapter demonstrates how we have achieved self-organisation of groups in an open agent society. The mechanisms presented in this chapter facilitate splitting and dismantling of groups that lead to the formation and destruction of groups respectively. Additionally the groups are also formed with agents of similar cooperativeness. Thus systems using our mechanisms are scalable, adaptive, self-organising and decentralised. We have also investigated the effects of arrival rates of agents on the separation of groups based on their cooperativeness.

Using the gossip mechanism, agents share their interaction experience with some other agents. Unlike most reputation mechanisms, where agents keep track of all the reputation information of other agents, our system distributes a subset of this information among the individuals in the group. This mechanism operates and converges to satisfactory results in

the context of a partial-view of truth about the world states, which is a realistic and scalable feature of open agent societies. If any peer leaves a group, only a limited amount of information is lost, which ensures the robustness of the system. Peers might leave the society (due to bad behaviour) and try to re-enter again. But our system filters out peers in separate groups, based on their behaviour. It does not matter how many times an uncooperative agent would enter, it will inevitably end up in the worst group.

The system presented is suitable for sharing digital goods in open P2P systems where the aim is to restrict freeriding. It has produced self-organisation, or the so called self-balancing of P2P systems, in a distributed and dynamic manner in open agent societies.

Our system takes advantage of social mechanisms, such as tagging to form groups, gossip to pass information, and ostracism to shun bad behaviour. As a result, it shows the self-organisation of groups based on behaviour in open agent societies. These mechanisms can also be applied to virtual worlds and electronic online communities. For example, in digital environments such as SecondLife (2003), tagging can be used by avatars to organize themselves into groups and gossip can be used to pass information about one another, and ostracism can be used to shun bad behaviour.

The next part of the thesis presents the discussion and conclusion.

Part III - Discussion and Conclusion

Chapter 8

Discussion

The discussion in this chapter is arranged in two sections. First, the contributions of this thesis are summarised in Section 8.1. Second, the limitations and future research directions are discussed in Section 8.2.

8.1 Contributions of this thesis

The overarching theme of this thesis (Section 1.2) is to facilitate separation or segregation of groups in different types of artificial agent societies in order to ensure that exploitation is restricted and resources are provided in sufficient supply to cooperative members of the societies. A society is divided into several groups in order to address the problem of non-cooperation, where the groups are formed based on cooperativeness.

Towards achieving this goal three main aspects have been considered when modeling artificial agent societies. They are

- **Nature of societies**: Societies may either allow or disallow new agents. Two types of societies considered by this thesis are the *open* and *closed* agent societies. The most dynamic and responsive modern agent societies are open societies, and most work in multi-agent systems is towards facilitating open agent societies. However, closed societies also exist in domains where external agents are not typically allowed (e.g. robotic rescue scenarios), since trust is a key aspect for the system to operate (i.e. untrusted new agents may hamper rescue operations). Figure 8.1 shows the progression from

closed to open agent societies (same as Figure 2.3).

- **Type of control in societies**: Societies have monitoring and controlling mechanisms to enable their improvement. Towards this end, this thesis considers *centralised*, *semi-centralised* and *decentralised* mechanisms. While centralised control is possible in closed agent societies, there is a need to move towards decentralised societies, since contemporary agent societies are open and dynamic in nature (e.g massively multi-player games where avatars can be viewed as autonomous agents and electronic auctions where agents can simultaneously participate in several auctions). This thesis has considered all these three types of control in agent societies gradually progressing from centralised control to decentralised control from Chapters 4 to 7 (see Figure 8.1).

- **Social mechanisms for control**: The controls in agent societies have been facilitated through several social mechanisms inspired by human societies. These include *tagging* for group formation, *gossip* for spreading information about agent behaviour, *referral* and *voting* for recommending agents based on their past behaviour, and *resource restriction* and *ostracism* for controlling agents.

Figure 8.2 shows the social mechanisms and processes implemented in Chapters 4 to 7 (note that Figure 8.2 is the same as Figure 3.1).

The research presented in this thesis has resulted in eight peer-reviewed publications and are listed in Table 1.1.

8.1.1 Highlight overview of the chapters

This subsection provides a summary of the important aspects of the core experimental chapters of this thesis. Figure 8.3 highlights the mechanisms developed in each experimental chapter of this thesis .

- Chapter 4

  In this chapter it has been shown that tags can be used in a society of independent agents using a system-level control approach. This chapter identifies the condition under which tagging can help in sustaining the knowledge possessed within the society (e.g. the *winning condition* was identified).

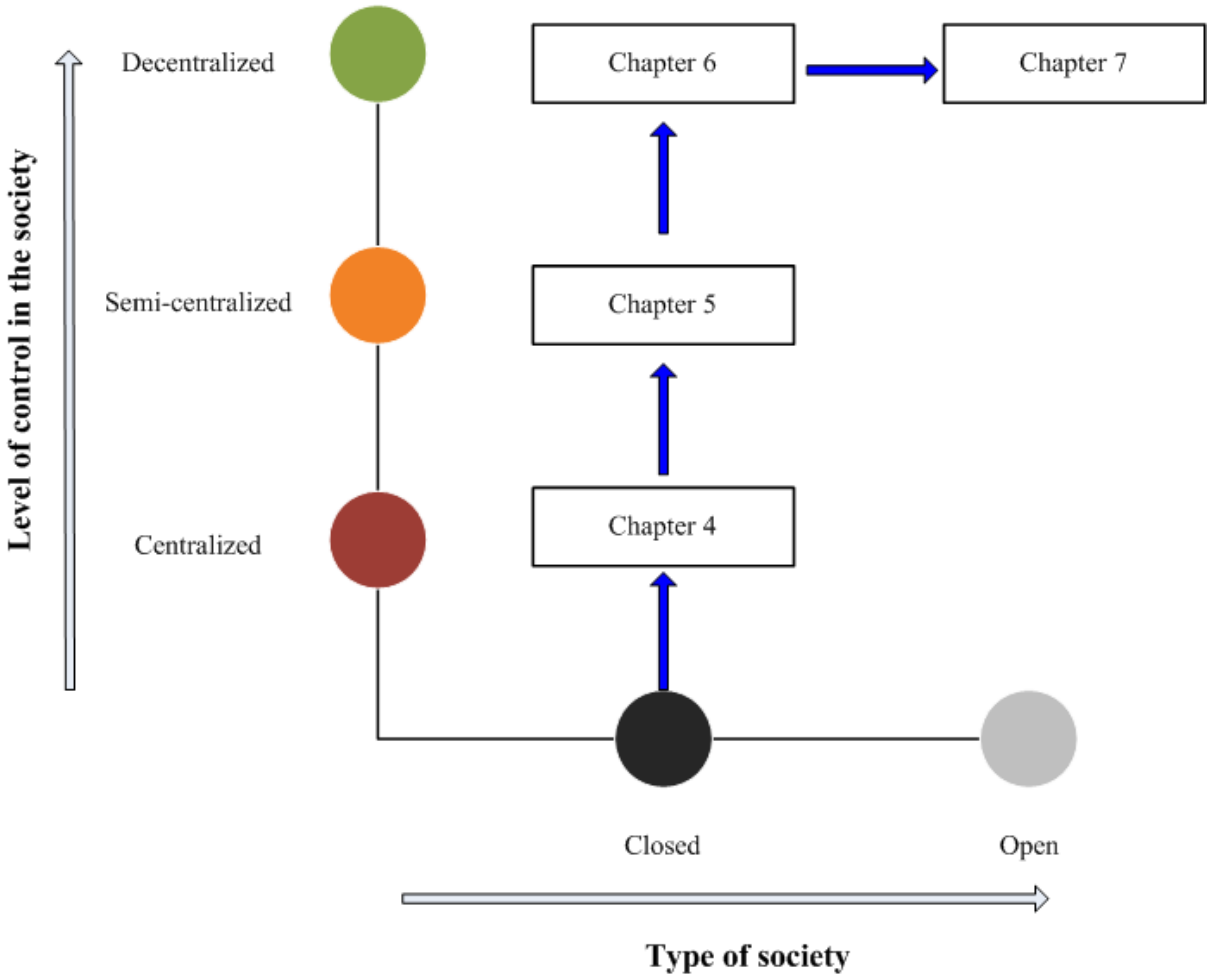

Figure 8.1: Progression of the experimental chapters.

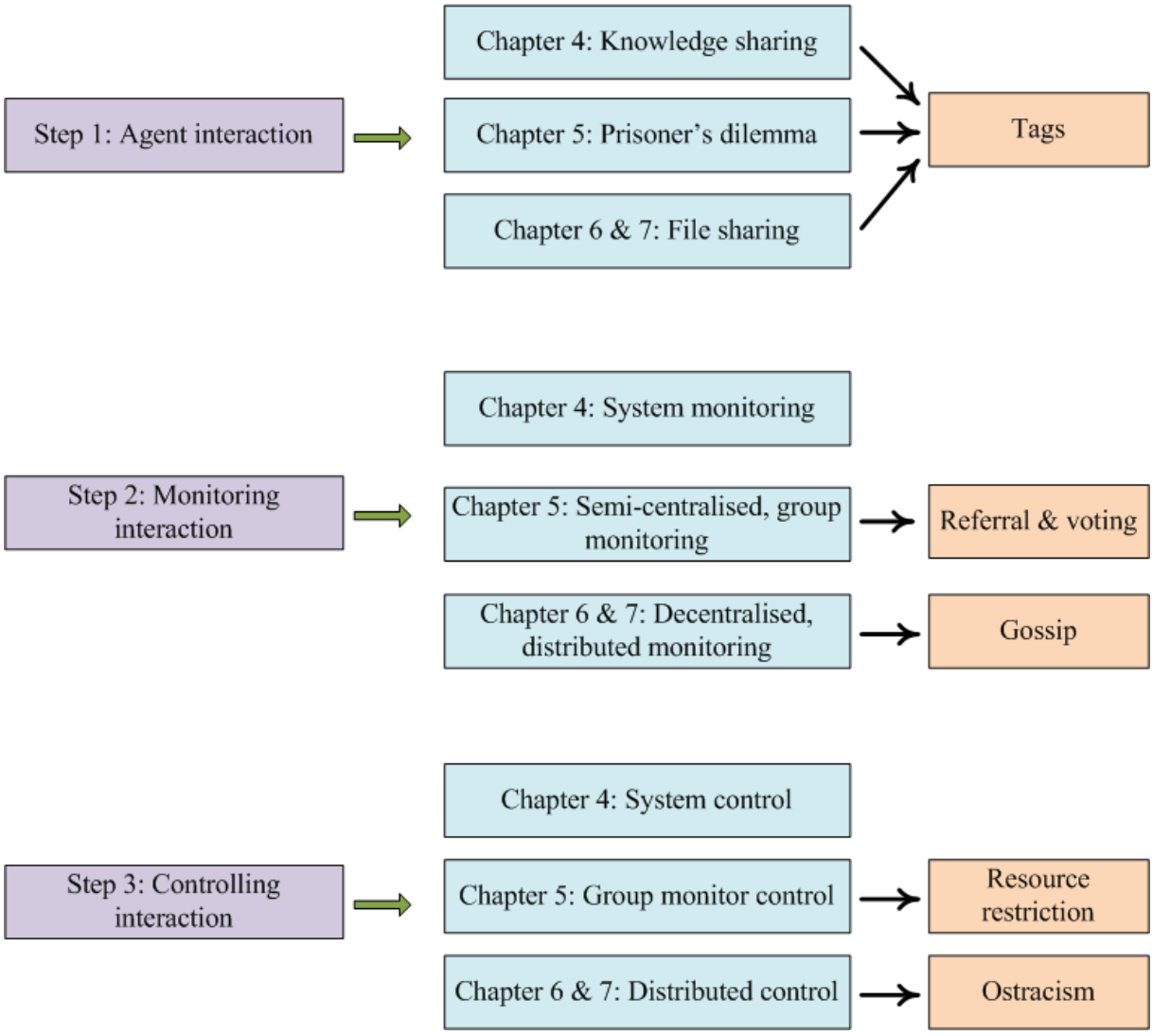

Figure 8.2: Processes employed in this thesis.

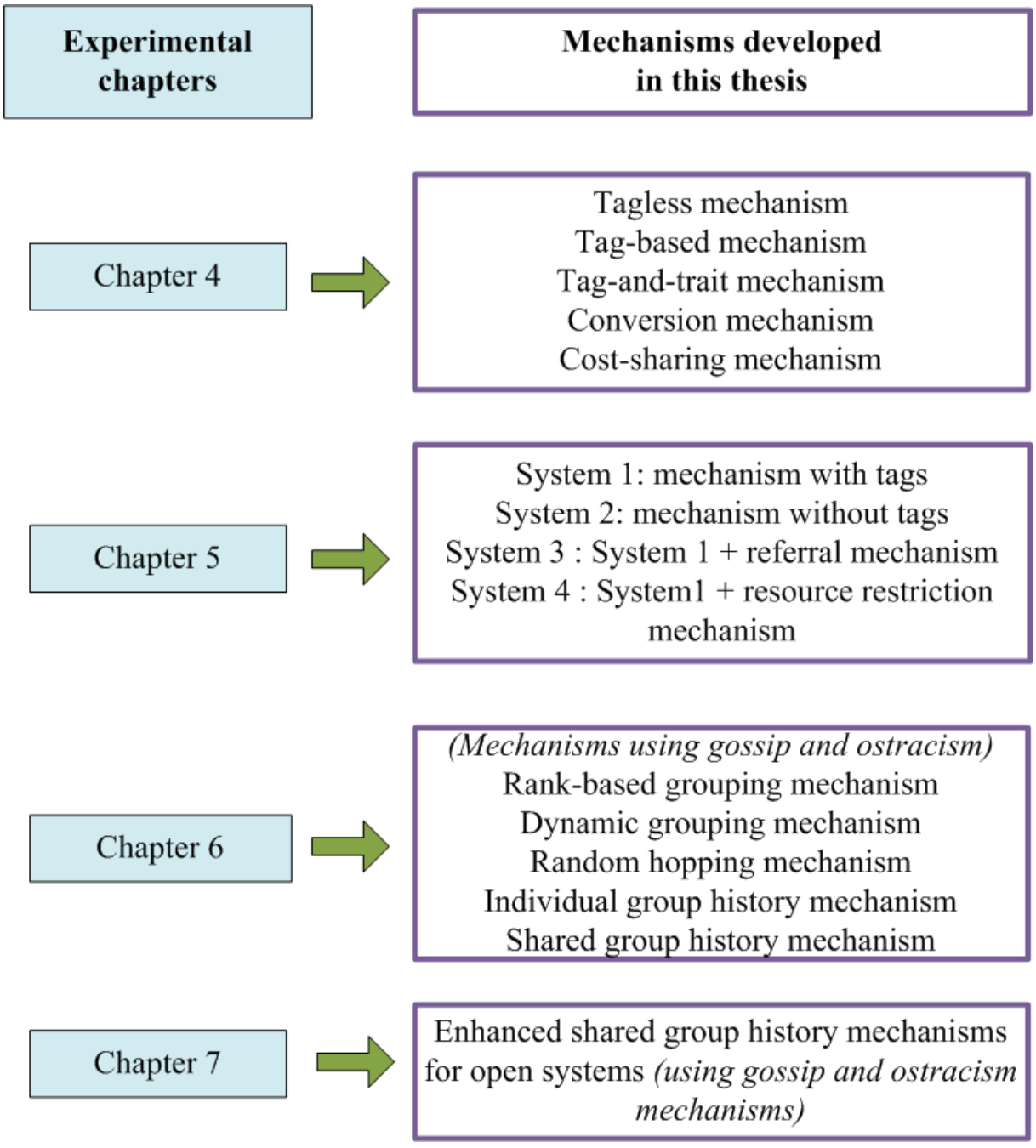

Figure 8.3: Mechanisms developed in this thesis.

- Chapter 5

  This chapter employed semi-centralised monitoring and controlling agents called *group monitors*. By using group monitors (semi-centralised control) this chapter discussed mechanisms that helped to avoid exploitation and improve overall societal performance in a multi agent society that had both cooperative and uncooperative agents. These are achieved through the segregation of groups based on cooperativeness of members in the society.

  Tagging was used for group formation, and referrals and voting were used for recommending agents. A resource restriction based mechanism was used to control the non-cooperating agents. It has been demonstrated that the proposed mechanisms help in improving the societal benefit.

- Chapter 6

  This chapter demonstrated how decentralised monitoring and controlling mechanisms can be used in closed agent societies. The primary mechanism investigated is the gossip-based mechanism for decentralised information spreading. The information collected using gossip can be used for making decisions, and agents can then use the information to decide whether to interact with other agents. These mechanisms were investigated in the context of sharing digital goods in a decentralised manner. Five different mechanisms were explored, and the results obtained were discussed.

- Chapter 7

  This chapter investigated the self-organisation of groups in an open system where agents can join and leave a society. This resulted in dynamic formation and the collapse of groups, which is necessary for dynamically changing systems. The system exhibited self-organising behaviour of groups based on cooperativeness. Additionally, this chapter also investigated the impact of arrival rates of agents on self-organisation. Similar to Chapter 6, social mechanisms such as tags, gossip, and ostracism were developed and implemented.

  This chapter thus describes how self-organisation of groups can be achieved in an open agent society that is scalable (i.e. the number of agents can be large), dynamic (i.e.

agents can come and go), and employs decentralised notions for monitoring and control (i.e. social mechanisms that are decentralised). These properties make the system suitable for deployment in contemporary electronic societies such as P2P societies.

8.2 Limitations and Future Work

The research area of cooperation in distributed societies of autonomous software agents is wide and there are many aspects that were outside the scope of the present work.

- **Consideration of agents changing behaviour:** In this thesis we have not considered agents changing their behaviour in their life time. In real life, bad agents may redeem themselves or may be forced to cooperate through institutional punishment mechanisms. In our future work, we intend to examine more advanced situations in which agents can dynamically alter their cooperation strategies. That would mean that a peer could start with a certain cooperative value, but later based on the circumstances (e.g. based on learning), decide to change its behaviour. For example, an agent could try to enhance its performance by becoming a "bad guy" temporarily and then returning to being a "good guy", since it may estimate that its potential rewards could be even higher because of occasional cheating in a good group of agents. Such behaviour changing mechanisms can be investigated in the future.

- **Lying problem:** The systems investigated in this thesis make use of recommendations from other agents to decide whether to interact with another agent or not and also to know the performances of other groups, relying on the fact that the other group agents are honest in revealing the information about their group (i.e. these agents do not lie). However, this may not be the case in general. Agents being autonomous (and intelligently self-interested) may not want to share their group information (e.g. cooperativeness of the group) with outsiders and, worse still, may lie when such information is requested. This behaviour may lead to an undesired state of affairs in a society. Additionally, bad agents can spread false gossip which may also have deleterious effect in segregation of groups. Our intention is to examine these issues in our future investigations. Another interesting avenue for research is to investigate additional mechanisms

for handling these types of misbehaviours in agent societies.

- **Varying gossip type:** In the future, we intend to examine the types of gossip in the system to determine conditions under which the gossip mechanism leads to separation of groups and conditions under which it does not lead to the separation (i.e. by experimenting with different types of gossip (e.g. about other groups, about other agents' cooperativeness, about other agents' trustworthiness in providing gossip information)).

- **Usage of tags:** Further investigations on tag-based mechanisms can be undertaken in the future. Tags cannot only be used as simple identifiers with a simple meaning attached to them but also to the extreme of more fully stereotyping agents in societies. Agents can be tagged based on their behaviour (e.g. associating 'red' colored agents as the non-cooperators). Agents can also tag other agents in multiple dimensions based on different behavioural attributes. We will consider these aspects for future work.

Chapter 9

Conclusion

Human societies have long developed and evolved social mechanisms for facilitating cooperation among individual members and subgroups. The advent of dynamic societies of the technological world, in particular the large and rapidly changing electronic societies, call for the use of new mechanisms to facilitate cooperation among artificial agents that can be modeled after those employed in human societies. One of the problems this thesis addresses is the problem of non-cooperation among agents commonly observed in the form of freeriding. This thesis has developed several solutions to this problem by facilitating the segregation of agents in a single society into several groups based on the cooperativeness of agents.

Towards facilitating cooperation in agent societies, new techniques developed in this thesis have addressed issues associated with several aspects of artificial societies including closed and open nature of societies and the types of control mechanisms such as centralised, semi-centralised, and decentralised mechanisms. This thesis investigates a tag-based approach for group formation, referral with voting and gossip-based approach for spreading information and resource restriction, ostracism for controlling agents in societies.

This thesis has developed new techniques that employ several social mechanisms that lead to grouping agents based on their cooperativeness. These mechanisms help in the separation of better performing groups from not-so-good groups. This also reduces the likelihood of bad agents exploiting the good agents in the better groups. It has been demonstrated that the performance of the groups is better when the social mechanisms developed here are adopted in an agent society.

This thesis has systematically developed a range of social cooperation mechanisms that can be applied to a spectrum of artificial agent societies with varying demands placed on them in connection with efficiency and control. These societies under study have ranges from closed, more tightly controlled agent groupings all the way to fully open systems that may contain vast number of agents and cannot accommodate the inefficiencies of centralised control in order to scale up appropriately in size. This issue of scalability is crucial and has been emphasised in this thesis so that social systems of autonomous software agents can accommodate growing populations of agents and still be responsive to dynamic conditions in their environment (including the allowing of new agents to enter and leave the society).

We note that, since not all societies are open and decentralised, the range of solutions investigated in this thesis can be applied in situations that warrant them (i.e. based on the nature of the societies that are investigated). We have also discussed the future extensions to the work carried out in this thesis.

Overall, this thesis offers new socially-inspired mechanisms that offer solutions towards restricting exploitation of freeriders in artificial societies through the segregation of groups. These mechanisms result in the improvement of the overall societal performance in a society that has both cooperative and uncooperative agents. We believe some of the mechanisms developed here in this thesis can be applied effectively to future ICT systems in order to make them more responsive to the likely vicissitudes of future conditions.

References

Adar, E. and Huberman, B. A. (2000). Free Riding on Gnutella. *First Monday*, *5*(10).

Aldewereld, H., Vázquez-Salceda, J., Dignum, F., and Meyer, J.-J. (2006). Verifying Norm Compliancy of Protocols. In O. Boissier, J. Padget, V. Dignum, G. Lindemann, E. Matson, S. Ossowski, J. Sichman, and J. Vzquez-Salceda (Eds.), *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems, AAMAS 2005 International Workshops on Agents, Norms and Institutions for Regulated Multi-Agent Systems, ANIREM 2005, and Organizations in Multi-Agent Systems, OOOP 2005, Utrecht, The Netherlands, July 25-26, 2005, Revised Selected Papers*, Volume 3913 of *Lecture Notes in Computer Science*, 231–245. Springer Berlin / Heidelberg.

Arcos, J. L., Esteva, M., Noriega, P., A.Rodríguez-Aguilar, J., and Sierra, C. (2005). Engineering open environments with electronic institutions. *Engineering Applications of Artificial Intelligence*, *18*(2), 191–204.

Arkin, R. C. and Hobbs, J. D. (1993). Dimensions of communication and social organization in multi-agent robotic systems. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson (Eds.), *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, Cambridge, MA, USA, 486–493. MIT Press.

Artikis, A. and Pitt, J. (2001). A formal model of open agent societies. In *Proceedings of the fifth international conference on Autonomous agents*, AGENTS '01, New York, NY, USA, 192–193. ACM.

Axelrod, R. (1984). *The Evolution of Cooperation*. NY: Basic Books.

Axelrod, R. (1986). An Evolutionary Approach to Norms. *The American Political Science Review*, *80*(4), 1095–1111.

Axelrod, R., Hammond, A. R., and Grafen, A. (2004). Altruism via Kin-Selection Strategies that Rely on Arbitrary Tags with which They Coelvolve. In *Evolution*, Volume 58, 1833–1838.

Babaoglu, O., Meling, H., and Montresor, A. (2002). Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, ICDCS '02, Washington, DC, USA, 15–22. IEEE Computer Society.

Bak, P. (1996). *How Nature Works: The Science of Self-Organized Criticality*. New York: Copernicus.

Balaji, P. G. and Srinivasan, D. (2010). An Introduction to Multi-Agent Systems. In D. Srinivasan and L. Jain (Eds.), *Innovations in Multi-Agent Systems and Applications - 1*, Volume 310 of *Studies in Computational Intelligence*, 1–27. Springer Berlin / Heidelberg. 10.1007/978-3-642-14435-6-1.

Batson, D. (1991). *The Altruism Question: Toward a Social-Psychological Answer*. Lawrence Erlbaum Associates.

Batson, D. (1998). Altruism and prosocial behavior. In G. Lindzey (Ed.), *The Handbook of Social Psychology* (4 ed.)., New York, 282–316. McGraw-Hill.

Bezos, J. (1995). Amazon homepage. `http://amazon.com`.

Bias, R. (2009). Up, Out, Centralized, and Decentralized. `http://cloudscaling.com/blog/cloud-computing/up-out-centralized-and-decentralized`.

Boella, G., van der Torre, L., and Verhagen, H. (2006). Introduction to normative multiagent systems. *Computational and Mathematical Organization Theory*, *12*, 71–79. 10.1007/s10588-006-9537-7.

Boissier, O., Padget, J. A., Dignum, V., Lindemann, G., Matson, E. T., Ossowski, S., Sichman, J. S., and Vázquez-Salceda, J. (Eds.) (2006). *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems, AAMAS 2005 International Workshops on Agents, Norms and Institutions for Regulated Multi-Agent Systems, ANIREM 2005, and Organizations in Multi-Agent Systems, OOOP 2005, Utrecht, The Netherlands, July 25-26, 2005, Revised Selected Papers*, Volume 3913 of *Lecture Notes in Computer Science*. Springer.

Bondi, A. B. (2000). Characteristics of scalability and their impact on performance. In *Proceedings of the 2nd international workshop on Software and performance*, WOSP '00, New York, NY, USA, 195–203. ACM.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2006). Randomized gossip algorithms. *IEEE Transactions on Information Theory*, *52*(6), 2508–2530.

Brabham, D. C. (2008). Crowdsourcing as a Model for Problem Solving: An Introduction and Cases. *Convergence: The International Journal of Research into New Media Technologies*, *14*(1), 75–90.

Bradshaw, J. M. (Ed.) (1997). *Software agents*. Cambridge, MA, USA: MIT Press.

Bulpitt, P. (1994). World of Warcraft. `http://www.battle.net/wow/`.

Cakar, E. and Muller-Schloer, C. (2010). Decentralised and Adaptive Collaboration in Multi-agent Systems. In *Proceedings of the 2010 Ninth International Symposium on Parallel and Distributed Computing*, ISPDC '10, Washington, DC, USA, 195–202. IEEE Computer Society.

Candale, T. and Sen, S. (2005). Effect of referrals on convergence to satisficing distributions. In *Proceedings of the fourth international joint conference on Autonomous Agents and MultiAgent Systems, (AAMAS'05)*, New York, NY, USA, 347–354. ACM.

Castelfranchi, C. (2000). Engineering Social Order. In *Proceedings of the First International Workshop on Engineering Societies in the Agent World: Revised Papers*, ESAW '00, London, UK, 1–18. Springer-Verlag.

Chan, H. K. and Chan, F. T. S. (2010). Comparative study of adaptability and flexibility in distributed manufacturing supply chains. *Decision Support Systems*, *48*, 331–341.

Chao, I., Ardaiz, O., and Sangüesa, R. (2007). Tag Mechanisms Evaluated for Coordination in Open Multi-Agent Systems. In A. Artikis, G. M. P. O'Hare, K. Stathis, and G. A. Vouros (Eds.), *ESAW*, Volume 4995 of *Lecture Notes in Computer Science*, 254–269. Springer.

Clutton-Brock, T. H. and Parker, G. A. (1995). Punishment in animal societies. *Nature*, *373*(6511), 209–216.

Cohen, B. (2004). BitTorrent homepage. `http://www.BitTorrent.com`.

Colombetti, M., Fornara, N., and Verdiccchio, M. (2002). The Role of Institutions in Multiagent Systems. In *Proceedings of the workshop on knowledge based and reasoning agents, VIII Convegno AI*IA 2002*, 118–227.

Conte, R. (2001). *Social Order in Multiagent Systems*. Norwell, MA, USA: Kluwer Academic Publishers.

Conte, R. and Paolucci, M. (2002). *Reputation in artificial societies: social beliefs for social order*. Norwell, MA, USA: Kluwer Academic Publishers.

Dasgupta, P. (2003). Incentive Driven Node Discovery in a P2P Network Using Mobile Intelligent Agents. In H. R. Arabnia, R. Joshua, and Y. Mun (Eds.), *Proceedings of the International Conference on Artificial Intelligence, IC-AI '03, June 23 - 26, 2003, Las Vegas, Nevada, USA, Volume 2*, 750–756. CSREA Press.

Davidsson, P. (2001). Categories of Artificial Societies. In *Engineering Societies in the Agents World II, volume 2203 of LNAI*, 1–9. Springer-Verlag.

Davidsson, P. (2002). Agent Based Social Simulation: A Computer Science View. *Journal of Artificial Societies and Social Simulation, http://jasss.soc.surrey.ac.uk/5/1/7.html*, *5*(1).

Davis, R. (1980). Report On The Workshop On Distibuted AI. Technical report, SIGART Newsletter.

Davis, R. (1982). Report on the Second Workshop on Distributed AI. Technical report, MIT Artificial Intelligence Laboratory Working Papers, WP-228.

Dawes, R. M. (1980). Social dilemmas. *Annual Review of Psychology*, *31*, 169–193.

de Pinninck, A. P., Sierra, C., and Schorlemmer, W. M. (2008). Distributed Norm Enforcement: Ostracism in Open Multi-Agent Systems. In P. Casanovas, G. Sartor, N. Casellas, and R. Rubino (Eds.), *Computable Models of the Law, Languages, Dialogues, Games, Ontologies*, Volume 4884 of *Lecture Notes in Computer Science*, 275–290. Springer.

Dellarocas, C. (2000). Contractual Agent Societies: Negotiated shared context and social control in open multi-agent systems. In *Social Order in Multi-Agent Systems*, Norwell, MA, USA, 113–133. Kluwer Academic Publishers.

Dellarocas, C. and Klein, M. (1999). Civil Agent Societies: Tools for Inventing Open Agent-Mediated Electronic Marketplaces. In *Agent Mediated Electronic Commerce (IJCAI Workshop)'99*, 24–39. Springer.

Di Marzo Serugendo, G., Gleizes, M.-P., and Karageorgos, A. (2005). Self-organization in multi-agent systems. *Knowledge Engineering Review*, *20*, 165–189.

Doran, J. and Palmer, M. (1995). *The EOS project: Integrating two models of palaeolithic social change*, 103–125. UCL Press.

Dugatkin, L. A. (1997). Cooperation among animals: an evolutionary perspective. Technical report, Oxford University Press, Oxford, UK.

Dunbar, R. (1996). *Grooming, Gossip and the Evolution of Language*. London: Faber and Faber.

Dunbar, R. (2004). Gossip in Evolutionary Perspective. *Review of General Psychology*, *8*(2), 100 – 110.

Durfee, E. (2006). Distributed Problem Solving and Planning. In M. Luck, V. Mark, O. Łtepnkov, and R. Trappl (Eds.), *Multi-Agent Systems and Applications*, Volume 2086 of *Lecture Notes in Computer Science*, 118–149. Springer Berlin / Heidelberg. 10.1007/3-540-47745-4-6.

Eiben, A. E. and Smith, J. E. (2008). *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer.

Epstein, J. M. and Axtell, R. (1996). *Growing artificial societies: social science from the bottom up*. Washington, DC, USA: The Brookings Institution.

Esteva, M., Rosell, B., Rodriguez-Aguilar, J. A., and Arcos, J. L. (2004). AMELI: An Agent-Based Middleware for Electronic Institutions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '04, Washington, DC, USA, 236–243. IEEE Computer Society.

Eugster, P., Felber, P., and Le Fessant, F. (2007). The "art" of programming gossip-based systems. *Operating Systems Review-Special topic: Gossip-Based Networking*, *41*(5), 37–42.

Fanning, S. and Fanning, J. (1999). Napster homepage. `http://www.napster.com`.

Fehr, E. and Fischbacher, U. (2003). The nature of human altruism. *Nature*, *425*(6960), 785–791.

Fehr, E. and Rockenbach, B. (2003). Detrimental Effects of Sanctions on Human Altruism. *Nature*, *422*, 137–140.

Feldman, M. and Chuang, J. (2005). Overcoming free-riding behavior in peer-to-peer systems. *ACM Sigecom Exchanges*, *5*, 41–50.

Ferber, J. (1999). *Multi-agent systems - an introduction to distributed artificial intelligence*. Addison-Wesley-Longman.

Ganesh, A. J., Kermarrec, A.-M., and Massoulié, L. (2003). Peer-to-Peer Membership Management for Gossip-Based Protocols. *IEEE Transactions on Computers*, *52*, 139–149.

Gilbert, N. and Conte, R. (1995). *Artificial societies: the computer simulation of social life*. UCL Press.

Gilbert, N. and Troitzsch, K. G. (1999). *Simulation for the social scientist*. Open University Press.

Gorodetsky, V., Karsaev, O., Samoylov, V., and Serebryakov, S. (2007). Multi-agent Peer-to-Peer Intrusion Detection. In V. Gorodetsky, I. Kotenko, and V. A. Skormin (Eds.), *Computer Network Security*, Volume 1 of *Communications in Computer and Information Science*, 260–271. Springer Berlin Heidelberg. 10.1007/978-3-540-73986-9-23.

Grizard, A., Vercouter, L., Stratulat, T., and Muller, G. (2007). Coordination, Organizations, Institutions, and Norms in Agent Systems II. Chapter A Peer-to-Peer Normative System to Achieve Social Order, 274–289. Berlin, Heidelberg: Springer-Verlag.

Gursel, A., Sen, S., and Candale, T. (2009). Stability in referral systems. *Multiagent and Grid Systems*, *5*(1), 19–36.

hadouaj, S. E. and Drogoul, A. (2004). A Study of Coordination within a Road Traffic Environment. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), 20-24 September 2004, Beijing, China*, 491–495. IEEE Computer Society.

Hales, D. (2000). *Tag Based Co-operation in Artificial Societies*. Ph. D. thesis, University of Essex, UK.

Hales, D. (2001). Cooperation without memory or space: tags, groups and the prisoner's dilemma. In *Multi-Agent-Based Simulation, Second International Workshop, MABS 2000, Boston, MA, USA, July, 2000, Revised and Additional Papers*, Secaucus, NJ, USA, 157–166. Springer-Verlag New York, Inc.

Hales, D. (2002). Evolving Specialisation, Altruism, and Group-Level Optimisation Using Tags. In J. S. Sichman, F. Bousquet, and P. Davidsson (Eds.), *Multi-Agent-Based Simulation, Third International Workshop, MABS 2002, Bologna, Italy, July 15-16, 2002, Revised Papers*, Volume 2581 of *Lecture Notes in Computer Science*, 26–35. Springer-Verlag Berlin.

Hales, D. (2004a). Change Your Tags Fast! - A Necessary Condition for Cooperation? In P. Davidsson, B. Logan, and K. Takadama (Eds.), *Multi-Agent and Multi-Agent-Based Simulation, Joint Workshop MABS 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers*, Volume 3415 of *Lecture Notes in Computer Science*, 89–98. Springer.

Hales, D. (2004b). Self-Organising, Open and Cooperative P2P Societies - From Tags to Networks. In S. Brueckner, G. D. M. Serugendo, A. Karageorgos, and R. Nagpal (Eds.), *Engineering Self-Organising Systems*, Volume 3464 of *Lecture Notes in Computer Science*, 123–137. Springer.

Hales, D. (2008). Understanding tag systems by comparing tag models. In C. H. I. Bruce Edmonds and K. G. Troitzsch (Eds.), *Social Simulation: Technologies, Advances and New Discoveries*, 68–80. IGI Global.

Hales, D. and Edmonds, B. (2003a). Can Tags Build Working Systems? From MABS to ESOA. In G. D. M. Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli (Eds.), *Engineering Self-Organising Systems*, Volume 2977 of *Lecture Notes in Computer Science*, 186–194. Springer.

Hales, D. and Edmonds, B. (2003b). Evolving social rationality for MAS using "tags". In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, New York, NY, USA, 497–503. ACM.

Hales, D. and Patarin, S. (2005). Computational Sociology for Systems "In the Wild": The Case of BitTorrent. *IEEE Distributed Systems Online*, *6*(7), 2.

Hamilton, W. D. (1964a). The genetical evolution of social behaviour. I. *Journal of Theoretical Biology*, *7*(1), 1–16.

Hamilton, W. D. (1964b). The genetical evolution of social behaviour. II. *Journal of Theoretical Biology*, *7*(1), 17–52.

Hardin, G. (1968). The Tragedy of the Commons. *Science*, *162*, 1243–1248.

Helbing, D., Farkas, I., and Vicsek, T. (2000). Simulating Dynamical Features of Escape Panic. *Nature*, *407*(6803), 487–490.

Hirshleifer, D. and Rasmusen, E. (1989). COOPERATION IN A REPEATED PRISONERS' DILEMMA WITH OSTRACISM. *Journal of Economic Behavior and Organization*, *12*(1), 87106.

Holland, J. H. (1992). *Adaptation in natural and artificial systems*. Cambridge, MA, USA: MIT Press.

Holland, J. H. (1993). The effect of labels (tags) on social interactions. SFI Working Paper 93-10-064, 1399 Hyde Park Road, Santa Fe, New Mexico 87501, USA.

Jackson, M. (2011). BitTorrent P2P File-sharing Dominates EU Broadband ISP Internet Traffic. http://www.ispreview.co.uk/story/2011/05/18/bittorrent-p2p-filesharing-dominates-eu-broadband-isp-internet-traffic.html. Published on 18 May 2011 in ISPreview, United Kingdom.

Jelasity, M., Montresor, A., and Babaoglu, O. (2004). Detection and removal of malicious peers in gossip-based protocols. In *Proceedings of second Bertinoro Workshop on Future Directions in Distributed Computing: Survivability: Obstacles and Solutions (FuDiCo II: S.O.S.), Bertinoro, Italy, June. http://www.cs.utexas.edu/users/lorenzo/sos/*.

Jelasity, M., Montresor, A., and Babaoglu, O. (2005). Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, *23*, 219–252.

John F. Nash, J. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, *36*(1), 48–49.

Kempe, D., Dobra, A., and Gehrke, J. (2003). Gossip-Based Computation of Aggregate Information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, Washington, DC, USA, 482–491. IEEE Computer Society.

Khan, S. K. A. and Tokarchuk, L. N. (2009). Interest-based self organization in group-structured P2P networks. In *Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference*, CCNC'09, Piscataway, NJ, USA, 1237–1241. IEEE Press.

Kollock, P. (1998). Social Dilemmas: The Anatomy of Cooperation. *Annual Review of Sociology*, *24*, 183–214.

Krishnan, R., Smith, M. D., Tang, Z., and Telang, R. (2004). The Impact of Free-Riding on Peer-to-Peer Networks. In *Proceedings of the Proceedings of the 37th Annual Hawaii*

International Conference on System Sciences (HICSS'04) - Track 7 - Volume 7, HICSS '04, Washington, DC, USA, 199–208. IEEE Computer Society.

Lee, R. B. (1972). Work Effort, Group Structure and Land-use in Contemporary Hunter-gatherers. 177–185.

Lehmann, L. and Keller, L. (2006). The evolution of cooperation and altruism ; a general framework and a classification of models. *Journal of Evolutionary Biology*, *19*(5), 1365–1376.

Lesser, V. R. and Corkill, D. D. (1983). The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. *AI Magazine*, *4*(3), 15–33.

Li, J. (2008). A Survey of Peer-to-Peer Network Security Issues. `http://www.cs.wustl.edu/~jain/cse571-07/ftp/p2p.pdf`.

Macy, M. W. and Willer, R. (2002). From Factors to Actors: Computational Sociology and Agent-Based Modeling. *Annual Review of Sociology*, *28*, pp. 143–166.

Martinovic, I., Leng, C., Zdarsky, F. A., Mauthe, A., Steinmetz, R., and Schmitt, J. B. (2006). Self-protection in P2P Networks: Choosing the Right Neighbourhood. In *Proceedings of Self-Organizing Systems, First International Workshop, IWSOS 2006, and Third International Workshop on New Trends in Network Architectures and Services, IWSOS/EuroNGI 2006, Passau, Germany, September 18-20, 2006*, Volume 4124 of *Lecture Notes in Computer Science*, 23–33. Springer.

McDonald, A. and Sen, S. (2005). The Success and Failure of Tag-Mediated Evolution of Cooperation. In K. Tuyls, P. J. Hoen, K. Verbeeck, and S. Sen (Eds.), *LAMAS*, Volume 3898 of *Lecture Notes in Computer Science*, 155–164. Springer.

Milinski, M., Semmann, D., and Krambeck, H.-J. (2002). Reputation helps solve the 'tragedy of the commons'. *Nature*, *415*(6870), 424–426.

Minar, N. (2002). Distributed Systems Topologies: Part 2. `http://openp2p.com/pub/a/p2p/2002/01/08/p2p_topologies_pt2.html`.

Mordacchini, M., Baraglia, R., Dazzi, P., and Ricci, L. (2010). A P2P REcommender System based on Gossip Overlays (PREGO). In *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology*, CIT '10, Washington, DC, USA, 83–90. IEEE Computer Society.

Morgan, S. (1999). Trademe homepage. `http://trademe.co.nz`.

Moya, J. (2010). Free Music Downloads on Googles Android Phones. `http://www.zeropaid.com/news/89497/free-music-downloads-on-googles-android-phones/`.

Németh, A. and Takács, K. The Evolution of Altruism in Spatially Structured Populations. *Journal of Artificial Societies and Social Simulation, http://jasss.soc.surrey.ac.uk/10/3/4.html*, *10*(3).

Nguyen-Duc, M., Briot, J.-P., Drogoul, A., and Duong, V. (2003). An application of Multi-Agent Coordination Techniques in Air Traffic Management. In *IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), 13-17 October 2003, Halifax, Canada*, 622–628. IEEE Computer Society.

Noriega, P. (1999). Agent Mediated Auctions: The Fishmarket Metaphor. Technical report, IIIA Monograph Series CSIC: Barcelona.

Nowak, M. A. and Sigmund, K. (1998). Evolution of indirect reciprocity by image scoring. *Nature*, *393*(6685), 573–577.

Nwana, H. S. (1996). Software agents: an overview. *The Knowledge Engineering Review*, *11*(03), 205–244.

Oh, J. C. (1999). Ostracism for improving cooperation in the iterated prisoner's dilemma game. In *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99.*, Volume 2, 891–896.

Okasha, S. (2009). Biological Altruism. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Winter 2009 ed.).

Oliveira, M. D., Purvis, M. K., Cranefield, S., and Nowostawski, M. (2004). Institutions and Commitments in Open Multi-Agent Systems. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), 20-24 September 2004, Beijing, China*, Los Alamitos, CA, USA, 500–503. IEEE Computer Society.

Omidyar, P. (1995). EBay homepage. `http://ebay.com`.

Omidyar, P. (2011). EBay user agreement. `http://pages.ebay.in/help/policies/user-agreement.html`.

O'Reilly, T. (2005). What Is Web 2.0. `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`.

Paine, R. (1967). What is Gossip About? An Alternative Hypothesis. *Man*, *2*(2), 278–285.

Pan, X., Han, C. S., Dauber, K., and Law, K. H. (2007). A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations. *AI Soc.*, *22*, 113–132.

Paolucci, M., Marsero, M., and Conte, R. (2000). What Is the Use of Gossip? A Sensitivity Analysis of the Spreading of Respectful Reputation. In *Tools and Techniques for Social Science Simulation*, 302–314. Physica-Verlag.

Post, S. and Neimark, J. (2007). *Why Good Things Happen to Good People*. Broadway.

Pouwelse, J., Garbacki, P., Epema, D., and Sips, H. (2005). The Bittorrent P2P File-Sharing System: Measurements and Analysis. In M. Castro and R. van Renesse (Eds.), *Peer-to-Peer Systems IV*, Volume 3640 of *Lecture Notes in Computer Science*, 205–216. Springer Berlin / Heidelberg. 10.1007/11558989-19.

Pujol, J. M. (2006). *Structure in Artificial Societies*. Ph. D. thesis, Technical University of Catalonia, Barcelona, Spain.

Ramaswamy, L. and Liu, L. (2003). Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences, HICSS 2003*, Volume 7, Los Alamitos, CA, USA, 220–229. IEEE Computer Society.

Rasmusson, L. and Jansson, S. (1996). Simulated social control for secure Internet commerce. In *Proceedings of the 1996 workshop on New security paradigms*, NSPW '96, New York, NY, USA, 18–25. ACM.

Rilling, J. K., Gutman, D. A., Zeh, T. R., Pagnoni, G., Berns, G. S., and Kilts, C. D. (2002). A Neural Basis for Social Cooperation. *Neuron*, *35*(2), 395 – 405.

Riolo, R. L. (1997). The Effects of Tag-Mediated Selection of Partners in Populations Playing the Iterated Prisoner's Dilemma. In T. Bck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, San Francisco, CA, 378–385. Morgan Kaufmann.

Riolo, R. L., Cohen, M. D., and Axelrod, R. (2001). Evolution of cooperation without reciprocity. *Nature*, *414*(6862), 441–443.

Riolo, R. L., Cohen, M. D., and Axelrod, R. (2002). Does similarity breed cooperation? (Reply). *Nature*, *418*(6897), 500–500.

Roberts, G. and Sherratt, T. (2002). Behavioural evolution (Communication arising): Does similarity breed cooperation? *Nature*, *418*(6897), 499–500.

Roberts, J. M. (1964). The Self-Management of Cultures. In W. H. Goodenough (Ed.), *Explorations in Cultural Anthropology:Essays in Honor of George Peter Murdock*, NewYork, 433–454. McGraw-Hill.

Rosedale, P. (2003). Second Life homepage. `http://secondlife.com`.

Rushton, P. J. and Sorrentino, R. M. (1981). *Altruism and helping behavior : social, personality, and developmental perspectives / edited by J. Philippe Rushton, Richard M. Sorrentino*. L. Erlbaum Associates, Hillsdale, N.J. :.

Rymaszewski, M., Au, W. J., Wallace, M., Winters, C., Ondrejka, C., Batstone-Cunningham, B., and Rosedale, P. (2006). *Second Life: The Official Guide*. Wiley.

Savarimuthu, S. (2010a). Self-organising groups (GUI for closed society). University of Otago, `http://unitube.otago.ac.nz/view?m=9GT31pqTPSk`.

Savarimuthu, S. (2010b). Self-organising groups (GUI for lifespan=300+). University of Otago, http://unitube.otago.ac.nz/view?m=JHaY1poMt9P.

Savarimuthu, S. (2010c). Self-organising groups (GUI for lifespan=500+). University of Otago, http://unitube.otago.ac.nz/view?m=7SK81pp4WP0.

Savarimuthu, S. (2010d). Self-organising groups (GUI for open society). University of Otago, http://unitube.otago.ac.nz/view?m=HbOw1pni7qS.

Schelling, T. C. (1969). Models of Segregation. *The American Economic Review*, *59*(2), 488–493.

Shoham, Y. and Leyton-Brown, K. (2009). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. New York: Cambridge University Press.

Sierra, C. (2004). Agent-Mediated Electronic Commerce. *Autonomous Agents and Multi-Agent Systems*, *9*(3), 285–301.

Singh, M. P. (1999). An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts. *Artificial Intelligence and Law*, *7*, 97–113.

Smith, R. G. and Davis, R. (1981). Frameworks for Cooperation in Distributed Problem Solving. *IEEE Transactions on Systems, Man and Cybernetics*, *11*(1), 61 –70.

Sober, E. and Wilson, D. S. (1998). *Unto Others: The Evolution and Psychology of Unselfish Behavior*. Harvard University Press.

Sommerfeld, R. D., Krambeck, H.-J., Semmann, D., and Milinski, M. (2007). Gossip as an alternative for direct observation in games of indirect reciprocity. *Proceedings of the National Academy of Sciences*, *104*(44), 17435–17440.

Steeb, R., McArthur, D. J., Cammarata, S. J., and Narain, S. (1986). *Distributed problem solving for air fleet control: framework and implementation*, 391–432. Boston, MA, USA: Addison-Wesley Longman Publishing.

Stirling, R. B. (1956). Some Psychological Mechanisms Operative in Gossip. *Social Forces*, *34*(3), 262–267.

Suleiman, R., Troitzsch, K. G., and Gilbert, N. (2003). Tools and Techniques for Social Science Simulation. *Journal of Artificial Societies and Social Simulation*, *6*(1).

Suls, J. M. (1977). Gossip as Social Comparison. *Journal of Communication*, *27*(1), 164–168.

Swaminathan, J. M., Smith, S. F., and Sadeh, N. M. (1997). Modeling Supply Chain Dynamics: A Multiagent Approach. *Decision Sciences*, *29*(3), 607–632.

Tankersley, D., Stowe, C. J., and Huettel, S. (2007). Altruism is associated with an increased neural response to agency. *Nature neuroscience*, *10*(2), 150–151.

Trivers, R. L. (1971). The Evolution of Reciprocal Altruism. *The Quarterly Review of Biology*, *46*(1), 35–57.

Tucker, A. W. (1950). A Two-Person Dilemma. Stanford University-technical reports, Readings In Games and Information, `http://www.rasmusen.org/x/images/pd.jpg`.

Vazquez-Salceda, J., Aldewereld, H., and Dignum, F. (2004). Implementing Norms in Multiagent Systems. In G. Lindemann, J. Denzinger, I. Timm, and R. Unland (Eds.), *Multiagent System Technologies*, Volume 3187 of *Lecture Notes in Computer Science*, 313–327. Springer Berlin / Heidelberg. 10.1007/978-3-540-30082-3-23.

Wasserman, P. D. (1989). *Neural computing: theory and practice*. New York, NY, USA: Van Nostrand Reinhold.

Weiss, G. (Ed.) (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. Cambridge, MA, USA: MIT Press.

Wilkinson, G. S. (1984). Reciprocal food sharing in the vampire bat. *Nature*, *308*(5955), 181–184.

Williams, K. D. (2001). *Ostracism: The power of silence*. NY: Guilford Press.

Wooldridge, M. J. (2009). *An Introduction to MultiAgent Systems* (2 ed.). Chichester, UK: Wiley.

Wooldridge, M. J. and Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, *10*(2), 115–152.

Yolum, P. and Singh, M. P. (2003a). Dynamic communities in referral networks. *Web Intelligence and Agent Systems*, *1*, 105–116.

Yolum, P. and Singh, M. P. (2003b). Emergent properties of referral systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS '03, New York, NY, USA, 592–599. ACM.

Yolum, P. and Singh, M. P. (2005). Engineering self-organizing referral networks for trustworthy service selection. *IEEE Transactions on Systems, Man and Cybernetics, Part A,*, *35*(3), 396–407.

Yu, B., Li, C., Singh, M. P., and Sycara, K. (2004). A Dynamic Pricing Mechanisms for P2P Referral Systems. *International Joint Conference on Autonomous Agents and Multiagent Systems*, *3*, 1426–1427.

Yu, B. and Singh, M. P. (2000). A Social Mechanism of Reputation Management in Electronic Communities. In M. Klusch and L. Kerschberg (Eds.), *Proceedings of the 4th International Workshop on Cooperative Information Agents IV, The Future of Information Agents in Cyberspace*, Lecture Notes in Computer Science, London, UK, 154–165. Springer-Verlag.

Yu, B. and Singh, M. P. (2002). Emergence of agent-based referral networks. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy,*, 1268–1269. ACM.

Zaharia, M. A. and Keshav, S. (2008). Gossip-based search selection in hybrid peer-to-peer networks. *Concurrency and Computation: Practice and Experience*, *20*(2), 139–153.

Zennstrm, N. and Friis, J. (2001). Kazaa homepage. `http://www.kazaa.com`.

Appendix A

Software Specifications

The software tools used are listed below:

- All the experiments were written in Java using JDK 1.6 (Java Development Kit) with the help of IDE (Integrated Development Environment), Eclipse 3.2.
- Graphs were created using either Gnuplot 4.2 or Microsoft Excel 2003/2007.
- Videos were captured using CamStudio 2.0.
- Thesis was written in LaTex with miktex 2.9 using the LaTex editor TeXnicCenter 1.0.

Appendix B

Source Code

The source code for Experiment 4.3.1 (the very first experiment of the thesis presented in Chapter 4) is given below. This experiment has three class files namely Game.java, Player.java and Gameconstant.java. Game.java is the main file to run. The output, *kcount* and *scount* are written in separate text files, outfileK and outfileS respectively. The values from these files can be plotted as a line graph similar to the graph shown in Figure 4.1 (Tagless model).

B.1 Game.java

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.File;
import java.util.ArrayList;
import java.util.Random;
public class Game {
        /**
         * @param args
         */
        private Player players[];
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Game main=new Game();
                for(int run=0;run<Gameconstants.noOfRuns;run++)
                {
                        main.createPlayers();
                        BufferedWriter outK,outS;
                        try
                        {
                        //file path
                        String filename="D:\\SharmilaS\\EclipseWorkspace" +
                                                        "\\Experiment_notag\\";
                        //file to store K
                        outK = new BufferedWriter(new FileWriter
                                (filename+"outfileK.txt"));
                        //file to store S
                        outS = new BufferedWriter(new FileWriter
                                (filename+"outfileS.txt"));
                        for (int i = 0; i < Gameconstants.noOfIterations; i++)
                                {
                                        main.playgame(i);
                                        main.reproduction();
                                        main.display(i,outK,outS);
                                }
                                outK.close();
```

```
                        outS.close();
                    }
                    catch(IOException e)
                    {
                        e.printStackTrace();
                    }
                }
            }
50          //To create agents
            private void createPlayers()
            {
                players = new Player[Gameconstants.noOfPlayers];
                int K,sharingstring;
                ArrayList sharerslist=generatuniquenos
                (Gameconstants.noOfPlayers_startS);
                for (int j = 0; j < Gameconstants.noOfPlayers; j++) {
60                  if(sharerslist.contains(j))
                    {
                        sharingstring=1;
                    }
                    else
                    {
                        sharingstring=0;
                    }
                    if(j<Gameconstants.noOfPlayers_startK)
                    {
70                      K=1;
                    }
                    else
                    {
                        K=0;
                    }
                    players[j] = new Player(j,K,sharingstring);
                    if(j<Gameconstants.noOfPlayers_startK)
                    {
80                      players[j].addScore(players[j],
                        Gameconstants.benefit);
                    }
                    else
                    {
                        players[j].addScore(players[j],0);
                    }
                }
            }
90          //To perform reproduction
            public void reproduction()
            {
                ArrayList playersreplist=playersreplist();
                ArrayList playerslist1= new ArrayList();
                ArrayList playerslist2= new ArrayList();
                int halfsize=playersreplist.size()/2;
                for(int fill=0;fill<halfsize;fill++)
                {
                    playerslist1.add(playersreplist.get(fill));
100             }
                for(int full=halfsize;full<playersreplist.size();full++)
                {
                    playerslist2.add(playersreplist.get(full));
                }
                for (int i = 0; i <playerslist1.size() ; i++)
                {
                    Player player1=players[((Integer)playerslist1.get(i))];
                    Player player2=players[((Integer)playerslist2.get(i))];
110                 double score1=player1.getScore(player1);
                    double score2=player2.getScore(player2);
                    if(score1>score2)
                    {//System.out.println(" winner :"+player1.getId());
                        player2.setK(0);
                        int sharingbit=player1.getSharingbit();
                        player2.setSharingbit(sharingbit);
                        player2.clearscoremap_entry(player2,0);
                        player2.clearPlaycount();
```

```
                        }
                        else
                        {//System.out.println(" winner :"+player2.getId());
                                player1.setK(0);
                                int sharingbit=player2.getSharingbit();
                                player1.setSharingbit(sharingbit);
                                player1.clearscoremap_entry(player1,0);
                                player1.clearPlaycount();
                        }
                }
        }

//To play game (share or not) between agents
        public void playgame(int iterationno)
        {
                int sharercount=0;
                ArrayList allplayers=playersarraylist();
                ArrayList reverseallplayers=reversearraylist(allplayers);
                for (int i = 0; i <Gameconstants.noOfPlayers ; i++)
                {
                Player player1=players[((Integer)allplayers.get(i))];
                Player player2=players[((Integer)reverseallplayers.get(i))];
          player1.addPlaycount();
                player2.addPlaycount();
                int k1=player1.getK();
                int k2=player2.getK();
                int s1=player1.getSharingbit();
                if(s1==1)
                {
                        sharercount=sharercount+1;
                }
                if(k1==1 && s1==1 && k2==0 )
                {
                player1.addScore(player1,Gameconstants.cost);
                player2.addScore(player2,Gameconstants.benefit);
                player2.setK(1);
                }
        }
}
//To write scount and kcount in the files at the end of each iteration
private void display(int itrNo,BufferedWriter outK,BufferedWriter outS)
        {
                int kcount=0,scount=0;
                try {
                        for (int i = 0; i < Gameconstants.noOfPlayers; i++)
                        {
                                if(players[i].getSharingbit()==1)
                                {
                                        scount++;
                                }
                                if(players[i].getK()==1)
                                {
                                        kcount++;
                                }
                        }
                        outK.write(kcount+"\n");
                        outS.write(scount+"\n");
                }
                catch(IOException e)
                {
                        e.printStackTrace();
                }
        }
        //To choose agents (10%) for reproduction
        private ArrayList playersreplist()
        {
                ArrayList replist=new ArrayList();
                int n;
                int size=Gameconstants.noOfPlayers/10;
                for (int i = 0; i < size; i++)
                {
                        do {
                                n=new Random().nextInt(Gameconstants.noOfPlayers);
```

```
                        }while(replist.contains(n));
                        replist.add(n);
                }
                return replist;
        }

        //Arraylist with agents
        private ArrayList playersarraylist()
        {
                ArrayList no=new ArrayList();
                ArrayList allplayers=new ArrayList();
                int n;
                for (int i = 0; i < Gameconstants.noOfPlayers; i++)
                {
                        do {
                                n=new Random().nextInt(Gameconstants.noOfPlayers);
                        }while(no.contains(n));
                        no.add(n);
                        allplayers.add(n);
                }
                return allplayers;
        }
        //Reverse (half) of the Arraylist with agents
        private ArrayList reversearraylist(ArrayList arraylist)
        {
                ArrayList reverseallplayers=new ArrayList();
                int halfsize=arraylist.size()/2;
                int n=arraylist.size()-1;
                for (int i=n; i>=halfsize; i--)
                {
                        reverseallplayers.add(arraylist.get(i));

                }
                int revarrsize=reverseallplayers.size();
                for (int i=0; i<halfsize; i++)
                {

                        reverseallplayers.add(revarrsize,arraylist.get(i));
                        revarrsize++;
                }
                return reverseallplayers;
        }
        //To select random (50%) to set S
        private ArrayList generatuniquenos(int sharers)
        {
                ArrayList sharerlist=new ArrayList();
                int n;
                for (int i = 0; i < sharers; i++)
                {
                        do {
                                n=new Random().nextInt(Gameconstants.noOfPlayers);
                        }while(sharerlist.contains(n));
                        sharerlist.add(n);
                }
                return sharerlist;
        }
}
```

B.2 Player.java

```
1
import java.util.HashMap;
public class Player {
        private int id;
        private int K;
        private int Sharingbit;
        private int playcount=0;
10      private HashMap playerScore=new HashMap();
        public Player(int id, int K, int Sharingbit)
        {
```

```
                this.setId(id);
                this.setK(K);
                this.setSharingbit(Sharingbit);
        }
        public int getId() {
                return id;
        }
        public void setId(int id) {
                this.id = id;
                        }
        public void setSharingbit(int Sharingbit )
        {
                this.Sharingbit=Sharingbit;
        }
        public int getSharingbit( )
        {
                return Sharingbit;
        }
        public void setK(int K)
        {
                this.K=K;
        }
        public int getK()
        {
                return K;
        }
        public void clearPlaycount()
        {
                playcount=0;
        }
        public void addPlaycount()
        {
                this.playcount=playcount+1;
        }
        public int getPlaycount(Player player)
        {
                return playcount;
        }
        public double getScore(Player player)
        {
                if(playerScore.get(player)== null)
                {
                        return 0;
                }
                else
                {
                double score=(Double)playerScore.get(player);
                return score;
                }
        }
        public void addScore(Player player, double score)
        {
                if(playerScore.get(player)== null)
                {
                        playerScore.put(player, score);
                }
                else
                {
                        double value = (Double)playerScore.get(player);
                        value+= score;
                        playerScore.put(player, value);
                }
        }
        public void clearscoremap_entry(Player player, double value)
        {
                playerScore.put(player,value);
        }
}
```

B.3 GameConstants.java

```
1
    public class Gameconstants {
            public static final int noOfPlayers = 100;
            public static final int noOfPlayers_startK = 20;
            public static final int noOfPlayers_startS = 50;
            public static final int noOfIterations = 1000;
            public static final int noOfRuns=1;
            public static final double benefit=1;
10          public static final double cost=-0.1;
    }
```