An Application of Agent-Based Simulation to a Natural Resource Dilemma

Understanding Payoff, Decision Making, and
Learning of Stakeholders through a
Simulated Environment

Yngve Svalestuen

Master of Science in Computer Science
Submission date: June 2014
Supervisor: Pinar Öztürk, IDI
Co-supervisor: Axel Tidemann, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Thesis statement

Evolutionary game theory (EGT) is a multidisciplinary field which focuses on how agents' decisions influence their behavior. EGT has been applied in many areas, ranging from biology to finance, but agent behavior is often represented by mathematical generalizations, and details are therefore generalized away. Agent-based simulation (ABS) is a technique where each agent is treated as a complex individual with many interactions with other agents and the surrounding environment. Both ABS and EGT are used to study social dilemmas, which are situations where the individuals have two choices where one is greedy (beneficial to the chooser, detrimental to the other agents) and a selfless choice (beneficial to all agents), but if too many agents make the greedy choice it has a negative impact on all agents. A highly relevant form of such a social dilemma are questions regarding climate problems, where countries and organizations that make "green" choices experience negative financial impacts in the short term. The rational strategy is to make environmentally irresponsible but profitable choices that in the long term result in worse conditions for the community. The goal in studying these dilemmas is to find mechanisms which promote cooperation between agents that ensure a sustainable development.

Summary

Aquaculture organizations establish facilities at the coast in Frøya. The facilities block the surrounding area from fishing and cause environmental damage to close natural resources. Fishers who depend on those natural resources get the opportunity to influence the aquaculture expansion through complaints about the municipality's coastal plan. Statistics show that fishers don't complain in this situation, and the aim of this project was to investigate why, as well as studying how baseless complaints can be avoided. Simulation based on rudimentary evolutionary game theoretic analysis were applied in order to model the fishers as intelligent agents with complex interactions. Fishers learn to not complain because they experience that complaining produces no results, and because there is a non-monetary information leak cost associated with complaining. Changing government's payoff for approving complaints to favor fishers while penalizing environmental damage may promote more fisher complaints as well as avoiding baseless ones. Avoiding cluttering of fishing spots is important to keeping complaints on a healthy level. With further development the simulation system could be part of a decision support system that promotes policies that are fair for the stakeholders.

Sammendrag

Oppdrettsorganisasjoner etablerer fasiliteteter på kysten på Frøya. Fasilitetene blokkerer de omkringliggende områdene for fiskeri, og påfører skade til nærliggende naturressurser. Fiskere som er avhengige av de naturressursene får muligheten til å påvirke ekspansjonen av oppdrett gjennom å klage på kommunens kystplan. Statistikk viser at fiskere ikke klager i denne situasjonen, og målet med dette prosjektet var å undersøke hvorfor, samtidig som å studere hvordan grunnløse klager kan unngås. Simulasjon basert på enkel evolusjonær spillteoretisk analyse ble brukt for å modellere fiskerne som intelligente agenter med kompliserte interaksjoner. Fiskere lærer å ikke klage fordi de opplever at å klage ikke produserer resultater, og fordi det er en ikke-finansiell informasjons-kost assosiert med klaging. Å endre de lokale myndighetenes payoff for å etterkomme klager som kommer fiskerne til gode, og samtidig straffe myndighetene for miljøskader, kan føre til at fiskerne klager mer samtidig som at grunnløse klager ikke oppstår. Det er viktig å unngå suboptimal distribusjon av fiskesteder mellom fiskerne for å holde et sunt og ærlig klagenivå. Med videre utvikling kan simulasjonssystemet bli en del av et støttesystem for avgjørelser som promoterer løsninger som er rettferdige for alle partene.

Preface

This thesis concludes my Master degree in Artificial Intelligence, Computer Science at the Norwegian University for Science and Technology (NTNU). With a five-year long period ending, I want to say a few words.

The prelude to this project was conducted during the preceding autumn, when I was writing a project on evolutionary game theory. EGT is an exciting field that is highly multi-disciplinary; an exciting feature in itself. For the first time I was exposed to scientific literature hailing from both biology and economics, as well as computer science. Seeing all these disciplines merge together in a single field was very interesting, and sparked an interest in multi-disciplinary work.

Agent-based systems often include aspects from many disciplines, and this project is no different. Learning about fishing and aquaculture has been fun and eye-opening to a world of complexities I was never aware of. Programming the system itself has been another kind of challenge; keeping the code clear and extensible while simultaneously growing the software to meet its requirements is no easy task.

I want to thank my advisers Pinar Öztürk and Axel Tidemann for our regular meetings where we discussed and theorized about the model. Without their feedback and ideas this thesis would have remained non-existent. I want to thank Jennifer Bailey and Yijae Liu for our first meeting where the initial ideas for the project were lit. I want to thank Rachel Tiller for being our connection with the Frøya fishermen and providing absolutely essential information about the fishing and aquaculture industries. My thanks also go to the Frøya fishers who showed up at our workshop and gave valuable feedback regarding the simulation.

My thesis is dedicated to my wife Laura, who has continuously supported me throughout the whole project, and continues to be my most valuable friend, supporter and love, in my life.

Yngve Svalesuten
Trondheim, 2014

Table of Contents

Summary **i**

Sammendrag **ii**

Preface **iii**

Table of Contents **viii**

List of Tables **ix**

List of Figures **xi**

1 Introduction **1**

1.1 Problem Description 1
1.2 Research Objectives 3
1.3 Report Outline 4

2 Background **5**

2.1 Agents 5
2.2 Communication 6
2.3 Game Theory 6
2.3.1 Classic Game Theory 6
2.3.2 Evolutionary Game Theory 8
2.4 Decision Making Systems 9
2.4.1 Simple rules 9
2.4.2 Artificial neural networks 9
2.4.3 Case-based reasoning 10
2.5 Learning Systems 10
2.5.1 Artificial evolution 10
2.6 Self-Organization 12
2.6.1 Attractors 12

3 State of the Art **13**
3.1 Agent-Based Simulation 13
3.2 Fishing and Aquaculture 14

4 Conceptual Design **17**
4.1 Stakeholders 17
4.1.1 Fishermen 17
4.1.2 Aquaculture Organizations 17
4.1.3 Municipality 18
4.1.4 Government 18
4.1.5 Other stakeholders 18
4.2 Sequence of Events 18
4.2.1 Definition of the Events 19
4.2.2 Defining the Sequence 20
4.3 Decisions 20
4.3.1 Different agents' approaches to decision making 21
4.3.2 Voting 22
4.3.3 Influences 23
4.3.4 Learning 23
4.4 Knowledge 24
4.4.1 Inferring Cell Quality from Complaints 25
4.5 Priorities 25
4.5.1 Profits 25
4.5.2 Wealth of the community 25
4.5.3 Existence of fishing as a business 25
4.5.4 Maintaining natural fish stocks in good health 27
4.5.5 Existence of aquaculture industry 27
4.5.6 As few as possible aquaculture establishments 27
4.5.7 Pleasing voters 27
4.5.8 Pleasing interests of investors in the local community 28
4.6 Information Flow 28
4.7 Data Set 28
4.8 Outputs and Results of the Experiment 28
4.9 Classifying the Simulation 30

5 Experimental Setup **31**
5.1 Simulation Overview 31
5.1.1 Phases 33
5.2 Decision Making 35
5.2.1 Mechanisms for decision making 35
5.2.2 Learning 36
5.3 Agent Communication 36
5.3.1 Messages 36
5.4 Other Rules Affecting the Agents 37
5.4.1 Map creation 37
5.4.2 Aquaculture building 37

5.5 Priorities of the Different Groups 38
5.5.1 Representing priorities realistically 38
5.5.2 Calculating priority satisfaction 38
5.6 Actions of the Different Groups 39
5.6.1 Fishermen actions 39
5.6.2 Municipality 39
5.6.3 Government 39
5.6.4 Aquaculture organizations 40
5.6.5 Other agents 40

6 Implementation **41**
6.1 Graphical User Interface 42
6.1.1 Phase Information Panel 42
6.1.2 Controls panel 42
6.1.3 Messages 43
6.1.4 Plots 43
6.1.5 Map 46
6.2 Configuration File Specification 46
6.3 Modules and classes 46
6.4 Program Dependencies 46
6.5 Extending the Program 47

7 Workshop in Frøya **49**
7.1 Presenting the Simulation 49
7.2 Feedback 49
7.3 Revisions to the Simulation 51
7.3.1 Environmental damage 51
7.3.2 Government decision 51

8 Experimental Results **53**
8.1 Experiment 1 53
8.2 Experiment 2 55
8.3 Experiment 3 57
8.4 Experiment 4 62

9 Discussion **65**
9.1 Why Fishermen Don't Complain 65
9.1.1 Importance of the learning process 66
9.1.2 Influence of complaint approval rate on complaining behavior . . 67
9.1.3 Factoring environmental damage into complaint rates 68
9.2 How to Remedy False Complaints 68
9.2.1 Complaints during the stabilized state 68
9.2.2 When complaining takes place 69

10 Future Work **71**
10.1 Focusing on Other Decision-Making Agents 71
10.2 Decision Support System 72

11 Conclusions **73**

Bibliography **74**

A Documentation of the program configuration **77**
A.1 Preprocessing 79
A.2 Python Classes 79
A.3 Priorities 79
A.4 Fields 80

B Documentation of modules and classes included in the system **85**
B.1 Modules in the **FisherSimulation** package 85

C Extending and Altering the Program: Interfaces and Plug-ins **97**
C.1 Map Structure 97
C.2 Learning Mechanism 99
C.2.1 Phenotype 100
C.2.2 Genotype 100
C.3 Voting Mechanism 101
C.4 Government Decision Mechanism 102
C.5 Coastal Planning Mechanism 102

D Software and Project Details **103**

List of Tables

4.1 Inputs for the voting decision for each agent 23
4.2 Overview of priorities 26

5.1 Actions of the different agents 40

8.1 Overview of experiments 53
8.2 Common experiment parameters 54
8.3 Relationship between approval rate and final world state 56
8.4 Average number of open cells and aquaculture for different outcomes . . 57
8.5 Relationship between outcome and average fitness 60
8.6 Effects of changing environmental damage on final states 62

List of Figures

1.1 Payoff for fishermen 2

2.1 Payoff matrix for the Prisoner's Dilemma 7
2.2 Overview of evolutionary cycle 11

4.1 Pseudocode showing the sequence of events 21
4.2 Information flow diagram 29

5.1 Process chart of the simulation 32

6.1 Screenshot of the graphical user interface of *FisherSimulation*. 42
6.2 Symbols in the map 46

8.1 Plots showing a collapsed state 58
8.2 Plots showing a stabilized state 58
8.3 Maps showing typical collapsed and stabilized states 59
8.4 Fishermen movement on a stable map 59
8.5 Average number of complaints for each round of several simulations . . . 61
8.6 Average number of complaints for each of the hearings in every round . . 61

Chapter 1

Introduction

This thesis is about fishermen and evolutionary game theory, simulated with an agent-based model. A simulation-based study is performed to investigate how fishers approach the establishment of aquaculture facilities in territories that are traditionally used exclusively for fishing. The fishers can choose to officially complain about the establishment of new aquaculture facilities during the municipality's coastal planning. The foundation for the research questions is a game-theoretic analysis that considers fishers as purely rational agents, with straight-forward choices to complain or not. The game-theoretic model predicts that fishers will always complain as much as they can in order to protect their incomes. However, empirical evidence from the municipality of Frøya, Sør-Trønderlag, Norway shows that fishers are welcoming to the establishment of these facilities, and don't officially complain about them, even if they are both directly and indirectly harming the fishing industry. This contrast between the game-theoretic prediction and observed behavior in the real-life scenario is the basis of the simulation, which attempts to recreate the fishers' situation, and investigate how intelligent agents behave in an environment that resembles the real-life scenario acts.

The simulation is used to conduct experiments that produce data that is analyzed to answer the research questions.

1.1 Problem Description

This project is heavily based on research by Tiller et al. (2014) on fishermen and other stakeholders in Frøya with regards to aquaculture expansion. In Frøya there are many fishermen who live off fishing from the coast. They are dependent on access to good fishing spots, as well as the overall health of fish. Meanwhile, aquaculture companies want to establish facilities close to the coast line, where conditions are optimal. These conditions often coincide with where harvest is more effective for the fishermen, so aquaculture companies and fishermen are in effect competing for the same spots.

In order to obtain permission from the government to build facilities, aquaculture companies have to apply for licenses from the municipality. The municipality creates a coastal

$$\begin{array}{ccc} & & \text{Fisherman} \\ & & \begin{array}{cc} \textit{Complain} & \textit{Stay Silent} \end{array} \\ \text{Government} & \begin{array}{c} \textit{Allow} \\ \textit{Deny} \end{array} & \begin{bmatrix} -c & 0 \\ r-c & r \end{bmatrix} \end{array}$$

Figure 1.1: Payoff for fishermen depending on government actions, in the scenario where a fisherman's fishery spot is being targeted by an aquaculture organization. The cost of complaining is c and the reward for the aquaculture being denied is r.

plan at a somewhat regular time interval for dividing the sea resources for different purposes. Areas where aquaculture is allowed is a part of this plan, and aquaculture companies can apply for licenses in these areas when the plan is finished. During the planning, fishermen and other stakeholders can influence the process through complaints about certain locations. For instance, the combined local fisherman community in Frøya has twice complained about areas that were where the municipality wanted to allow aquaculture expansion, because they felt the areas were too valuable as fishing spots. The complaints are not always followed, however, and both the municipality and the regional government of Frøya are very interested in having aquaculture.

The fishermen in Frøya are in a special situation, since they actually want aquaculture in their community. They have experienced the value of having aquaculture in the community, which improves living conditions through the municipality having more money. This means better infrastructure for everyone, better schools for their children, and better social circumstances as youths stay in the community instead of moving to the cities when they grow up.

Even so, the fishermen's main priority, according to Tiller et al., is the continued existence of fishing as a commercial industry. When considering this, the aquaculture facilities pose a threat as well as a benefit, since they compete for the same coastal resources. The logical course of action for the fishermen is to complain at every opportunity, to the establishment of aquaculture facilities. A simple game theoretic analysis yields the payoff matrix depicted in fig. 1.1. As long as the reward r for aquaculture being denied is greater than the cost of complaining c, it seems that it would be beneficial to always complain. However, the observed behavior of the fishermen at Frøya is that they don't complain much, and when they do, it's too late. Together the reward r and the cost c of an outcome makes up the payoff for that outcome. Payoff is equivalent to utility, which is the value a rational agent tries to optimize.

A possible reason for not complaining might be that complaining reveals good spots. Fishermen usually have their personal spots that belong to the family, and are respected by others. They may also have secret spots that not everyone knows are good fishing spots. A risk associated with complaining about aquaculture in a specific area is that other fishermen might infer that the complaining fisherman has found a really good fishing spot and that's why he's complaining.

The point of the simulation is to explore this and other possible explanations as to why fishermen in Frøya don't complain—what factors are in play that make their actions rational? Another possible problem associated with the situation in general occurs if the fishermen come to the conclusion that complaining is always good, even if the area in question is not that important to them. In this case, a simulation may be able to explore different policies to discourage this behavior.

1.2 Research Objectives

The project is divided into research objectives, which are smaller goals that in combination provide the direction for the project. The objectives are stated as research questions, with working hypotheses, and they will be further visited in later chapters. The discussion chapter will focus on the research questions and how they are answered through the experiments.

Research Question 1: What can a simulation tell us about the fishers' decisions to not complain?

One reason why fishers don't complain may be the cost associated with complaining. There is no direct monetary cost to filing a complaint, but there is a significant effort cost. Complaints are usually issued by the organized group of fishers, so in order to complain they need to do the necessary organization to collaborate on such a complaint. There is also the cost associated with each fishers' individual time spent on writing and documenting the complaint. Complaining may directly negatively impact their profits if the fishers have to take time off work in order to write the complaint, or it may dig into their spare time, which is also a kind of cost. If the cost is too large for the fishers, especially compared to its expected return, it is natural that they will opt out of complaining.

Another reason why complaining may become unpopular is of course the expected return itself. Even assuming that every time the fishers manage to protect an area from aquaculture expansion through complaining will result in a constant benefit, there is no guarantee that a complaint will be approved. If the government's complaint approval rate is observed by the fishermen to be too low, this may cause the fishers to not complain at all, because they observe no benefit or a benefit that occurs far too rarely, compared to the cost. Even if the benefit when it occurs is far greater than the cost associated with complaining, this effect may still take place Social learning dictates if they don't perceive any benefit, even if there might be one, they might learn to not complain.

The experiments are designed to help understand if the cost of complaining may be influential, and if the benefit of complaining is enough to encourage complaining behavior. They are designed to identify when complaining behavior disappears from the population.

Research Question 2: Under what conditions do false complaints occur, and what policies can be used to discourage this behavior?

Since the project was started, it has been speculated and reported that fishers may produce false complaints. False complaints can be considered a problem because aquaculture facilities are considered an overall positive for the communities they are in, especially if they don't conflict with any fishers. If there are false complaints, this may hurt the community unnecessarily by preventing facilities that would be problem-free.

One measure that can be taken into consideration in order to stop false complaints is to introduce a monetary cost of complaining. Such a cost would discourage complaining about areas that have no associated benefit if the complaint goes through, i.e. no good fishing area would be destroyed by having aquaculture there anyway. A problem with this strategy could be that fishers would be discouraged from complaining altogether, especially considering that low complaint rates are already a problem, and existing non-monetary costs may be a part of this problem.

Another solution to false complaints could be that the entity that receives complaints, or even better a neutral third party, could perform their own investigation to check if the complaints were based on legitimate concerns. Then the municipality or government could identify or even punish false complaints in order to weed out this behavior. However, the action of checking if a complaint is about a good fishing area or not, may become very expensive. It may also not be feasible since the areas that are good by the fishers standards may vary or be based on subjective criteria.

The experiments are designed to investigate when false complaints occur in the system, and identify conditions that prevent them.

1.3 Report Outline

The project report is structured as follows. This chapter (chapter 1) includes a general and high-level description of the project, as well as problem statement and statement of research questions and working hypotheses. Chapter 2 provides the theoretic background of the project, including agent theory and game theoretic concepts. Chapter 3 presents state-of-the-art research relevant to the project, which includes current agent-based simulation techniques coupled with aquatic and ecologic scenarios. Chapter 4 provides a high-level description and discussion of elements that are fundamental to the simulation, presented separately from the simulation. Chapter 5 describes the experimental setup which includes implementation-independent details of the simulation. Chapter 6 documents the implementation details of the software that performs the simulations. Chapter 7 documents and discusses results from the fisher workshop that was held in Frøya municipality. Chapter 8 presents specific experiments and their results, tied to the research questions. Chapter 9 discusses the results with regards to the research questions. Chapter 10 discusses possible further developments of the software and the project topic.

The appendix includes documentation of the software created during the project. Appendix A documents the configuration setup, while appendix B documents modules and classes included in the system. Appendix C describes interfaces that can be implemented in order to extend the functionality of the software. Appendix D gives an overview of project details regarding the software.

Chapter 2

Background

This chapter gives an introduction to each concept in the theoretical background of the thesis.

2.1 Agents

The definition of an *agent* varies from author to author, but Wooldridge (2002a) defines them as computer programs placed in an environment and that can autonomously perform actions in the environment in order to fulfill their objectives. The actions do not necessarily *control* the environment, but do at best *influence* it. They receive inputs from the environment as well, through sensory systems, which may invoke reactions that produce more output to the environment.

Since the environment of an agent is such an intrinsic part of its definition, the environment needs to be elaborated as well. Environments can be fully observable or only partially so, deterministic or stochastic, single agent or multi-agent, static or dynamic, episodic or sequential, known or unknown and discrete or continuous. An observable environment is one which agents can easily gather information about. Multi-agent systems have environments with more than one agent, and they can be cooperative, competitive or a mixture of both. A deterministic environment has predictable behavior given enough information about it, however this property is irrelevant if the environment becomes sufficiently complex because agents usually do not have enough processing power or information-gathering abilities to calculate the necessary predictions. Static environments are different from dynamic ones in that an agent can assume that only its own actions impact a static environment. Episodic environments have a list of events that happen in order, without the occurrence of the next event being dependent on the previous one, while in sequential environments actions affect subsequent events. Discrete environments are distinct from continuous ones by the property that they have a finite number of possible states. In a known environment, agents have knowledge about the laws that govern the environment, and thus the ability to reason about consequences from perceived information (Russell and Norvig, 2010a).

2.2 Communication

Agent communication is a heavily studied topic. It is important because, in contrast to more traditionally object-oriented systems, agents are autonomous and do not necessarily obey the commands of others. A communication scheme can therefore not consist of only method invocations, but needs to consider the *will* of the agents. Messages may consist of information provided by the sender to the recipient, or it may be a request for information, which is simply information that the sender wants information from the recipient. A communicated message is just information; it does not have to be obeyed by its recipient (Wooldridge, 2002b).

Several languages have been developed for the purpose of agent-based communication. KIF and KQML are two such languages. KIF was constructed to convey the logical meaning of a message, while KQML is a system for expressing all the other information regarding the message such as sender, recipient, what message it was a reply to and so on. FIPA is another messaging scheme that's similar to KQML (Wooldridge, 2002b).

2.3 Game Theory

Evolutionary game theory is a subsidiary field of game theory, which was originally a field from economics and has developed into a more general study of multi-agent interactions where the decision of each agent has an impact for everyone. *Classic* game theory traditionally uses strategic reasoning to find the optimal choice for each agent. *Evolutionary* game theory applies evolutionary dynamics to classic game theory, and substitutes pure strategic reasoning by a single agent with a large population that are subject to selection over a large time span in order to derive the optimal choices.

2.3.1 Classic Game Theory

Many of the concepts used in evolutionary game theory are directly adopted from the classic variant. This section aims to cover the most important concepts that are used in both classic and evolutionary game theory.

Payoff and strategies

Since game theory is a theory about decision making, one needs a concept to describe which decisions lead to certain outcomes. *Payoff* is a measure of the utility each agent receives for taking certain actions, given the actions of the other agents affecting the outcome. Payoff is often given as a number and presented in a table, known as a *payoff matrix*. Figure 2.1 shows the payoff matrix for the *Prisoner's Dilemma* game in which two prisoners get the opportunity to confess or stay silent about their crimes. If they both choose to confess, they both receive a moderate punishment. If they both stay silent, they receive a very harsh punishment. If one confesses and one stays silent, the one that confessed receives a light punishment, but the one who stayed silent is let go. The choice to confess is often stated as *to cooperate*, while the choice to stay silent is *to defect*.

$$\begin{array}{ccc} & & \text{Prisoner B} \\ & & \begin{array}{cc} C & D \end{array} \\ \text{Prisoner A} & \begin{array}{c} C \\ D \end{array} & \left[\begin{array}{cc} \text{-1, -1} & \text{-5, 0} \\ \text{0, -5} & \text{-3, -3} \end{array}\right] \end{array}$$

Figure 2.1: Payoff matrix for the Prisoner's Dilemma game. C: Cooperate (confess), D: Defect (stay silent). The payoffs are the numbers of years in prison multiplied by -1; a payoff of -5 means 5 years in prison. The numbers in the matrix represent the payoffs for each agent; the first number for prisoner A and the second for prisoner B. When player A cooperates and player B defects (C, D), player A receives -5 payoff and player B receives 0.

A *strategy* is the choice an agent can make. For the prisoner's dilemma, the obvious choices are to either confess or to stay silent, but sometimes *mixed* strategies are also allowed. A mixed strategies is a distribution of probabilities to the set of pure strategies, so for the prisoner's dilemma a mixed strategy could be "defect 70 % of the time, otherwise cooperate".

Nash Equilibrium

The Nash equilibrium is a set of choices agents can make that are *best responses* to each other. This way when two or more agents are in a Nash equilibrium, there is no reason for any of them to change their decisions. A best response to a certain move is simply the strategy which yields the highest payoff. In terms of the prisoner's dilemma, defecting is the best response to both cooperating and defecting, or (D, D). Therefore, the Nash equilibrium of the prisoner's dilemma is mutual defection. The prisoner's dilemma has exactly one Nash equilibrium of two pure strategies, namely mutual defection, but *every* game has at least one *mixed* strategy Nash equilibrium.

Pareto Optimality

In the prisoner's dilemma, mutual defection is the Nash equilibrium, and gives both players a payoff of -3. However, even if it is the Nash equilibrium, the situation could be improved for both players without either player suffering. For example, if both players cooperate, they both receive a payoff of -1 instead. This is clearly a more desirable outcome overall. The concept of *Pareto optimality* captures this notion of improving outcome without hurting the others' outcomes. An outcome is Pareto optimal, or Pareto efficient, if no player's payoff can be improved without lowering another player's payoff. (C, D) and (D, C) are also Pareto optimal in the prisoner's dilemma because any change in the strategy for a player in order to improve the payoff for one would lower the other player's payoff since both end up with -1, compared to 0 for the defector and -5 for the cooperator previously. For example, if the player defecting in (C, D) changes his strategy to C, it will lower his payoff while improving that of the other player. Any change in strategy from (C, D) would do the same; lowering the payoff of the player who initially defected. Since no change is possible that benefits one player while not hurting the other, the outcomes (C, D) and (D, C) are Pareto-optimal.

2.3.2 Evolutionary Game Theory

The two most important concepts in classic game theory are payoff and Nash equilibrium. In evolutionary game theory, these concepts are refined to fit with the evolutionary dynamics. This section presents these refinements.

Fitness

Evolutionary dynamics are based on the survival of the fittest, which is defined in terms of *fitness*. Fitness is a measure of how well suited an agent is for its environment, and in evolutionary game theory *payoff* is used as the measure of fitness. Fitness decides which entities are reproduced, and thus which strategies are successful and propagate to the next generation.

Evolutionary Stable Strategies

A refinement of Nash equilibrium, an *evolutionary stable strategy* is a strategy that if there is a population consisting of that strategy, no mutant strategy that is evolutionary stable can take over the population. In other words, the first criteria for a strategy to be evolutionary stable is that it has to be better when played against itself than every other strategy. Otherwise, if another is equally good against it, the stable strategy has to be better against every other strategy than that strategy than that is against itself. Formally, a strategy s is evolutionary stable if for every strategy t:

1. $\pi(s, s) > \pi(t, s)$, or
2. $\pi(s, s) = \pi(t, s)$ and $\pi(s, u) > \pi(u, u)$

where $\pi(t, s)$ is the payoff of strategy s played against t.

Replicator Dynamics

Assuming an infinite population of agents playing different strategies, the evolution of the strategies can be described by the *replicator dynamics*. The replicator dynamics is a formula that describes how the fraction of agents playing a certain strategy will change in relation to the other fractions of agents playing other strategies, depending on their payoff against each other:

$$\frac{d}{dt}p_i(t) = p_i(t)(\pi_i(t) - \bar{\pi}(t))$$

Where $p_i(t)$ is the fraction of players performing strategy s_i at time t, and $\pi_i(t)$ is the *average* payoff of performing strategy s_i in the population at time t, and $\bar{pi}(t)$ is the average payoff over *all* strategies s at time t (Gintis, 2000).

Social Learning

The most straight-forward way to interpret evolutionary game theory and the replicator dynamics is in the natural evolutionary sense; i.e. as a life-death reproduction cycle. However, evolutionary game theory is increasingly being applied to social science problems where generations of evolutionary development is too slow of a process to model real people changing behavior. On the contrary, the evolutionary dynamics can be interpreted as *imitation* such that individuals imperfectly copies the strategies of their best-performing peers. They copy imperfectly to model mutation, but this aspect can also be left out and leave the process as a pure optimization without discovery of new strategies.

When the replicator dynamics are considered as imitation, a question arises on how the imitation takes place. Laland and Rendell (2010) reviews various ways to copy other agents, so-called *imitation dynamics*, concluding that the most common methods are *copy-the-better* which copies the best-performing individual in the population, and the probabilistic variant *copy-the-proportional*.

2.4 Decision Making Systems

Decision making can take many forms, for instance in the game theoretic view where decision making means following a strategy. However, a decision-making agents often have the requirement to be able to reason about their beliefs and desires in order to theorize about different decisions. Sometimes these desires and beliefs are explicitly defined and used by the agent, making it a decision-theoretic agent (Russell and Norvig, 2010b). Conversely, agents may be very simple and have straight-forward rules for always choosing the same decisions, or based on simple conditions.

2.4.1 Simple rules

The most straight-forward way to model decision-making agents is to preprogram their decisions before they are initiated. If the goal of the model is to observe how different decisions perform against each other, assigning each decision to a subset of agents is a reasonable strategy. Such non-intelligent agents are equivalent to agents performing a single *strategy* in the game-theoretic sense. A natural extension of the simple rules-strategy would be to assign mixed strategies to agents.

2.4.2 Artificial neural networks

An artificial neural network can be used for decision making without making the desires and beliefs of an agent explicit. Neural networks consist of a set of neurons with connections between them. The neurons are ordered in three layers: input, hidden and output. The input layer contains the nodes that together with the recurrent connections control the network; these are manipulated directly with data from the outside world. The hidden layer provides the data processing, and generally consists of a larger amount of neurons than the other layers with more connections between them, as well as recurrent connections which provide memory to the network. The output layer typically contains few nodes that which states are interpreted as a decision. For instance a single neuron that can have positive or negative values can be interpreted as answering a yes/no question. The connections between the neurons therefore encode the beliefs and wants of the agent implicitly.

There are three aspects that make neural networks differ from one another: the number of neurons, connectivity between the neurons, and the weights on each connection. The number of neurons and connectivity between them are typically decided upon creation of the network, while the weights are adjusted through a training process. However, there are examples of systems that adjust connectivity and the number of nodes during the training as well.

2.4.3 Case-based reasoning

Case-based reasoning is a learning method that can be used for decision making (Mitchell, 1997). Case-based reasoning systems are comprised of cases, which are descriptions of problem situations with solutions for them. When presented with a new problem situation, the system uses similarity metrics to find the most similar previous situations and adapts their solutions to the current situation. It is a lazy paradigm, which means that it creates generalizing models based on its data when asked for a new decision. These queries are referred to as “cases”, and they are elaborate representations about the problem. Descriptions of the current state of the environment, any parameters and relationships between them can all be part of the query. When a query is answered it is also stored with its generated solution, for use in the future.

The case-based reasoning paradigm is defined by a number of features that differ it from similar learning systems. First of all, the detailed query descriptions is an important feature. Comparing queries to find the most similar ones is therefore a complex task. Case-based reasoning systems can also adapt previous solutions in a number of ways to fit the current problem, for instance combining two or more previous cases and their solutions in the new solution. Case-based reasoning is highly knowledge-based, and is used in dynamic, learning environments. Applying case-based reasoning to a system which cannot learn makes little sense because its advantage is the possibility to continuously expand and adapt its knowledge of cases.

2.5 Learning Systems

Studying a decision making mechanism that cannot adapt can be useful in order to determine its current effectiveness, but an assumption in for instance evolutionary game theory is that agents can *change* their behavior. A common way of changing behavior is through learning, which can be implemented in various ways. One common way of considering learning in evolutionary game theory is as simple imitation of the best-performing agent, or the slightly more complex method of copying the best-performing agents with a probability proportional to the difference in performance to that agent.

2.5.1 Artificial evolution

Evolutionary algorithms are important tools to in both artificial intelligence research and other areas of application. They are a form of unsupervised learning, where the central requirements to their application are *a*) a way of separating *genotypes* from *phenotypes*; and *b*) a way of measuring the fitness of said phenotypes. A genotype is a low-level (often binary) representation of the entities being evolved which can be subject to mutation. The

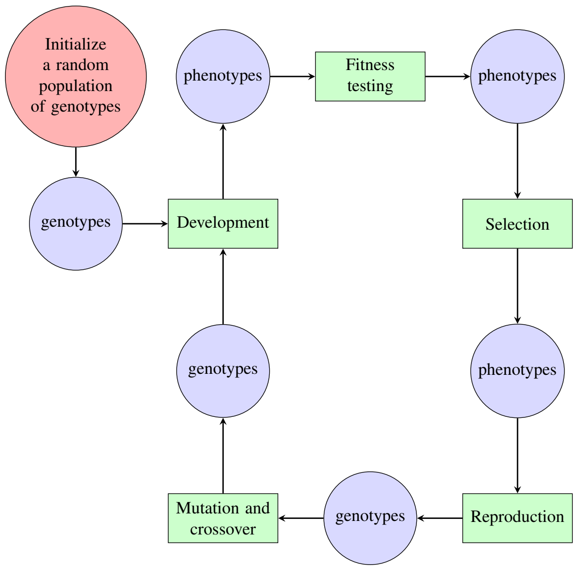

Figure 2.2: Overview of the evolutionary cycle, showing events (green) and entities sent between them (blue).

corresponding phenotype is the actual entity, built from the genome description (Floreano and Mattiussi, 2008).

Artificial evolution is usually implemented in a cycle inspired by natural evolution. There is *development* from genome to phenotype, testing of the *fitness* of each phenotype, *selection* of the best-fitting individuals, *reproduction* of the selected samples, and *mutation* as well as *crossover* of their genomes, before a new development cycle takes place. Figure 2.2 shows this cycle, distinguishing between the events that transform the entities, and the entities themselves.

There are many algorithms for performing selection, among them are *proportional* selection and *rank selection*. Proportional selection assigns probabilities to each genome according to its relative fitness, giving the highest probability to the best-performing individual. Then it selects n genomes repeatedly using this probability. Since the proportional selection mechanism considers absolute fitness values, it is sub-optimal when there is either low variation between the fitness values, and when one has much higher fitness than the others. In the first case, the probability difference in choosing between the best performers and worst performers won't be much, and the evolution will be very random. In the second case, genetic diversity will suffer since the best entity will be copied much more commonly than the others. Rank selection mitigates these issues by instead of using probabilities based on the absolute fitness values, it assigns probabilities depending on the relative ranking of each individual (Floreano and Mattiussi, 2008).

When reproducing a generation, it is common to completely replace the old generation with new genotypes and phenotypes derived from the reproduction mechanism. However, to prevent the loss of good performers through random mutation, *elitism* is a technique of saving the best individuals or genomes without mutation and separate from the selection, that are guaranteed to go on to the next generation (Floreano and Mattiussi, 2008).

The two most common mutation operations are direct mutation on random bits, and crossover. Regular mutation can be implemented by randomly choosing one or more bits in the genome and transforming them into something random. Crossover takes two genomes, cuts them in half at corresponding points, and joins the pieces together in the opposite configuration, producing two new genomes. Crossover has the possibility to produce huge changes, so it needs to be carefully implemented.

2.6 Self-Organization

Agent-based simulations and evolutionary processes often become *self-organized* systems. A system can be described as self-organizing when global patterns are created by local interactions (Heylighen, 2001). This creation of a global pattern is commonly referred to as *emergence*. Self-organized systems are characterized by a number of features in addition to the global order occurring from local interactions. In contrast to centralized systems, self-organized systems don't have a single entity controlling the overall behavior, but control is *distributed* among the parts. Self-organizing systems are also robust to noise, errors and in general outside influences. One of the reasons for this robustness is the distributed control; they do not possess the vulnerability to its central control unit being destroyed like traditional centralized systems do. Self-organization often occurs because of *feedback loops* in the system. Feedback loops can either reinforce signals to explosively grow development (positive feedback), or stabilize it, both in a circular fashion. Feedback effects can make self-organized systems difficult to predict, because the interactions of inter-dependent positive and negative feedback loops as a result of a stimulus can become highly complex.

Agent-based models are self-organizing because they model individual agents in order to create and observe emergent behavior. Studying emergent properties of a self-organized community is one of the main reasons why agent-based simulations are being applied to social science (Davidsson, 2002). Evolutionary game theory models strategic interactions as a dynamic, self-organizing system. The dominance of a single optimal strategy emerges from the population which consists of the individual strategies being played against each other. The state of a best strategy dominating the population is an *attractor*.

2.6.1 Attractors

During the lifespan of a self-organizing system it *evolves*, which involves changing states. Assuming that the system is deterministic makes some states are more likely to end up in than others. In deterministic systems a state a can only lead to a single state b. When the system changes between states without outside influence, it will eventually end up in an *attractor*. An attractor is a set of states from which the system can never leave; one can say that the system has "closed in on itself" (Heylighen, 2001). An attractor can consist of one or more states that repeat in a cycle. A *basin* is a property of an attractor, and is defined as the set of states outside of it that necessarily end up in the attractor.

Chapter 3

State of the Art

This chapter intends to explore the current scientific research related to the project. The main topics are agent-based models and evolutionary game theory, since the simulation is agent-based, and the project is rooted in an evolutionary game theoretic question.

3.1 Agent-Based Simulation

At the intersection of computer simulation in general, social sciences and agent-based computing one finds agent-based social simulation, according to Davidsson (2002). In this combination of the disciplines, the thinking from each affects the others. From the view of social scientists, the introduction of computer simulation enforces constraints on their theories, forcing them to be clearer about assumptions, boundaries and input-output. Additionally, when models from social science are implemented in an agent-based way, one can study which properties of the overall model emerge from simple agent-based interactions. In this way, methodology from agent-based systems can aid social scientists in understanding complex social systems from a bottom-up perspective. The fields of computer simulation and agent-based models affect contribute from different angles as well, since computer simulation traditionally rely on mathematical generalizations of a number of agents or entities as a whole, while agent-based computation techniques concentrate on the individuals and global behavior that emerges from their interactions.

There are several platforms created for multi-agent research, and Le Page et al. (2012) studies the community and usage of CORMAS, an agent-based simulation platform. It was created as a tool for multi-agent system, especially those with resource dynamics. Thus it was quickly used for situations where sustainability is a goal. According to the study, 3 % of models developed with the tool are fishery-related, with water being the primary resource with 34 % of the models. The article concludes that many applications created with CORMAS help teach stakeholders about consequences and viewpoints with regards to the resources at hand. They also remark that the focus over time has shifted from research on renewable energy situations, to using CORMAS simulations as a teaching tool.

REEFGAME represents one of the simulations developed with CORMAS that function as a teaching tool about resource sensitive situations (Cleland et al., 2012). The simulation is a game where each player is a fisher, who has different attributes that differentiates them from other fishers. The players can communicate during the game, but it comes at a cost, which introduces asymmetry of information. The game world is divided into a grid of cells, where each ocean cell may have resources like coral reef and algae. If a resource is worked too much, it dies. In addition to choosing different locations of fishing, the fishers may also try to pursue other kinds of work. Players were observed to choose strategies that prioritized safe but satisfying choices rather than risky and potentially very profitable ones. Two main goals with the program was to gain a better understanding of the fishers' situations, as well as finding an answer to how policies can be formed that promote conservation of natural resources. One of the most important motivations for using a game to perform social experiments was that the game can provide a situation with fewer unrealistic constraints than a controlled experiment, such as structured communication. During the game, participants can talk among each other freely. The combination of a game simulation with a social experiment was therefore a valuable tool.

Another agent-based simulation related to natural resources that was applied as a teaching tool was studied by Rebaudo et al. (2011). They modeled how human interaction influenced pest infection. The potato moth is an invasive insect in Ecuadorian farms, and it is spread by farmers visiting other towns to sell their infected product. Farmers can hinder the spread of the infection by traveling less or changing their patterns. Thus they end up in a common-pool resource problem where everyone has to avoid spread, contributing to the common good. Knowledge of the interactions and pest dynamics is important, so the simulation is used as a teaching tool to inform farmers on how to more optimally move. The simulation is structured in several levels. First of all there are the agents, the farmers, which interact in an agent-based fashion. Secondly there is a separate layer for modeling the pest spread, utilizing a cellular automaton.

A more traditional application of CORMAS, where the simulation is used for a study instead of teaching, was studied by Janssen and Ostrom (2006). The situation studied involves a community of fishers that contrary to the game theoretic prediction have created a society where agents exhibit policing behavior on their own initiative, without direct profit as a selfish goal. The simulation includes agents who suddenly experience a resource shortage, which gives the opportunity to create a maximum harvest-rule which can be enforced by voluntarily policing agents. Agents who harvest more than the limit specified by the rule are punished if discovered by the police. However, the Nash equilibrium of the scenario is that no agents contribute to the police effort, and agents end up cheating which causes a resource collapse. Through spatial structure restricting harvest freedom and separate trust games, the agents in the simulation manage to build trust and confidence, and still evolve a rule-enforcing society.

3.2 Fishing and Aquaculture

The various stakeholder's reactions to aquaculture was studied by Tiller et al. (2014). Fishermen, foreign workers, tourists, high school students, aquaculture representatives and researchers were all present in the study, which focuses on charting the different motivations and priorities of each group as a whole. The situation in Frøya, where the study concentrated, is extraordinary since fishers and the municipality are very accepting of the aquaculture industry. This contradicts the more common situation where local fishermen are heavily opposed to aquaculture, both environmental damage and sturdy competition from the facilities. The technique applied was stakeholder workshops, which is a meeting where representatives of one of the group talk about the situation using a methodology for identifying important aspects. It was revealed that while fishermen are positive to aquaculture, they are most concerned with the continued existence of fishing as a commercial industry, meaning that it should not be out-competed by aquaculture.

Combining aquaculture and agent-based simulation, Pereira et al. (2004) study a way to simulate ecological systems that include intelligent agents. A general problem among ecological models is that they employ mathematical models that are very straight-forward. The use of intelligent agents is therefore contrasted with more traditional models of natural resource scenarios which are implemented using simple mathematical equations that don't consider the intelligent decisions of humans, and the influences of these decisions. They employ a specific messaging system that can be changed for different application domains.

Later the same group present a specific system called *EcoSimNet* for simulating especially aquatic ecosystems (Pereira et al., 2009). The system can be used to combine the actions and desires of intelligent agents such as humans, with ecosystem dynamics. The optimization process works by running many simulations in parallel and in several generations sequentially. Results from each generation are shared and used in the next generation, creating a system that is similar to artificial evolution. The platform was used to implement a system for finding the best aquaculture configuration for farming shellfish on a coastline. The use of parallel computations allow the time necessary to compute solutions to be drastically decreased. *EcoSimNet* is shown to be a capable platform for optimization and agent-based systems, specifically decision support systems.

Cruz et al. (2007) implement a simulation system that places an intelligent aquaculture agent within an environment with the goal of finding the optimal configuration of farming sites within an aquaculture area, with respect to farming output. The agent operates in a multi-agent environment with many ecological and environmental factors such as tide and water quality. The agents communicate with each other and the environment using a high-level communication language. Four optimization algorithms are developed to search for the optimal configurations: simulated annealing, tabu search, artificial evolution and reinforcement learning. The tactic of using these optimization tactics to find the solution is shown to be more effective than using mathematical models.

Since lice infesting salmon is a problem for aquaculture facilities, Groner et al. (2013) build an agent-based model for simulating the spread of lice in a population. The simulation includes wrasse, which is an organism that preys on the sea lice, and running the simulation explores how best to apply wrasse to the salmon. The lice and wrasse dynamics are both made up of probabilistic events, introducing stochastic effects to the model. The model parameters are based on literature where data was available. The use of an agent-based model to investigate sea lice infestations is reported to be original to the study, and has the advantage that the model can incorporate any kind of static and dynamic events, in contrast to more constrained models. The model concentrate on the sea lice themselves, while both the salmon and wrasse behavior are simplified significantly. This shortcoming was overcome by repeating the simulation many times, in order to filter out stochastic effects that result from lack of information.

Chapter 4

Conceptual Design

This thesis focuses on the decision of fishermen facing aquaculture expansion, and the scenario is investigated mainly through a simulation. The simulation is meant to cover the most important part of the fishermen's decisions, while abstracting away unimportant details. This chapter describes on a high level what aspects are included in the simulation.

The fisher agents make decisions whether to vote or not based on information about the environment and the coastal plan. The decision is predicted through game theoretic analysis to always result in the maximum number of complaints. Information about the environment is communicated between agents using a messaging system. The decision to complain or not applies an artificial neural network, which is adapted through the application of artificial evolution. The learning system is modeled after evolutionary game theoretic social learning, where agents copy their best-performing peers.

4.1 Stakeholders

Identifying the different people involved in the situation is crucial to creating a complete simulation. The stakeholders deemed relevant to the simulation are based on the stakeholders used by Tiller et al. (2014). The stakeholders are implemented as agents in the system.

4.1.1 Fishermen

Fishermen are of course vital to the system which tries to explore fishers' decisions. Fishers are the agents who vote on coastal plans by filing complaints. They also harvest fish from the ocean, which may conflict with the aquaculture facilities.

4.1.2 Aquaculture Organizations

Aquaculture facilities are built by organizations. These organizations decide where to establish a facility, apply for a license from the municipality, and do the building. After a facility is built, the organization runs it to harvest its output.

4.1.3 Municipality

Coastal plans are created with semi-regular intervals, and they are created by the local municipality. The municipality is also responsible for reworking coastal plans in case they need to be, based on complaints. When aquaculture organizations pursue an area for aquaculture expansion, they apply for a license at the local municipality.

4.1.4 Government

There are several functions performed by government instances that are grouped together as a single “government”. These functions include receiving complaints, deciding if a coastal plan needs reworking and providing the municipality with aquaculture licenses. Grouping these governmental functions that may be performed by separate subsidiaries gives an easier overview of the model.

4.1.5 Other stakeholders

In the paper by Tiller et al. (2014), several stakeholders were interviewed regarding their attitude towards aquaculture expansion. These stakeholders include local high school students, foreign workers, enthusiasts, tourists and academics. One could consider high school students, foreign workers and enthusiasts and other interested locals a single group, together forming a local community of civilians. This group has strong opinions on the aquaculture and should not be excluded from a complete analysis of the consequences of aquaculture expansion. Aquaculture is mostly a source of wealth for the locals. Separate from civilians, tourists also play a role in voicing their opinions on aquaculture expansion. Since tourists are concerned with natural beauty and cultural heritage, such as the fishing industry, tourists can reasonably be assumed to overall be against aquaculture expansion. In fact, tourists often have a preferred area where they don’t want aquaculture expansion.

The stakeholders of civilians and tourists will be considered in the design of the simulation, but they will be excluded from the simulation itself in order to keep it as simple as possible. The focus of this thesis is answering questions about the complaining behavior of the fishers themselves. Even if support from the local community for aquaculture or opposition from tourists which coincides with fisher complaints may have an impact on the perceived effectiveness of fisher complaints, this effect is not the focus of the project and is therefore assumed to be minimal. A potential future development of the project topic could be to explore the effects of including other rich stakeholders, and this possibility is discussed further in chapter 10.

4.2 Sequence of Events

There are several events that are important to the fishermen’s choices. To define an order, or at least prerequisites for their occurrence, is crucial to the simulation since it means defining the framework for the whole scenario.

4.2.1 Definition of the Events

The most important events are captured below, with their contents described. A priority was to define events with as concisely as possible, at the same time as making the set of them capture everything that's necessary to make the simulation realistic.

Coastal Planning

The municipality's plan for use of coastal areas and resources is definitely very important for the fishermen, since it's the document that dictates where aquaculture can be established, and where there are protected fishing areas. It occurs at a somewhat periodic interval. With each new coastal plan, fishermen get the opportunity to influence it through complaints.

Hearing

Before the coastal plan is finalized, it goes through a hearing which is when the fishermen and other stakeholders can influence it through complaints, or perhaps suggestions. The hearing is done in three stages. If there are complaints to the municipality for a coastal plan, it is first treated by the first stage. If no consensus is reached, the plan is reworked and the hearing goes to a second stage. After the third stage a plan is forced through, even if complaints still remain. The hearing event may be considered a part of the coastal planning event.

Fishing

One of the most important part of fishermen's lives is the fishing itself. For the most part, fishermen stick to the same spots where they return to fish trip after trip. Important aspects to consider with regards to the simulation are: *a*) the amount of fish a fishing spot yields; *b*) the fishermen's knowledge of good fishing spots; and *c*) negative influence from aquaculture facilities on the amount and quality of fish.

Building

Aquaculture being built is an important event that influences the fishermen directly, by potentially pushing them out of their territories. Fishermen can of course not fish inside an aquaculture facility, but there is also a blocked radius around the facility where fishing cannot be done. This radius is typically around 100 m. During the workshop, which is elaborated in chapter 7, it was clarified that the establishment of an aquaculture facility has important environmental consequences for the surrounding area, namely it gets polluted with chemicals and excrements. This reduces the natural fish population in a large radius around the facility, and what is left is of poor quality.

In order for an aquaculture company to receive a license to build in a certain location, they first have to wait for the national government to distribute licenses to the municipality, and then they can apply for those licenses based on the coastal plan. Since the focus of this project is the fishermen's decisions, and not the aquaculture companies' process towards building facilities, this process can be joined with the building process. In this way, when the municipality receives licenses from the government (which may happen periodically), they are immediately given to aquaculture companies who immediately establish in any area where aquaculture is allowed according to the coastal plan.

An aspect to consider of the previous two events is that fishing and building are typically intertwined; that is fishing takes place both before and after aquaculture is built, and all aquaculture is not built at the same time. A way to simulate the first of these two aspects is to have a fishing event both before and after building, so that consequences of fishing can be seen both with and without the new aquaculture facilities, in a single planning round.

4.2.2 Defining the Sequence

The most straight-forward way to structure the events in the simulation, is as a defined sequence with conditional loops, that cycles until the simulation is over. For this simulation, it means that when the government decides that a plan should be reviewed, the sequence goes back to a planning stage, before a new hearing round, and so on, until the government decides to approve the plan. After that happens, the events not part of this cycle takes place, before the full cycle is over and a completely new coastal plan is made.

Another way to structure the events could be a more open-ended world where fishermen do their normal activity (i.e. fishing) while the municipality "suddenly" creates a coastal plan and the normal activities are interrupted by, or continue in parallel with, hearing, government complaint approval, and aquaculture establishment. This structure may be more realistic and reflect closer how things actually work both in Frøya and other fisher-aquaculture communities. However, it is more complicated both to build and analyze, since the time can vary between each coastal plan and subsequent aquaculture establishment. This means the consequences felt by the fishermen would change with not only the severity of the establishment, but also by the amount of time before the new planning. If two plans were made in rapid succession, the fishermen may not have time to experience too severe consequences from an aquaculture facility that actually did destroy a significant amount of natural resources.

For the sake of simplicity, the simulation is designed with a linear sequence that can jump back to the planning stage if the plan is decided to be reworked. Figure 4.1 shows the sequence as pseudocode, including the processing of events (as "Compute" statements of the form "Compute EVENT NAME(*information*)") and information processed and produced by the events (as variable assignments of the form "$Name \leftarrow Value$"). The jump-back effect is controlled by a while-loop that iterates until an approved coastal plan has been finalized, computing coastal planning and hearing.

4.3 Decisions

The ability to make autonomous decisions are what define the agents as intelligent. There are several kinds of decisions made in the system, including fishers complaining, the government approving and rejecting complaints, and the municipality creating the coastal plan.

```
while simulation is running do
    $Approved \leftarrow False$
    $Coastplan \leftarrow$ NULL
    while $Approved$ is $False$ do
        $OldCoastplan \leftarrow Coastplan$
        Compute CoastalPlanning($OldCoastplan$)
        $Result \leftarrow$ result of CoastalPlanning stage
        $Coastplan \leftarrow$ coastal plan from $Result$
        Compute Hearing($Coastplan$)
        $Result \leftarrow$ result of Hearing stage
        if coastal plan is approved in $Result$ then
            $Approved \leftarrow True$
        end if
    end while
    Compute Fishing
    Compute Building($Coastplan$)
    Compute Fishing
end while
```

Figure 4.1: Pseudocode showing the sequence of events

4.3.1 Different agents' approaches to decision making

The different kinds of agents have different roles, and therefore perform different decision making. Differences in decisions may also result from different priorities.

Fishermen

Fishermen decide whether to complain about single plans to establish aquaculture at a certain location. This decision is based on multiple factors. Fishermen have a "home" area where they currently do their fishing. At the beginning of the simulation, this area is assumed to have plenty of fish, and the fisher using that field has been using it for a long time. The home area can change during the simulation, either because the spot degrades in fish quantity or quality, or because the area is blocked by aquaculture expansion.

One factor is the distance between the threatened area and the "home" area of the fisherman. This distance is important because aquaculture facilities produce waste and the waste damages areas closer more than those that are far away. There is also a psychological factor, where a lot of aquaculture close to the fisher's area would feel more threatening than if they were far away.

Another factor is the conditions of the home area. A fisher that has good conditions in his or her home area may not care if a spot on the other side of the map is threatened by aquaculture. However, if the facility threatens to destroy or block the home area and that area is really good, a fisher would want to protect it because it would be very difficult to find another spot that is just as good. However, if the home location is relatively bad, it may not be worth it to try to protect it even if the facility threatens to block it, because it may be easy to find a new spot that is just as good.

The last factor to consider is the conditions of the area that is threatened by aquaculture expansion. A better fishing spot may be considered more valuable and therefore more deserving of protection in order to preserve the community's ability to fish also in the future, even if no fishers are currently using it. If the spot does not provide good yield, it may not be worth the investment for fishers to complain about it.

Complaining has zero monetary cost, but there is an associated *information* cost when agents reveal their good fishing spots by complaining about them. This cost is not an explicit part of the decision making process, but rather a consequence of complaining, and thus this information sharing cost is learned through experience.

The output is merely the decision to complain or stay silent about the aquaculture establishment.

Another decision fishers make is changing to another fishing location. Fishers change their locations for two reasons: either they are pushed away from their current fishing spot because of aquaculture expansion that blocks it; or they observe that their current spot has so low quality that they try their luck somewhere else. Fishers have a constant measure of what constitutes a location that is so bad that they have to move, and this measure is equal to the performance of the average location. When changing, fishers sort their knowledge of other locations and try the one they think is the best of those. If they don't know any locations that are better than what they think is the average location, the fisher will try another random non-blocked fishing spot.

Government

The government makes decisions on what to do with complaints. There is a negotiation process going on where the government is in the middle of those for and against aquaculture. Technically all stakeholders may be either for or against establishment of aquaculture. Especially tourists may be against aquaculture because they are concerned with the cultural and natural experience of the place. Still, this simulation focuses on fishermen, and other stakeholders will be assumed to always approve of aquaculture being built. The inputs to the government's decision mechanism are the votes by each agent.

4.3.2 Voting

Both fishermen, aquaculture owners, civilians and tourists can vote or voice their opinion on the establishment of an aquaculture facility. These votes will be considered by the government when making the decision to allow or deny the facility. Different kinds of agents care about different aspects of the building, and they have different information. Therefore the information used to make decisions is different for each type of agent. Table 4.1 shows an overview of the influences for each agent.

Voting Procedure

There are several possible ways to do the voting itself. Since complaints from fishermen carry knowledge as discussed in section 4.4, it would be unfair to do the voting in a cycle, asking each agent in turn, because the latter agents would have the advantage of more

Table 4.1: Inputs for the voting decision for each agent

Agent type	Inputs
Fishermen	Distance from the targeted location to the home location Quantity of fish in the targeted area, if known (otherwise it is assumed that the location is bad) Quantity of fish in the home location
Aquaculture owners	Quantity of aquaculture facilities already on the map

information. Another option is to do the voting simultaneously which would leave all the fishermen with the same opportunity of information at the time of the vote. However, this is not very realistic since in a meeting where fishermen voice their opinion, some would speak before others, which would mean that the later voters gain more information. In this case, however, the order of votes is not decided by an external regulator, as would be the case if it was implemented as a set order, but by other factors such as motivation and personality. This behavior could be reflected in the simulation if agents have varying degrees of *conviction* for their opinions. This way the regulator (government) could ask the agents to state their opinions in order of most convicted of one option, to least decided. This way latter agents could potentially use the new information and adjust their decision.

4.3.3 Influences

Decisions are influenced by agents' priorities. As such, the agents make decisions to maximize fulfillment of their priorities. The correlation between a decision and its subsequent fulfillment of priorities, or consequences, is made individually by each agent as a learning process.

4.3.4 Learning

A background for the project was a study of evolutionary game theory, which can be interpreted as social learning. With an evolutionary game theoretic view, agents should be able to learn and adjust their decision making when they perceive that some agents which make some decisions are more successful than others who make other decisions.

The simplest evolutionary game theoretic application of learning is having agents imitate the strategies of their best-performing peers with a probability proportional to the difference in fitness (payoff). Such a solution is possible if the mechanism is equivalent to a game theoretic strategy, pure or mixed. However, since the simulation calls for a more sophisticated technique, more sophisticated learning is necessary. A common way of applying learning to neural networks and other classification algorithms is artificial evolution, which involves separating the mechanism into genotype and phenotype. The genotype is a simple (often binary) representation of the mechanism, which can be subject to mutation operations such as crossover and bit mutation. The phenotype is the actual mechanism built from the genome description. Agents can learn by copying the genotypes of their most successful peers.

Learning and the decision mechanism

Learning is an important aspect to decision making. Some decision making mechanisms are more suitable for learning than others. For example, neural networks can be easily implemented on top of an evolutionary algorithm where genotypes represent weights in the network. In this case the evolutionary mechanism of selective reproduction can be replaced with selective imitation, consistently with the social learning interpretation of evolutionary game theory. On the other hand, a ruled-based mechanism can be more difficult to adapt over time.

The positive benefits of not applying a learning method are not to be underestimated. An argument can be made that the motivations, variety of agents and decisions make the simulation sufficiently complex that learning is not necessary to produce valuable results. For instance, fishermen could be initialized with simple rules where some always vote no to new aquaculture establishment, and some always say yes to the establishment. Analyzing the performance of these two types of agents in relation to each other could be sufficient to give an idea of why some fishermen in the real world don't complain when a quick analysis shows that the optimal choice would be to do so.

There is still good reason to include some form of learning for the agents. Real people learn and adapt to their environment constantly, and it is realistic to think that the fishermen of Frøya do the same; if complaining about voting seems to give higher profit to those fishermen who do, others will follow suit. Similarly the simple agents discussed previously can observe and copy other better-performing fishermen. Therefore these simple agents can do learning, but the mechanisms that they learn are only simple strategies.

Real agents do actions and make choices based not only on blind copying, but depending on many factors. A neural network where the weights are based on a genotype subject to social evolution can emulate this complexity better than a strategy that chooses either A or B. The point here is that an agent that copies the over-arching strategy will not consider its own conditions compared to the conditions of the agent it is copying, while if it copies the weights of a neural network it can copy more under-lying behavior and rather copy the other agent's better-performing "intelligence".

4.4 Knowledge

When sophisticated decision making techniques are used, knowledge is very important. Knowledge can be either public or private, and shared through communication.

The knowledge and assumptions of fishermen are mostly tied to the quality of fishing spots. Fishermen agents are initialized with a single "home" cell, in which they know the quality of fishing. Knowledge of other spots must be obtained through inferring form messages or trying out other fishing spots. This knowledge is used when fishers locate new fishing spots. The knowledge that a fishing spot exists comes from inferring from other agents' complaints, but knowledge of specific quality of that resource can only be obtained from fishing there.

4.4.1 Inferring Cell Quality from Complaints

Agents perform an estimate of cell quality based on complaints about that cell. The baseline for the guess is slightly above average, with some random variation. Without random variation for this guess, all fishermen will guess the same value for each cell and they will always choose the same cell to migrate to if their home is occupied. Introducing variance in the guess makes the behavior more realistic where fishermen will migrate to different spots.

4.5 Priorities

Different types of agents have different priorities. The main priority for profit-making agents such as fishermen and aquaculture facility owners is of course profit. Some agents have apparently contradicting priorities, such as fishermen who want profits which can be negatively influenced by aquaculture, at the same time as they want some amount of aquaculture in order for the local community to function well. Table 4.2 shows an overview of the priorities for the different types of agents.

4.5.1 Profits

Working agents need a living, so the main priority for most agents is their own profits. For fishermen, this means how much fish they can harvest and sell at a good rate. For aquaculture agents, this means the harvest rate from their facilities combined with the market rates. For other agents, civilians and tourists, they have their own incomes that are irrelevant of the simulation, so they don't care about their profits. Measuring profits is as simple as looking at the agent's income. Regularly resetting their capital can turn it into a measure of income.

4.5.2 Wealth of the community

Increasing wealth in the local community has positive effects on all members. It can mean better infrastructure, better schools for children and in general a more healthy community with more people, that is more attractive to people from the outside. Frøya is a magnet for foreign workers, which is a symptom of the growing opportunities there. Community wealth benefits everyone, and it is the main reason why fishermen do want and accept aquaculture in their waters, since aquaculture facilities are very profitable. Measuring community wealth can be done by an average or sum of community member's wealth.

4.5.3 Existence of fishing as a business

If fishers can't keep a stable income by fishing, it is natural that they have to seek other livelihoods. The main priority found by Tiller et al. (2014) was that fishermen want the continued existence of a commercial fishery at Frøya. The fisheries have cultural significance, and are also healthy for the community, because it provides more diversity in the local industry. Other agents that want fishing to continue are civilians and tourists. Civilians need fishing to exist because they have traditional ties with the industry, as well as a

Table 4.2: Overview of priorities

Agent Group	Priorities	Influences
Fishermen	Profits	Market rates Quantity of fish in fishing spots
	Wealth of the community	Money from aquaculture going back to the community Fishermen as part of the community having profitable jobs
	Existence of fishing as a business	Fishermen having profit
	Maintaining natural fish stocks in good health	Fishing which depletes natural fish stocks Disease spreading in the population, naturally or from aquaculture populations
Aquaculture companies	Profits	Quantity of fish in aquaculture, influenced by disease Market price of fish Receiving permission to build aquaculture facilities
The local community (civilians)	Wealth of community, often through locally owned aquaculture	Profits of each individual community member
	Existence of both aquaculture and fishing industries	Profitability of the industries
Tourists	Existence of fishing industry	Profitability of the fishing industry
	As few as possible aquaculture establishments	World map cells prioritized by the individual tourist
	Wealth of community	Profitability of locally owned aquaculture
Government	Pleasing voters	Fulfilment of voters' priorities
	Pleasing interests of investors in the local community (aquaculture owners)	Aquaculture establishments running well

high school education dedicated to fishing. For tourists it is a matter of cultural heritage to keep the fishing industry intact. Measuring the existence of fishing as a business can be done by counting how many fishermen still do fishing, if they are given the opportunity to quit. If they aren't the health of the business as a whole can be measured by the average capital of each fisher.

4.5.4 Maintaining natural fish stocks in good health

A key part of maintaining a stable fisher's income is that the natural fish resource is in good health and quantity. Their health can be diminished by aquaculture facilities, as discussed in chapter 7. Bad fish health also evokes disgust and bad feelings in the fishers. Measuring natural fish health can be done by counting the amount of fish in each area in the simulated world.

4.5.5 Existence of aquaculture industry

Aquaculture's continued existence is important to the community not only because of its overall profitability, but also at a more personal levels since many community members have work in aquaculture facilities. Even agents who work other places may be tied to the aquaculture in other ways, for instance at the slaughterhouse. Another option is that a spouse or close family member works at an aquaculture facility. Therefore the continued existence of the aquaculture industry is important to almost all community members. Since aquaculture facilities never disappear in the simulation, their existence can be measured by looking at the cause of why they would disappear: if they aren't profitable enough for the owners. This is a function of the average profit of each aquaculture facility.

4.5.6 As few as possible aquaculture establishments

Some agents want less aquaculture, and some want as little as possible. Tourists typically have some areas that they care about, probably close to their travel destination. Aquaculture in these areas will be perceived very negatively for the tourists, as the motivation to travel to these areas is more often than not seeking less industry and more nature. Since the aquaculture industry also can prey on the fishing industry as a whole, the negative impact of aquaculture is even greater for the tourists since they care about the cultural heritage of Frøya and therefore want the fishing industry to be healthy, as it always has been. Measuring this priority can be done by tourists having a set of areas that they care about, and checking if these areas have aquaculture facilities or not.

4.5.7 Pleasing voters

Like all democratic government, the municipality wants to please voters by making a coastal plan that everyone is happy with. Community happiness can be measured by the average priority fulfillment of all community members.

4.5.8 Pleasing interests of investors in the local community

Investors are important to the community's wealth, so pleasing them specifically may be an extra requirement. Investors are typically the aquaculture owners. Measuring their level of satisfaction may be done by averaging the priority fulfillment of aquaculture agents.

4.6 Information Flow

Information is crucial to the decisions of fishermen and government, and for aquaculture organizations when choosing a location, as well as the municipality for planning. The flow of information is a description of how the agents in the system communicate; who says what to who.

Figure 4.2 shows an overview of information flow in the system. Information is communicated between agents through a basic messaging system. Starting with the municipality, the first message to be sent is a declaration that a new coastal plan has been created, and this message is sent to all interested parties, which includes fishermen, aquaculture organizations and other stakeholders. These agents are given the opportunity to vote using complaints, which produces more communication from these agents to the entity to which they complain; in this case the government. In addition to the complaints themselves that are sent to the government, they are also observed by fishermen, who gather information from the complaints of other fishermen. After the government has processed the complaints and come to a conclusion on whether to approve or reject the coastal plan, one out of two messages can occur. Either the approved *complaints* are sent back to the municipality with the intent that the coastal plan is reworked. The other option is that the approved coastal plan is sent to all the stakeholders, signifying that it is final. After a final plan has been created, aquaculture organizations will apply for aquaculture licenses based on the plan, and these licenses go to the municipality.

4.7 Data Set

The data going into the simulation is the configuration of parameters used in the simulation. A different configuration can be made to reflect different situations. For example, the government can be tuned to be very approving of fishers' complaints, or to often deny them, in order to reflect real governments that are skeptical or open to aquaculture, respectively.

4.8 Outputs and Results of the Experiment

The simulation produces behavior in the agents. This behavior is observed directly, and functions as output of the simulation. For instance one can observe the amount of complaints done by fishers, and how often the government approves complaints. Another output is how the world changes when aquaculture facilities are built. A map can be drawn representing the state of the world.

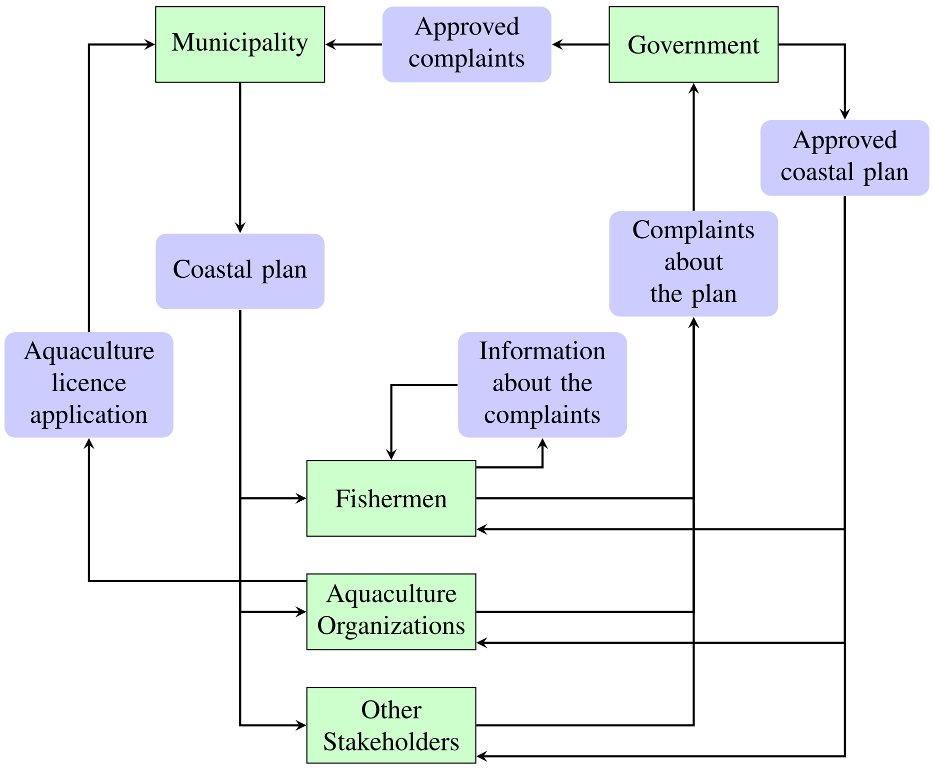
Municipality
Approved complaints
Government
Approved coastal plan
Coastal plan
Complaints about the plan
Aquaculture licence application
Information about the complaints
Fishermen
Aquaculture Organizations
Other Stakeholders

Figure 4.2: Information flow diagram for the relevant stakeholders. Agents are represented by green boxes, while pieces of information are blue. “Other stakeholders” include civilians and tourists.

This thesis uses simulation experiments to answer questions about the behavior and motivation of the fishermen. The simulations are defined with the goal to give insights to answer the research questions. More specifically, the simulation should confirm or the hypothesis regarding fishers' payoff matrix, represented in fig. 1.1 and discussed in the introduction.

4.9 Classifying the Simulation

Corresponding the environment properties an agent-based system can have, as elaborated in section 2.1, the simulated environment is:

Partially observable Fishermen have to experience the fishing quantity in order to judge a fishing spot, and the information they gather from this may be incomplete if there are more than one fisherman working the same spot, which reduces the output, but does not diminish the "real" value of the resource.

Stochastic There are some a few random elements of the design, such as fishers' guesses about resource quality from complaints, and the aquaculture owners' choice of place.

Multi-agent The system includes multiple fishermen, aquaculture owners, government and municipality, and is therefore highly multi-agent.

Dynamic Since aquaculture establishment is a consequence of not only one agent, but all the complaints and approvals that were made, the process is dynamic.

Mostly episodic The system consists of a set sequence of events, which would make it episodic. However, the government's decision decides if the next step is a renewal of the plan or a continuation of the sequence, so that makes the simulation sequential.

Discrete There is a finite number of possible states for the system which consist of a discretely divided map and discrete time events such as the event sequence.

Mostly unknown Agents may have knowledge of the consequences of not complaining, but this knowledge is mostly gathered from experience while complaining or non-complaining behavior is learned.

Chapter 5

Experimental Setup

The experiment is done as a simulation and the results are analyzed in light of evolutionary game theory. This chapter describes the setup, which includes details regarding the simulation setup that are not implementation-specific. This includes a clearly defined process, a description of how decision making is handled, a description of the learning system, a documentation of agent communication, documentation of other aspects, handling of priorities, clearly defined actions by the various agents and details on how to run the experiment.

5.1 Simulation Overview

This project uses experiments to gather insights into corresponding real-life situations. The experiments are conducted with a simulation which is constructed as a thick agent-based simulation with many agents that exhibit complex behavior. This behavior is arranged in a sequence of events. Figure 5.1 shows an overview of the process, which continues a number of rounds.

Each round is initiated by a coastal plan being laid out by the municipality. The main priority of the municipality of Frøya is to have aquaculture where it is possible, so in the beginning as many spaces as possible will be reserved for aquaculture. The plan is sent to all “voting” agents, which includes fishermen, aquaculture owners, tourists and civilians. After the plan has been distributed, the *hearing* phase starts. Each voting agent can issue a predetermined number of complaints about aquaculture establishment on cells in the plan. The complaints from the hearing are considered by the government. If one or more of the complaints are approved, the plan goes back to the municipality for review, which triggers a renewed planning, another hearing and another government decision. This cycle can end up going indefinitely so there has to be a mechanism to prevent an infinitely long planning stage. When the plan is approved, normal activities are pursued by all agents, as indicated by the “fishing” activity. At some point the government will decide to distribute licenses and aquaculture building can begin. A number of aquaculture agents will build on spots allocated in the coastal plan for aquaculture, where this number is restricted by

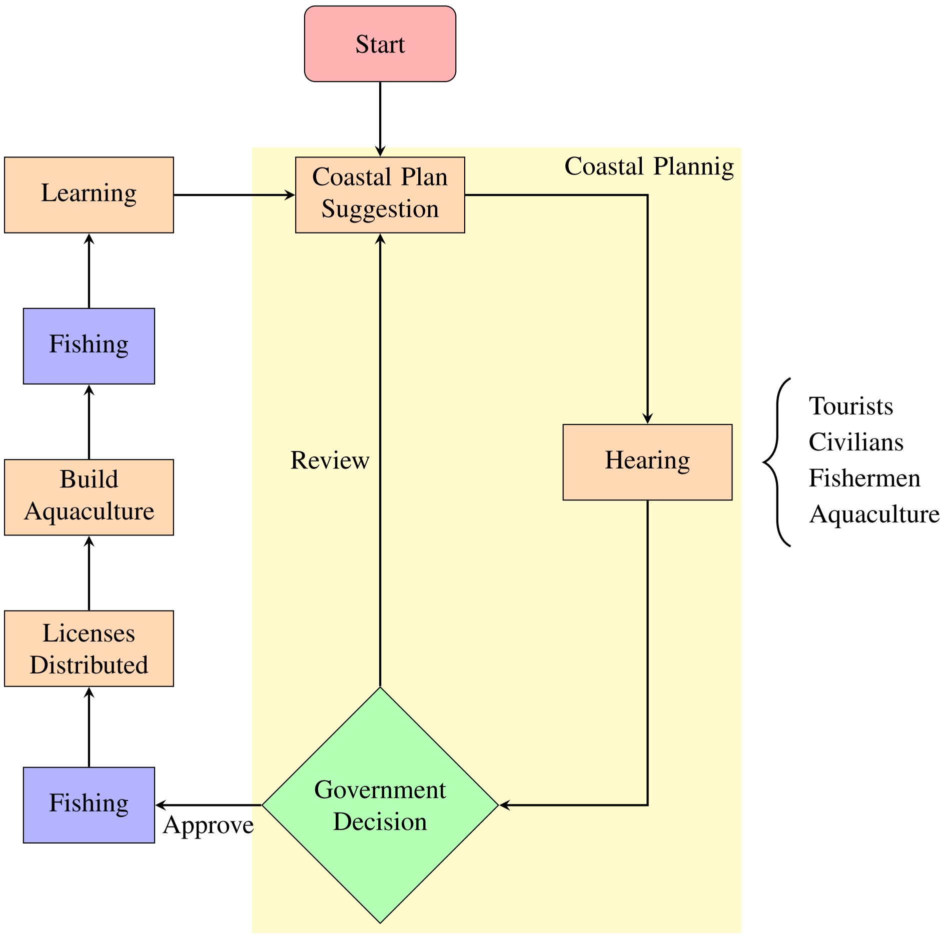

Figure 5.1: Process chart of the simulation. The government decision has two options: To *approve* a coastal plan, or decide that it has to be *reviewed*.

the number of licenses issued by the government. After aquaculture has been built, normal activities continue until it is time to lay out a new coastal plan. Before a new planning phase is started, there is a learning phase where agents adapt to decisions and observed consequences.

The simulation is run for a predetermined number of rounds. Profit is calculated differently for each type of agent.

5.1.1 Phases

The simulation consists of four main processes, a decision and a repeated activity. The four processes are, in order: coastal planning where the municipality puts forth a suggestion, a hearing where stakeholders review the plan, licensing and building of facilities, and learning. The decision consists of the government taking all the votes into account and through its own decision making process that depends on the priorities of the government deciding whether or not individual complaints are approved. The recurring activity named "fishing" is profit-making activities that the different agents normally do.

Process 1: Coastal Plan Suggestion

A suggestion for a coastal plan is presented by the municipality. Initially it typically includes every spot possible to establish aquaculture in. If a plan is denied by the government, this process handles each approved complaint and removes them from the list of sites reserved for aquaculture.

Process 2: Hearing

All non-government agents participate in the voting. The vote consists of each agent voicing their opinions either way on the establishment of the new aquaculture agent's facility. Knowing the vote of others might be crucial to the decision to vote, so the voting process may either be done in a number of rounds, simultaneously, or another protocol. For example, there could be a time period where votes are accepted and agents with a higher incentive would vote first. Then as time moves on the other voters gain more information from the first voters so they will gain more incentive to vote sooner. At the end of the period, agents who haven't voted will be prompted to give a final vote. Other protocols may also be feasible.

All complaints are redistributed to all voting agents by the government when they are received. Thus all votes are public and agents can infer information from them. When a complaint about a cell is issued, agents who observe that complaint will make certain assumptions about its location. Since fishermen are more likely to complain about locations with lots of fish, in order to protect the good fishing spots from aquaculture expansion, agents assume that locations that are complained about have a value that is slightly above average. This guess may be subject to some random variance.

Decision: Government Decision

The government decides whether to allow or deny the establishment of the given aquaculture facility. The decision mechanism might be that allowing it requires unanimous vote, a simple majority vote, or any other mechanism that is designed to satisfy the government's priorities. For example, a likely scenario is that new aquaculture agents offer a monetary compensation to the government or the local community when establishing. Such a compensation will influence the government decision.

The government decision takes place on three levels, reflecting the different institutions the hearing goes through in the real-life situation.

Process 3: Licenses and building

At the beginning of this phase, a predetermined number of licenses is distributed to the municipality. For each license, an aquaculture agent is created and chooses a cell reserved for aquaculture in the plan. An aquaculture facility is built on this location.

When the facility is built, the cell is blocked for fishing. The blockage also affects all cells in a radius around the facility. This radius is variable, but a typical value will be 100 m. Fishermen cannot fish in blocked cells, and new aquaculture facilities cannot be established either.

Any spawning fish population in the cell where the facility is built will be destroyed as a result of the building process.

Process 4: Learning

Between rounds, agents adjust their decision making protocol based on observations of their own and others' performance. For instance an evolutionary algorithm interpreted as social learning can adjust the weights in neural networks so that fishermen are more likely to complain, if they observe that fishermen who do complain end up with a larger profit.

Activity: Fishing

The agents do their normal profit-generating actions twice in each round. For fishermen this involves fishing and selling the harvest, while aquaculture agents produce and sell farmed fish. Civilians receive a small fraction of the profit of locally owned aquaculture facilities, to reflect enrichment of the local community. Taxes on revenue are also paid during this activity.

For fishermen, the fishing activity is divided into two parts: locating a fishing cell, and the fishing itself. Fishermen know about a set of locations, with one being their "home" location. In the first phase they will relocate to a new home if they know of a location that has better yield. If all known location, including the home cell, give a worse yield than an average cell, they will relocate to a random unblocked cell on the map. New locations are known through the votes of other fishermen. In the second part of the fishing phase, fishermen will harvest fish from their home location. If several fishermen are located on a single cell, they split the yield evenly. The fisherman's knowledge array is updated with the location's observed output.

5.2 Decision Making

An important feature of the agents in the system is intelligence, which can be defined as the ability to make decisions. The most important decision in the simulation; the central decision that is being investigated, is fishers' decisions whether to complain or not. Decision making can be implemented through different mechanisms.

5.2.1 Mechanisms for decision making

Decisions can be made through different mechanisms. The inputs and outputs are the same for every mechanism; only the way of determining the outcome is different. This section covers two different decision making mechanisms for fisherman voting: using artificial neural networks, and rule-based decisions.

Rule-based decisions

A way of studying the consequences of the simple decisions is to use a very simple decision-making process in order to produce predictable quantities of complaints and non-complaints. Fishermen can be configured to use a simple rule-based approach to complaining where they choose a straight-forward complaining strategy based on a probability. There are three different rules that can be applied by the system: they can either always complain, never complain, or complain with a probability that's proportional to the distance from the home cell to the threatened cell. This mechanism is not applied in any of the experiments, but included in the implemented simulation system.

Case-based reasoning

A case-based reasoning system could be applied to fisher complaint decisions. Fishers experience each situation where they are prompted for complaints as a case, and the decision whether to complain or not would be recorded as the case solution. No case-base reasoning system is implemented because using neural networks was considered to be a more straight-forward approach and comparing different decision making techniques was not a goal for the project.

Neural network

Applying a neural network for fishermen decisions has the advantage of making fewer assumptions about the nature of the fishers' decisions. The neural network is implemented with input nodes corresponding to the inputs discussed in section 4.3: the distance between the threatened cell and the home cell; and fishing conditions at both the home location and the threatened locations. For each agent, the network is activated once per aquaculture area in the plan, and the output is the complaint strength of each area; i.e. a measure of how much the agent wants to complain. The cells with the highest complaint index are complained about, if they are above a certain threshold. The artificial neural network applied to fishers in the simulation uses six hidden layer neurons.

The network applied in the system has a static structure which consists of three input neurons, six hidden-layer neurons and one output neurons. All the input neurons are connected to all the hidden-layer neurons. The hidden-layer neurons have recurrent connections to themselves as well as the output neuron. The hidden-layer neurons also receive connection from a bias neuron which applies a constant bias value. The weights on edges are evolved using artificial evolution.

5.2.2 Learning

Learning is applied both to the neural network and rule-based system through artificial evolution. When the learning mechanism is applied, it uses fitness values calculated by the priority satisfaction of each agent to sort the phenotypes of their decision mechanisms. A number n of elites are selected and their genotypes are preserved for the nest generation. Rank selection is applied to the remaining phenotypes m in order to find m new phenotypes that continue to the next generation. Their genotypes are then extracted and subjected to mutation and crossover, with a low probability for each genotype in the case of mutation and a low probability for each pair of genotypes in the case of crossover. The mutated genotypes and the preserved genotypes of the elites are then developed into $n+m$ new learning mechanisms that are reattached to the agents.

The artificial neural network phenotype is a description of all weights between neurons in the network. The neurons of each type are already decided and are therefore static attributes of the system. The genotype is a randomly generated string of bits that maps to a phenotype. When the bit string is divided into segments each segment represents a weight in the network. The phenotype for the rule-based mechanism is a selected rule from the three possible ones. The genotype is a randomly generated bit string that encodes one of these rules.

5.3 Agent Communication

Communication is a way of sharing information. In this model, agents communicate using messages. The messages go through a central repository, a directory, which keeps track of all the agents in the system, and their roles. Agents can ask the directory for a list of agents of any given role, for instance the fishermen can ask the directory for the government in order to complain about a cell.

5.3.1 Messages

FIPA (Wooldridge, 2002b) was studied for inspiration to the messaging protocol. However, the complexity was not necessary and a very simple protocol was devised instead. The different types of messages will be reviewed next.

Hearing and plan distribution

The first message issued each round is a notice to each agent that a coastal plan has been created, and the contents of the plan. The plan contains a list of areas that has been reserved for aquaculture, and agents can complain about these areas.

Vote for a targeted cell

The actual voting messages are the messages that carry the most private information, at least in the case of fishermen. Other fishermen will infer that if a fisherman complains about the establishment in a certain location, it is because he or she knows that the location has good fishing conditions.

5.4 Other Rules Affecting the Agents

Some details that are not implementation-specific, but not easily classified are documented in this section. The dynamics of map creation and aquaculture establishment are important and have consequences on the evolution of the map.

5.4.1 Map creation

In order to keep the simulation small and uncomplicated, the map is a small grid structure with randomly placed resources. This way it models an ocean area with varying levels of good fishing. A grid is defined by width and height, and width and height in meters for each cell. Another aspect of the map is the frequency of good fishing spots, which are the randomly distributed resources.

Fishing spots

Good fishing spots are spots where the fish spawns. Fishermen want to fish in these spots. Aquaculture agents want to establish aquaculture in these spots since they provide good conditions, but the fishermen are vehemently against it since disturbing a spawning ground affects the entire coast line population.

During periods of “normal activity”, the fishermen will go fishing at the best location they have knowledge of. The fish they harvest goes through a marketplace where it is converted to profits.

Fishermen

Fishermen are initially distributed on good fishing spots as long as there are fishing spots left, the rest are distributed in random locations. This setup attempts to model the fishers’ real situation, where they have held their own fishing locations for generations that the other fishers respect.

5.4.2 Aquaculture building

Building an aquaculture facility blocks fishing from the surrounding cells. This is according with standard regulations, that nobody can go too close to a facility. Another influence of aquaculture facilities being built is a radius of environmental damage caused by the facility. Within this radius, fish resources are destroyed at a rate proportional to the distance from the facility, so resources at the center will be completely destroyed if the damage proportion is 1.

5.5 Priorities of the Different Groups

The driving force behind both learning (through fitness calculation) and decision making is priorities of the agents implementing those mechanisms. Agents with different priorities may act differently to fulfill their wishes. Two problems of designing priorities are how to represent them realistically, and calculating how well an agent fulfills its priorities.

5.5.1 Representing priorities realistically

Some priorities influence each other, and some are completely dependent on each other. For example, profits from fishing is dependent on both the market rates for wild fish and quantity of fish farmed. Therefore fishermen only really care about their own profits, not about the market rate itself even if it is realistic to think that fishermen care about the market value of their fish. Thus, even if it is not represented explicitly, it is present in the model since they care about profits which is directly influenced by market rates. On the other hand, it can be reasonable to assume that fishermen genuinely care about the health of the fish, not only because it affects the quantity of fish they catch, but also because the health of fish populations is important to other industries and cultures. Therefore fishermen caring about the health of natural fish stocks is treated separately to their own profits. Similarly, the continued existence of fishing as a business is important as a separate entity from own profits because it represents both the value of a community, and the fact that fishermen care about their successors, namely their children attending school receiving an education in the fishing business.

5.5.2 Calculating priority satisfaction

The complete satisfaction of an agent is simply calculated by the weighed average of the satisfaction of each individual priority of that agent. The total priority satisfaction for an agent a, e_a thus becomes:

$$e_a = \sum_{p \in P} S_p(a) W_p(a)$$

where $S_\pi(\alpha)$ is the satisfaction of priority π for agent α and $W_\pi(\alpha)$ is the relative weight of priority π for agent α. The weights W are configured for each priority for each agent type and $\sum_W W(\alpha) = 1$.

Some of the satisfactions for priorities are straight-forward to calculate. An agent's own profits can be calculated by simply returning the capital of that agent. Community wealth is similarly measured by taking the average of all citizens' capital. The market controls the prices of wild fish and farmed fish. Satisfaction for the priority of having an existing fishing industry can be measured by the fraction of number of fishermen that still fish for a living, divided by the number of fishermen in total. Natural fish health is simply measured by the average quantity of fish in each cell, which is negatively impacted by aquaculture facilities.

5.6 Actions of the Different Groups

Different types of agents perform different action that collectively define their behavior. Actions can interact with the world or other agents, or consist purely of internal processing. This section identifies all the actions the different agents can do, that are relevant to the simulation.

5.6.1 Fishermen actions

Fishermen are the central agents of the simulation. The most important action they do which influences their well-being is fishing, during the fishing phase. The fishing consists of identifying and moving to a fishing place, and then fishing there. If one or more fishers decide to fish in the same place, they naturally have to share the resource.

Another important action the fishermen do is complain about a potential threat to their livelihood. Elements in the coastal plan which dictate that areas that the fishermen are concerned about should be reserved for aquaculture are threats to the livelihoods of fishermen. Complaints about the coastal plan can influence the plan, and are therefore important actions. Another related action is the choice to *not* complain. Fishers may want to stay silent in order to not reveal the location of their secret fishing spots.

Related to the fishers' complaints is the ability to observe what other fishermen vote. This action gathers information about the location of potentially good fishing spots. This is the way fishermen learn about new fishing spot, and the mechanism is an important motivation for fishers to not complain. Agents cannot make definite conclusions about the quality of a fishing spot based on complaints about it, so they have to rely on guesses. Different agents may guess differently. Picking a random number from a distribution with a defined mean and standard deviation is a sufficient strategy for modeling these guesses because it captures these differences, while keeping the guessing process simple and straightforward so that the simulation can have focus elsewhere.

5.6.2 Municipality

The municipality is central to the simulation because it is the entity that creates the coastal plan. This means one of its actions is planning, and another is reviewing of the plan in case of approved complaints. Another function of the municipality is to gather and distribute tax benefits from working agents to the community. In the model, this benefit is modeled as a monetary payout to each agent which is a community member. In other words, two more actions of the municipality are tax collection, and tax benefit distribution.

5.6.3 Government

The government has two responsibilities in the simulation: one is to act as a separate entity from the municipality to receive and treat the complaints, and the second is to distribute

Table 5.1: Actions of the different agents

Agent type	Actions
Fishermen	Fish for profit
	Complain about planned aquaculture
	Stay silent to hide information
	Observe other fishers' complaints
Municipality	Coastal planning
	Review the coastal plan
	Collect taxes from working agents
	Distribute tax benefits to community members
Government	Decide hearing outcome
	Distribute aquaculture licenses to municipality
Aquaculture owners	Establish facility in a planned location
	Work the facility for profit

licenses to the municipality for aquaculture organizations. From the first function there the government has to do the action of deciding what to do with complaints. From the second function, the government has to do the action of distributing licenses.

5.6.4 Aquaculture organizations

Aquaculture owners are obviously important since they control the facilities the simulation focuses on. In order to build a facility, the owner has to apply for a license from the municipality, for a certain location in the coastal plan. Therefore an action they perform is to pursue a location for aquaculture expansion. They can also work their facilities after they have been established, to gain profit, and they pay a tax on this profit. Aquaculture owners could also be considered to be a part of the hearing process, but the focus of this simulation is on fishers' decisions, so aquaculture agents are assumed to always approve new aquaculture areas.

5.6.5 Other agents

Civilians and tourists can work and complain about the establishment of aquaculture, but they are not used in the simulation, as discussed in section 4.1.

Chapter 6

Implementation

The software written for this project is entitled *FisherSimulation*. The source code for the simulation is written in Python, following standard object-oriented programming techniques. Before writing the software, other alternatives were explored.

CORMAS is a platform for creating agent-based simulations, especially with an ecological focus (Le Page et al., 2012). Therefore, it seemed to be a perfect fit for this project which involves intelligent agents (fishermen, government, municipality, aquaculture organizations) and ecological resources (fish). However, setting up and working with the CORMAS platform is cumbersome, especially without Smalltalk experience. During the prestudy for the project, significant time was spent trying to use and learn CORMAS. The model used by Janssen and Ostrom (2006) to simulate fishers self-organizing in Lofoten was obtained and attempted to be ran. Unfortunately, it did not run on any of the available versions of CORMAS. The authors of the CORMAS platform were contacted and were able to help modify the source code of the model in order to run it for 10 time steps, but they decided that the model was erroneous or unfinished. The creators of the platform regularly host workshops, which is the most common way of learning it. I was not able to attend any CORMAS workshop, and combined with the troubles experienced when trying to run the old model, writing custom software was concluded to be the most effective way of making a good simulation.

As any software, *FisherSimulation* can logically be divided into two more or less separate parts: The developer is mainly concerned with the implementation which includes project organization, programming paradigms and of course the code itself. The user, on the other hand, is faced with a user interface, graphical or otherwise, which they have to learn and use in order to interact with the program. This documentation attempts to cover both topics thoroughly in order to guide both users who want to benefit from the simulations offered by the program as-is, and future developers who wish to extend, loan or simply study the implementation.

FisherSimulation was created as part of this master thesis applying a multi-agent system to a real-world problem. Fishermen in Frøya, Sør-Trønderlag, Norway are faced with two choices when aquaculture companies are threatening their fishing spots. When the

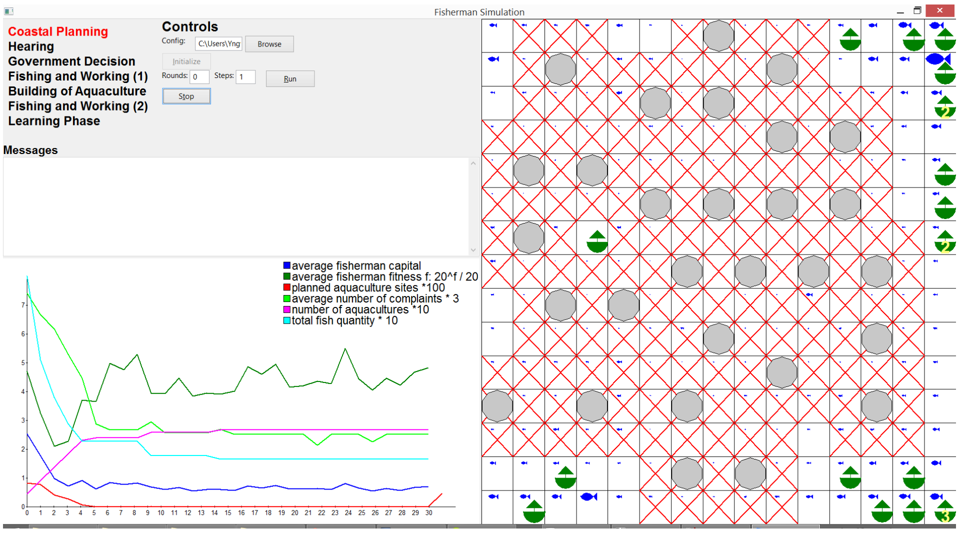

Figure 6.1: Screenshot of the graphical user interface of *FisherSimulation*.

municipality creates a coastal plan and releases it for hearing, fishermen can choose to complain about areas in the plan. This might influence a decision-making institution to rework the plan, accommodating the complaints. After the coastal plan has been finalized, the aquaculture companies may establish in areas allowed by the plan when licenses are distributed. If the fishermen don't complain, or the plan is not reworked, they risk losing precious fishing spots to aquaculture facilities.

6.1 Graphical User Interface

Figure 6.1 shows a screenshot of the user interface of *FisherSimulation*.

6.1.1 Phase Information Panel

The time between each new coastal plan is divided into phases, which are displayed in the top-left corner of fig. 6.1. The current phase that is being computed is highlighted with red. After the third phase, *Government Decision*, the simulation may either jump back to *Coastal Planning* or continue to the first fishing phase, depending on what the government decides. If any complaints from the *Hearing* phase are approved, the simulation will jump back to the planning stage.

6.1.2 Controls panel

To the right of the phase overview, there is a set of controls for manipulating the initial conditions such as map size and agent priorities, and controlling execution of the program. A custom configuration file can be loaded by the explorer. The program expects a *JSON* file with a number of required objects and parameters, detailed in section 6.2. Below the configuration file box, the *[I]nitialize* button creates the simulation process which can be processed using the *[R]un* button. The default amount of steps to be processed is only one at a time, but this can be controlled using the *Rounds* and *Steps* inputs to the left of the *[R]un* button. If a number of whole rounds is specified, this counts as a whole round from a first coastal planning, through any number of government decision-replanning loops, through learning and back again to a new coastal planning. After all complete rounds are counted, the steps are counted as well. The *S[t]op* button discards the current simulation and enables a new one to be initialized. It will also abort any currently running multi-step process.

6.1.3 Messages

Below of both the phase overview and the controls panel, there is a text box entitled "messages". Agents in the simulation communicate using messages, which are sent through a central repository. All messages recorded by this repository are displayed in the messages box. A message can be either single-target or a broadcast, and in the case of the only latter a compromised list of recipients is shown. All messages are displayed with sender, recipient(s), a timestamp which is equal to the total number of messages sent when the message was created, and a summary of the contents. When a message is received by an agent, the agent invokes a reaction method in the agent that is specified by the message type. The most common messages are:

Distribution of a plan After the coastal plan is finalized, the municipality sends it to all other agents.

Votes for a hearing round After an agent decides their complaints about a coastal plan, it sends the complaints to the government. The complaints are sent as a list of objects that refer to the cell it regards.

Broadcast of an agent's votes Because votes are public, the government broadcast the information every time it receives votes from an agent. The broadcast is distributed to all fishers, aquaculture organizations and other stakeholders.

6.1.4 Plots

Below the message box, there is a graphical box containing plots that describe the state of the system in each round. The statistics are shown as line plots, some of which are normalized statically for easier viewing. The default plots are:

Planned number of aquaculture sites (Red) which is equal to the number of areas reserved for aquaculture in the coastal plan. This number might change during a round, because the *Coastal Planning* phase might be revisited up to three times. The number of planned aquaculture sites can become very large since the only initial restriction is the number of areas in the map, so this number is normalized by division by 100. The number of planned aquaculture sites P (and the normalized P_n) is equal to:

$$P = \sum_{s \in S} \omega(s), \quad \text{where } \omega(s) = \begin{cases} 1 & \text{if } s \text{ is reserved for aquaculture} \\ 0 & \text{otherwise} \end{cases}$$

$$P_n = \frac{P}{100}.$$

where S is the set of all slots in the coastal plan.

Average number of complaints (Green) which is equal to the amount of complaints issued by every single fisherman, averaged over the population of fishermen. The number of complaints by a fisherman may increase three times in a single round, because the *Hearing* phase may be revisited, in the case when the government decides to review the plan. This number is normalized by division by 3 in order to keep it relatively similar to the other graphs in magnitude. The average number of complaints C (and the normalized C_n) is equal to:

$$C = \frac{\sum_{f \in F} c_f}{|F|}$$

$$C_n = \frac{C}{3}$$

where F is all fishermen, c_α is the number of complaints by fisherman α, and $|F|$ is the number of fishermen.

Number of aquacultures (Purple) which is equal to the number of aquacultures on the map. This number increases by a constant number equal to the amount of licenses distributed each round, as long as enough areas are reserved for aquaculture in the coastal plan. In other words, if enough complaints are approved, this number won't increase. It is normalized by division by 10 because there are many slots for aquaculture, compared to the other graphs' values. The number of aquaculture facilities is equal to A, and the normalized number of aquacultures A_n is equal to:

$$A_n = \frac{A}{10}$$

Average fisherman capital (Blue) which is equal to the amount of money, capital, each fisherman owns, averaged over all of them. Fishermen mainly receive revenue from fishing activities, but since they are community members, they also receive tax benefits from aquaculture companies. Typically fishermen's capital slightly increase in the beginning of the simulation, when the number of aquaculture facilities is increasing. Later it declines when the fishermen are pushed away from their preferred fishing spots, either by aquaculture facilities establishing, or by blocked areas as a consequence of those facilities. Capital is reset (spent) each round. The average fishermen capital K (and the normalized K_n) is equal to:

$$K = \frac{\sum_{f \in F} k_f}{|F|}$$

$$K_n = K$$

where F is all fishermen, k_α is the capital of fisherman α, and $|F|$ is the number of fishermen.

Average fisherman fitness (Dark green) which is equal to the evolutionary fitness of each fisherman, averaged over all fishermen. Fitness is a term from evolutionary algorithms, the learning mechanism used in the system. For fishermen, their fitness is calculated as their combined satisfaction of various priorities. The most important priority for fishermen is their own profits, which is their capital. Another priority that is important for the fitness is community wealth, which is influenced by aquaculture which gives back to the community. This balance often causes fitness to stay relatively stable, even if fishing spots are disappearing, since it means that the community will be richer as a consequence of more aquaculture industry. The average fisherman fitness is normalized by the formula $\frac{20^f}{20}$ where f is the average fitness. This is done because the fitness is subject to very small changes, but the power expression enhances these changes so that they are more visible. In order to fit with the other plots, it is divided by 20 again. The average fisherman fitness E (and the normalized E_n) is equal to:

$$E = \frac{\sum_{f \in F} e_f}{|F|}$$

$$E_n = \frac{20^E}{20}$$

where F is all fishermen, e_α is the fitness of fisherman α, and $|F|$ is the number of fishermen. The fitness measure e_α is equal to the priority satisfaction of that agent, which is calculated as explained in section 5.5.2.

Total fish quantity (Cyan) which is equal to the sum of quantity of fish in each cell in the world. The quantity of fish in a cell is measured as a number between 0 and 1, and cells with the "spawning" attribute naturally have more fish than others. Usually this number will decrease rapidly as more aquaculture facilities are built, which diminish fish resources in a configurable radius, by a configurable amount, around them. It is normalized by division by 10 since in order to be similar to the other plots. The total fish quantity Q (and the normalized Q_n) is equal to:

$$Q = \sum_{c \in M} q_c$$

$$Q_n = \frac{Q}{10}$$

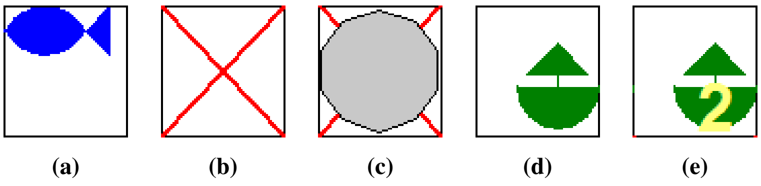

Figure 6.2: Symbols in the map. Cells in the grid have symbols indicating that they are: **a**) a very good fishing spot (a smaller fish means that the spot is worse), **b**) blocked from fishing activities and possibly further aquaculture expansion, **c**) occupied by an aquaculture facility, **d**) the current “home” location of a single fisherman, or **e**) the home location of several (in this case 2) fishermen.

where M is the set of all cells in the map and q_α is the quantity of fish in cell α.

6.1.5 Map

On the right side of the user interface, the map displays fishermen, aquaculture sites, good fishing spots and blocked cells. It is laid out as a grid, with a configurable number of rows and columns. Figure 6.2 explains the symbols on the map. Each cell may contain zero, one, or several symbols. For example, a good fishing spot that gets aquaculture built on top of it will still be a good fishing spot, and displayed as such. Fishermen may move around on the board during the *Fishing and Working* phases, in order to relocate to better fishing spots. During the *Hearing*, fishermen who complain will become darker, and gradually more red in order to indicate the complaint intensity and distribution in the population.

6.2 Configuration File Specification

Configuration of parameters, initial conditions and other settings is done through a single *JSON* file. There are 9 top-level objects, one for each of the six agent types, and three more for general configuration. The configuration objects for the agent types all specify the priorities for that agent, but only the fishermen’s priorities are used in the simulation. A complete description of all the expected items is given in appendix A.

6.3 Modules and classes

The project code is created in an object-oriented fashion, and is divided into three packages: graphical user interface, configuration and the logic. Appendix B gives a complete documentation of all the classes and modules used in the logic package, which performs the simulation.

6.4 Program Dependencies

The simulation program and the graphical user interface were developed in Python 2.7. Libraries used are listed below:

NumPy (NumPy, 2014) Numeric calculation library

wxPython (wxPython, 2014) Graphical user interface library

6.5 Extending the Program

In order to make the program easy to extend, a goal during development has been to make configuration generic and parts replaceable. In order to easily understand what are needed of the different replaceable parts, well-documented interfaces have been created. Interfaces and extending is documented in appendix C.

Chapter 7

Workshop in Frøya

After initial development of the simulation, a workshop was held in Frøya municipality with seven representatives of the fisherman community there. The software simulation was shown and compared to the fishermen's real situation, where they face the threat of aquaculture pressing them out of their traditional fishing spots, as well as other consequences such as environmental damage.

The simulation is built largely on research by Tiller et al. (2014), and the simulation was presented as the second part of an updating session with the same community. The goal for the primary stakeholder workshop was to investigate how the fishermen's attitude towards aquaculture had changed over the previous two years. The Frøya fishermen are considered to be in a very special situation since they have previously been very open to aquaculture, while other fisherman communities are vehemently against it.

7.1 Presenting the Simulation

The simulation was described and presented to the fishermen as a tool to understand their decisions with regards to complaints about aquaculture expansion. It was important to emphasize that the simulation was built on several assumptions and simplifications about the fishermen's situation and possibilities of influence, the complaining process, as well as the aquaculture companies.

The strategy was to try to make the fishermen explain what they saw in the simulation, rather than explain the simulation to them. This required a basic introduction of the symbols and graphs in the interface, but little more than that.

7.2 Feedback

The fishers had many opinions on how well the simulation reflected their situation. Much of the feedback implied that revisions had to be made to the program, but some was positive. Below is a recollection of the main points brought up by the fishers.

Complaint system is not realistic The aquaculture companies take areas no matter if the fishermen complain or not. This was a big point. Their experience showed that complaining was useless, even if they paid a large effort into the complaint; showed thorough proof.

Maybe quantity over quality would give results? Also, in our model, complaining has no cost. Since the fishermen had put a lot of effort into their complaints, that means they also paid some cost to make them. Should our model include a cost?

Respect The fishermen respect each others' areas and will not go and fish in each others' spots, unless they are being forced away from their own.

This aligns well with the model where fisherman agents only move when they are forced to, since they assume all other spots than the one they start with are worse.

Spots are secret Several of the fishermen had secret spots that they did not want revealed. When asked if this had an impact on complaining, the answer was unclear.

In the model, agents capitalize on other agents revealing their good fishing spots. However, the influence of this behavior on decisions is only indirect, through payoff-based learning.

Aquaculture is good for the community All the fishermen were clear on the point that if it were not for the existence aquaculture in Frøya, it would be a much smaller place with a lower standard of living.

Fisherman agents in the simulation receive direct benefits from the existence of aquaculture, since they pay taxes that are distributed to community members (including fishermen). Fishermen also have community wealth as an important priority, which is positively influenced by this taxation as well.

Complaining doesn't work Past experience showed the fishermen that complaining doesn't work.

If agents in the model experience no increased fitness from complaining behavior, they will stop complaining.

Aquaculture is more profitable Some fishermen expressed frustration with experience with the regional government ridiculing the fishing industry and the notion of leaving room for it in the coastal plan, because aquaculture is so much more profitable for all parties.

In the simulation, aquaculture is more profitable than fishing, but there is no notion of the governing parties prioritizing aquaculture more because of it. Any such prioritizing present is built into the protocol, where all slots are initially planned for aquaculture in each planning round.

The regional government is not "on their team" Even if the municipality rules that some areas are reserved from aquaculture, the regional government has in the past overruled those decisions and allowed aquaculture companies to establish there, to the protests of the fishermen.

In the simulation, the decision-making government is rightly separate from the municipality, but the plan is final and cannot be overruled to allow for rich aquaculture companies to establish in reserved zones.

Pushed away from the good spots The fishermen recognized the scenario showed in the simulation, where the aquaculture companies eventually establish in all the good spots, and the fishermen are pushed out to the corners of the map and to spots that aren't as efficient.

Fishermen must move Another behavior by the simulated agents that reflected reality in the eyes of the fishermen, was the fact that aquaculture facilities can stay still and continue in their spots once established, but the fishermen have to move around and constantly search for new spots.

7.3 Revisions to the Simulation

The workshop resulted in some revisions to more realistically reflect the fishers' situations. Firstly, an important factor for the fishers that originally was not a part of the simulation is environmental damage caused by aquaculture. Introducing a configurable damage property solves this problem. Secondly, the simulation needs to reflect that the government does not always favor fishers' complaints. To reflect this behavior, a new decision mechanism is added for governments which uses a probability to decide whether complaints should be approved or not.

7.3.1 Environmental damage

Aquaculture cause environmental damage proportional to a constant ϵ in a configurable radius around ρ the area. It inflicts linearly less damage based on the distance, the maximum amount ϵ at the center and 0 at the edge of the radius.

7.3.2 Government decision

The government decision can favor aquaculture or fishermen, by a configuration. To support this kind of tweaking, a new government decision mechanism was implemented, `ApproveProbability` in the decision making module (DM). It receives a configurable probability which it uses for each complaint to probabilistically decide whether that complaint is approved or not. This way, the government decision can easily be tweaked to be in favor of the fishermen (high probability), or in favor of aquaculture companies (low probability).

Chapter 8

Experimental Results

The plots and map symbols used to present the results in this chapter are explained in sections 6.1.4 and 6.1.5.

Experiments are defined with an overall topic and one or more research questions. Table 8.1 shows an overview of the experiments done. Experiments are ran with different parameters in order to analyze the influence of those parameters. Some configuration is common to all the experiments, and these parameters are shown in table 8.2.

8.1 Experiment 1: Changing complaint approval rate

After the workshop, the default decision mechanism for the government's vote approval was changed to a configurable probabilistic one. This way different levels of aquaculture-friendliness of government can be explored, and the consequences for the fishers of these settings can be observed.

Observing how changing complaint approval rates impact complaining behavior can help clarify why complaining rates are so low in the real-life scenario in Frøya because feedback from the fishers dictates that the complaint approval rate at Frøya is very low. If there is a relationship between complaint approval rate and non-complaining behavior,

Table 8.1: Overview of experiments, showing a short description and related research questions for each experiment.

Experiment	**Description**	**Research Question(s)**
1	Changing complaint approval rate	1
2	Identifying outcomes	1 and 2
3	Evolution of complaining behavior	1 and 2
4	Changing amounts environmental damage	1

Table 8.2: Common parameters for the experiments, showing the parameter name and the value of the parameter.

Parameter	Value
General settings	
Number of maximum complaints	10 per hearing
Maximum number of hearing rounds	3
Aquaculture blocking radius	25 m
Map size	15 × 15 cells
Cell size	25 × 25 meters
Frequency of good fishing spots	0.1
Number of fishermen	20
Complaint decision mechanism	Artificial neural network
Fisher learning mechanism	Evolution
Aquaculture licenses each round	5 licenses
Fishermen priority weight distribution	
Own profits	0.5
Community wealth	0.1
Fishing industry health	0.15
Natural fish health	0.2
Aquaculture industry health	0.05
Complaint decision evolution parameters	
Elitism	3 phenotypes
Selection mechanism	Rank selection
Crossover rate	0.005
Mutation rate	0.005
Genome mutation rate	0.0005

this can be of aid when answering research question 1.

Table 8.3 shows 10 runs for each of the three approval rates $r \in \{0.2, 0.5, 0.8\}$.

In general, there are two kinds of outcomes: either the average number of complaints stabilizes at a non-zero level and there is a stable area that is continuously protected from aquaculture, or agents learn to not complain and the map is filled with aquaculture. We call this outcome a "collapse". Summarized, the two outcomes are:

Stabilize The number of non-blocked cells is non-zero, and the level of complaints has stabilized at a non-zero average rate.

Collapse The whole map has become occupied by aquaculture facilities and blocked cells in a radius around them. Complaining behavior has disappeared from the system.

The two outcomes are equally likely at all three approval rates, with $\frac{5}{10}$ of the cases for 0.5 and 0.2, and $\frac{4}{10}$ for 0.8 approval rate. Furthermore, the approval rate influences how many open cells are left in a stable situation, and how many aquaculture facilities are built. Table 8.4 shows the relationship between approval rate and average number of open cells and aquaculture facilities, for each of the two outcomes. With the lowest approval rate of 0.2, there are on average 24.2 open cells left and 35.4 aquaculture facilities in the stabilized outcome. With the medium approval rate of 0.5 there are on average 53.2 open cells and 28.6 aquaculture facilities when the simulation has stabilized. When the approval rate is higher at 0.8, there are on average 70.667 open cells and 25.333 aquaculture facilities in the stable outcome.

8.2 Experiment 2: Types of outcomes

The two identified outcomes can both occur at any approval rate, but the stable outcome is different for different approval rates. The number of aquaculture facilities in a collapsed scenario does not have any relation with the approval rate, and is merely a function of the size of the world.

Analyzing when the two different kinds of outcomes happen is fundamental to understanding why and how they can occur in corresponding real-life situations. A collapsed outcome can be tied with the fishers' tendency in Frøya to not complain, as questioned by research question 1. A stabilized outcome, on the other hand, may be indicative of either an honest level of complaints, or potentially over-complaining, as questioned by research question 2. Studying the different outcomes more closely can therefore aid in answering both these questions.

Both fig. 8.1 and fig. 8.2 are captured from runs with a complaint approval rate of 0.5.

Taking a look at the process, fig. 8.1 shows a typical simulation run where the situation collapses into complete aquaculture dominance, described by a number of plots. We see that the average number of fisherman complaints (Green) rapidly decreases until it reaches 0 and never raises again. The number of aquaculture facilities (Purple) steadily increases until there are no available cells left. At the same time, average fisherman capital (Blue) decreases steadily until it reaches 0. Average fisherman fitness (Dark green) decreases at first quite rapidly, before it slowly rises and stabilizes when the maximum number of aquaculture facilities is reached. The total fish quantity (Cyan) rapidly decreases at the

Table 8.3: Results of varying approval rate on the final state of the world, showing for each run the configured complaint approval rate along with the simulation outcome through number of open cells and number of aquaculture facilities. Open cells are cells that are not within the blocking radius of an aquaculture facilities, and fishers can fish in these cells. The total number of cells in the world is $15 \times 15 = 225$.

	Approval Rate	**Open cells**	**Aquacultures**
1	0.5	0	43
2		55	27
3		0	45
4		55	30
5		53	27
6		43	30
7		60	29
8		0	44
9		0	48
10		0	49
11	0.2	0	49
12		19	40
13		24	33
14		24	37
15		0	44
16		29	32
17		0	44
18		25	35
19		0	42
20		0	45
21	0.8	69	25
22		60	26
23		0	43
24		78	26
25		66	22
26		0	46
27		0	44
28		0	47
29		85	23
30		66	30

Table 8.4: Average number of open cells and aquaculture for stable and collapsing outcomes, given the three different approval rates. The configured approval rate is shown in relation with the outcome data showing the average final states of the worlds, separated by outcome. Stable outcomes are compared by the number of open cells left and the number of established aquaculture facilities. The number of open cells in a collapsed outcome is always equal to 0, so they are not shown.

	Stable		**Collapse**
Approval Rate	**Open cells**	**Aquacultures**	**Aquacultures**
0.2	24.2	35.4	44.8
0.5	53.2	28.6	45.8
0.8	70.667	25.333	45

start, and then stabilizes at a low level when the maximum number of aquaculture facilities is reached.

On the other hand, fig. 8.2 shows a similar plot of a simulation run, but the scenario stabilizes with a relatively stable amount of complaints. The number of aquaculture facilities increases until it reaches a stable point that's below the maximum amount. The average fisherman capital decreases steadily in the beginning, and stabilizes at a non-zero level once the number of aquaculture facilities does. The average fisher fitness stabilizes at a slightly higher point than in the previous scenario, but instead of being completely constant it has continuous fluctuations. The total fish quantity rapidly decreases, but stabilizes when the number of aquaculture facilities does.

In a collapsed world, like depicted in all the cells are eventually either occupied by aquaculture facilities or blocked, and fishermen cannot fish. The amount of fish on the map have also diminished to almost nothing. On the other hand, in a stabilized world there are some cells that fishermen protect. These cells also have some fish left in them. Some fishermen have to share spots. The fishermen have not found the best spots in the map, and they are not distributed optimally to fish from the spots they know. Figure 8.3 shows final map states of the two different scenarios.

When a world has stabilized, no more aquaculture is built. However, the fishers do not settle completely for some cells. Figure 8.4 shows that fishers still move in a stabilized environment, even if the aquaculture sites do not change, and the fish quantity in the cells does not change.

In order to investigate the direct relationship between kind of outcome and average fitness, 10 runs where performed and recorded with regards to average fitness at the end state and the outcome type exhibited. Table 8.5 shows the results of these runs. The average fitness of fishers is higher in the stabilized runs with an average of 1.8574, compared to the collapsed runs which average at a fitness of 1.4440.

8.3 Experiment 3: Evolution of complaining behavior

The overall complaining behavior of fishermen evolves over time, sometimes resulting in collapse when nobody complains and complaining behavior disappears from the behavior

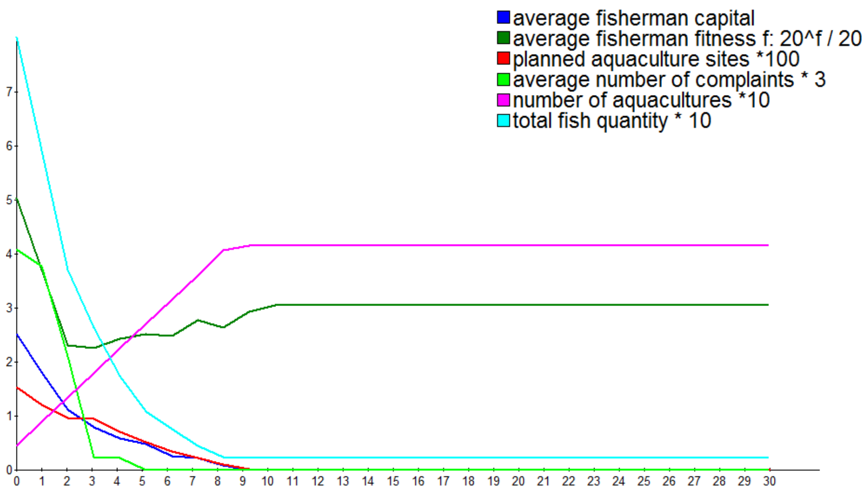

Figure 8.1: Plots showing a typical simulation progress resulting in a collapsed state. The horizontal axis represents the round and the vertical axis represents values for the different plots. The plots are normalized by scaling factors displayed in the label section. A collapsed state is characterized by complaint behavior quickly disappearing from the population, as shown by the light green plot. At the same time the number of aquaculture facilities, shown in pink, rises to its maximum value. The fisher fitness stabilizes once the number of aquacultures does.

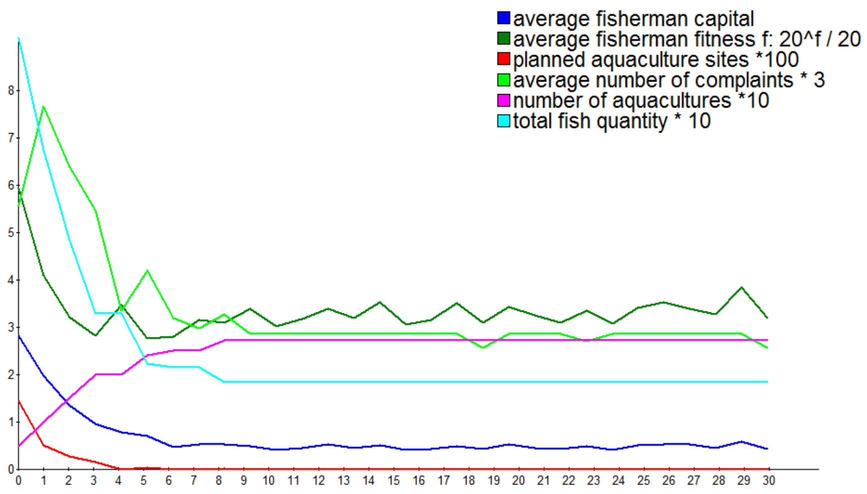

Figure 8.2: Plots showing a typical simulation progress resulting in a stabilized state. The horizontal axis represents the round and the vertical axis represents values for the different plots. The plots are normalized by scaling factors displayed in the label section. A stabilized state is characterized by complaining stabilizing at a non-zero rate, as shown by the light green plot which stabilizes at around 9 complaints per agent. The number of aquaculture facilities subsequently stabilizes below its theoretical maximum value, and the average fisher fitness fluctuates while staying somewhat constant.

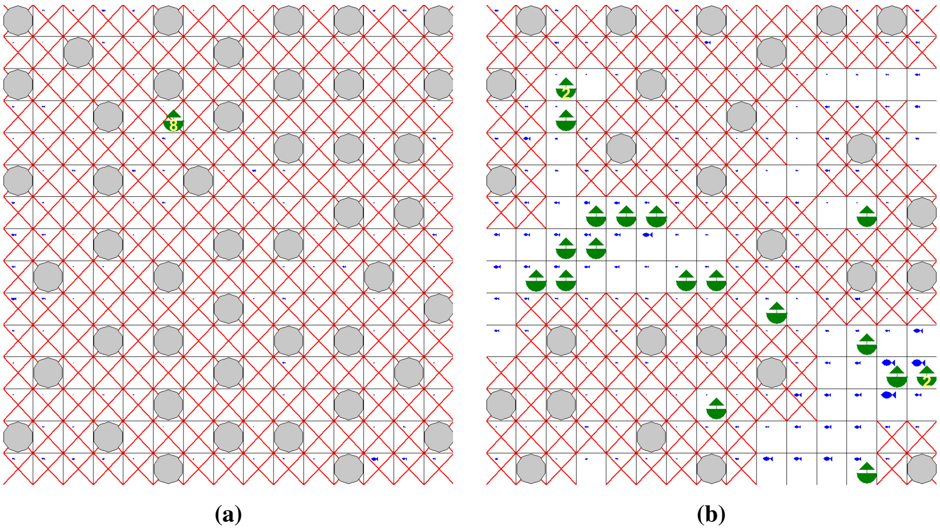

Figure 8.3: The world map showing **a**) the world map after a collapse; and **b**) the world map in a stabilized state. The map symbols are explained in section 6.1.5. In the collapsed state all cells are blocked because of aquaculture expansion, and fishers cannot harvest. In the stabilized outcome there are open cells left which contains some fish. Fishers are distributed relatively evenly among these resources.

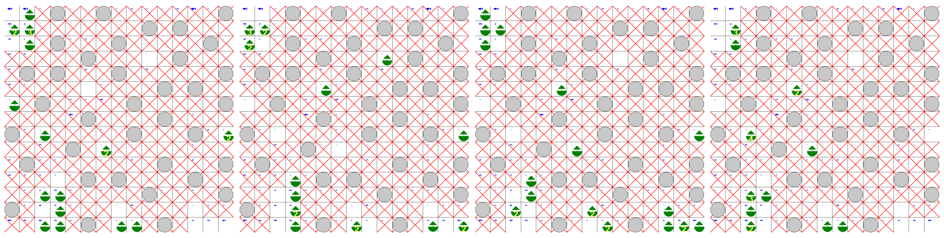

Figure 8.4: Snapshots of four consecutive rounds (27, 28, 29 and 30) of a stabilized world, showing that fishermen still move around even when no more aquaculture facilities are built. The map symbols are explained in section 6.1.5. All the cells are visited by fishers during these four rounds.

Table 8.5: Relationship between outcome and fitness, averaged over 10 runs with 6 of them resulting in collapse and 4 of them resulting in stabilization. Each run is displayed with the outcome characterized by number of open cells and number of aquaculture establishment, as well as the final average fitness of fisher agents. Runs where the number of open cells ends up being equal to 0 are tagged as collapsed, while the others are tagged as stabilized. All 10 runs are done with the same configuration settings.

Run	**Outcome**		**Fitness**
	Open cells	**Aquacultures**	
1	0	46	1.4523
2	0	48	1.4666
3	52	34	1.9880
4	0	46	1.4366
5	0	45	1.4287
6	60	31	1.8758
7	36	31	1.7306
8	60	23	1.8353
9	0	47	1.4441
10	0	46	1.4361
Average	**Stabilized**		1.8574
	Collapsed		1.4440

pool, other times it stabilizes at some level where it keeps the aquaculture expansion in check.

Studying how the level of complaints change over time, and how these changes impact the final result can provide further understanding of the outcome states. Since a collapsed outcome is relevant to research question 1 and the stabilized outcome is relevant to research question 2, analyzing the evolution and change of complaints may be helpful in answering both questions.

Figure 8.5 shows plots the average amount of complaints for 30 rounds of 20 different runs with the same configuration. Exactly half of the plots are non-zero at round 30, which means that the outcome is stable, and half are zero which means that the outcome has collapsed. The plot highlighted in green represents the stabilized outcome with the lowest amount of complaints. The plot highlighted in blue represents the second lowest amount of complaints, which is also significantly different from the other stabilized plots. Both the plots highlighted in green and red show different behaviors from the rest, since they both quickly fall at the beginning, but manage to recover into a stabilized state after almost reaching zero complaints on average. The red plot shows a development where the amount of complaints rapidly increases again and stabilizes at a level that's similar to the other stabilized plots, where the green one stabilizes at a very low point.

Figure 8.6 shows how complaints distribute over the different hearings in each round, over three runs. For all the runs the votes are distributed relatively evenly in the beginning, before complaints in the third hearing disappear after a few rounds.

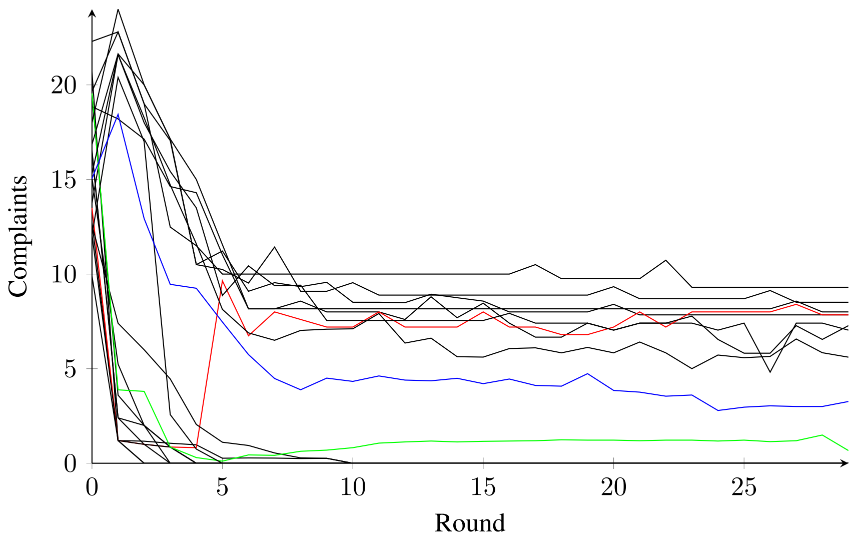

Figure 8.5: Average number of complaints for each round of 20 simulations using the same configuration. The three highlighted plots are of special interest because they have attributes that distinguish them from the rest. Non-highlighted plots show regular collapsed and stabilized outcomes. Out of the non-highlighted plots, there are 8 stabilized runs and 9 collapsed ones. The three highlighted runs are all classified as stabilized. The red plot shows the only run where the average number of complaints rapidly falls in the beginning, but recovers and stabilizes around the same level as the normal stabilized runs. The green plot also shows a run that falls rapidly and recovers, but the recovery is at a much lower level. The blue plot shows the lowest stability level with a normal profile.

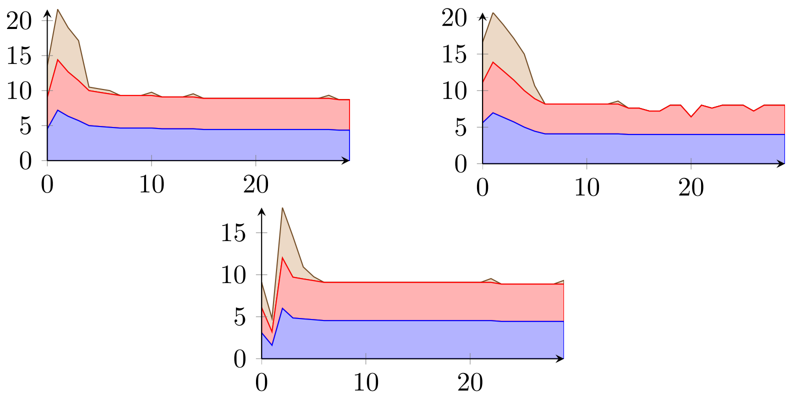

Figure 8.6: Plots of number of complaints for each round for three different stable simulation runs, showing number of complaints for each of the three hearings in the same round. The blue area shows the first hearing, the red area shows the second hearing, and the brown area shows the third hearing. The horizontal axis shows the round, and the vertical axis shows average number of complaints for the fishermen.

Table 8.6: Effects of changing the environmental damage variables on the simulation's final states, as an average of 20 runs for each configuration. Each 20-run average is defined by its configuration settings which consist of the radius of environmental damage caused by aquaculture facilities, and the amount of damage done signified by a proportion. The outcomes are described by a collapse rate, which is how often the configuration results in a collapsed outcome. Outcomes are divided into stable and collapsed outcomes, where the stable ones are compared by number of open cells and number of aquaculture facilities. The collapsed outcomes are only compared by number of aquacultures since there are no open cells left when an outcome is collapsed.

Configuration			**Stable**		**Collapse**
Damage radius (m)	**Proportion**	**Collapse rate**	**Open cells**	**Aqua-cultures**	**Aquacultures**
50	1	0.5	44	32.4	46.2
100	1	0.35	47.308	29.231	46.857
50	0.5	0.45	51.364	29.818	45.889

8.4 Experiment 4: Changing environmental damage

After the workshop, an environmental damage factor was introduced to the simulation.

Environmental damage was during the workshop mentioned as one of the main reasons why fishers are skeptical, and even opposed to, aquaculture expansion. Even so, the level of complaints is low. Seeing and analyzing how environmental damage impacts complaint levels in the simulation may be helpful in understanding the real-life implications. Since research question 1 asks why fishers don't complain, even when environmental damage is a real factor, the impact of environmental damage levels on simulated agent behavior may reflect the fishers' decisions.

Table 8.6 shows collapse rate, average numbers of open cells and aquaculture facilities in stable outcomes, and average number of aquaculture facilities in collapsed outcomes, for each of three configuration of different environmental damage. In all the configurations, the cell size is 25 x 25 meters. The first configuration uses a damage radius of 50 m, and a damage proportion of 1. That means that at the center of the aquaculture facility, the resource will be completely destroyed, and each cell at a distance from it will receive damage proportional to the distance. Cell at exactly 50 meters away will not receive any damage. The second configuration uses a damage radius of 100 m which means that cells twice as far away from the area will be damaged, and cells at 50 meters will be more damaged. The third configuration uses a damage radius of 50 m, but a damage proportion of 0.5, which means that cells at the center of the facility will be halfway destroyed, and cells at a distance will be damaged less than with a proportion of 1.

The scenario with the highest amount of environmental damage, with a 100-meter radius and a proportion of 1, yields the lowest collapse rate with 0.5, but the scenario with the lowest environmental damage, with a 50-meter radius and a proportion of 0.5 yields the highest amount of open cells in a stable world with an average of 51.364 cells. The medium-damage scenario, with a 50-meter radius and a proportion of 1, gives the lowest amount of open cells in a stable world (44 on average), and the highest amount of aquaculture facilities (32.4 on average compared to 29.231 and 29.818).

Chapter 9

Discussion

This thesis was based on two research questions that both regard the complaining behavior of fishers when facing aquaculture expansion. Fishers are observed to not complain, which contradicts intuitive predictions. It is also predictable that if fishers decide to complain in the first place, they will complain as much as they can, and even complain about locations that are not important to them. This behavior is referred to as “falsely” complaining. This discussion attempts to explain and answer these issues, one by one.

9.1 Why Fishermen Don’t Complain

One of the main goals of doing the simulation was to study how complaining takes place, and not least when it does *not* take place. The simulation can produce two opposite outcomes: either the complaint rate stabilizes and fishers manage to preserve an area from aquaculture expansion; or the complaint rate plummets, reaches zero, and the world is flooded by aquaculture until it is full.

In experiment 2, the different outcomes were analyzed. The collapsed outcome corresponds with the situation in Frøya, where fishers do not complain, while the stabilized outcome only occurs in the simulation. Collapsed outcomes are a result of fishers no longer complaining, which can be seen by looking at the plots for a collapsed run. Figure 8.1 shows that the level of complaints falls rapidly, which causes continued growth of the number of aquaculture facilities. On the other hand, fig. 8.2 shows by example that if the level of complaints stays above a certain threshold, the number of aquaculture facilities may stabilize accordingly.

We can also see from table 8.5 that the fishermen fitness is slightly higher in the stabilized outcome than the collapsed one, which means that the stabilized outcome is preferable to the collapsed one. Considering a population of infinite size, evolutionary game theoretic replicator dynamics predict that the agents learn the strategy with the highest payoff. This means that effects other than game theoretic learning caused the population of fishers to shift to non-complaining behavior. If the fitness of the complaining population is higher than the non-complaining one, the expected result of an evolutionary game theoretic process would be stabilization; i.e. a non-zero level of complaints.

One reason why fishers learn to not complain and end up with collapsed outcome may be stochastic effects in the simulation. There are several random or pseudo-random events that may disturb the outcome. Firstly the map is created by random, with good fishing spot and subsequently initial fisher location generated randomly. However, the layout is always similar with a relatively even distribution. The decision mechanisms in the population are also distributed by random in the beginning, and the neural network genotypes are created randomly, so one could think that the initial population may be created without any complainers. However, looking at fig. 8.1 one can see that there is a good amount of complaining in the first round, so there are certainly complaining agents in the initial configuration of some collapsed runs.

Another random element in the simulation is the distribution of aquaculture facilities within the aquaculture-reserved areas. An aquaculture organization chooses a random cell from the plan. However, when comparing the maps in fig. 8.3, there are no obvious dissimilarities in the aquaculture layout that separate the two outcomes.

A more subtle difference between collapsed and stable runs may be the learning process of the agents. The learning mechanism is based on social evolution, where agents observe others' payoff and copy the best-performing strategies. If no relationship between complaining and profit is observed, it is understandable that the population may end up not complaining.

9.1.1 Importance of the learning process

The fishers are learning agents, and they implement a form of social learning where they copy the decision making mechanism of their best-performing peers. A clear assumption of the model is therefore that fishers' complaint rates will have an impact on their fitness. This assumption is justified by the logic that since complaints have an impact on aquaculture expansion, and aquaculture expansion have an impact on fishers' income since they may be pushed away from their preferred, and better-yielding fishing spots. Since fishers' fitness is dependent on their capital, fishermen who don't complain will suffer. However, individual fishermen may be freeloaders of this effect if their peers complain about expansion in areas they care about, the non-complaining fishermen may also receive the benefits of reserved zones. Therefore, other agents can perceive that agents with non-complaining behavior do as well as the complainers do, and overall complaining may falter.

When complaining behavior completely disappears from the population, it never reappears again in any of the experiments. Figure 8.5 shows that for 20 rounds this holds true; not once does a plot that ends up with a non-zero value touch the bottom of the graph. In theory, mutation should be able to make agents recover their complaining ratios even if there is no complaining agents to copy, however to ensure this requires infinite time. Since aquaculture facilities cannot disappear and the map is of finite size, the fishers are on a timer to learn that complaining is beneficial. If by change agents don't observe that complaining gives a benefit, even if on average at an infinite time scale it would, the situation may collapse. This phenomenon reflects the situation in Frøya, since fishers there have tried to complain, but received no benefits. This effect may be tied more to the government's decision to approve or reject complaints, rather than the fishers' decisions to complain or not.

9.1.2 Influence of complaint approval rate on complaining behavior

Since complaints have never worked for the fishers at Frøya, it is important to explore how complaint approval rate influences the complaint rate. Experiment 1 was conducted with the goal of exploring the consequences of varying approval rates. The two outcomes of collapse and stabilization are shown to be equally likely with every tested approval rate, but the size of the protected area in the stabilized outcome becomes larger with increasing approval rate, as shown by table 8.4.

The fact that the rate of collapsed outcomes stays the same with varying approval rates suggests that the approval rate is not decisive of complaining behavior disappearing from the fisher population. A low approval rate may therefore not be the only reason why fishers are not complaining at Frøya. Increasing the approval rate of the government may therefore also not be sufficient to push fishermen to complain more.

The complaint approval rate does clearly influence how much area the fishers manage to protect in a stabilized outcome. There is a linear relationship between the two figures. The total size of protected areas is shown by the "open cells" statistic in table 8.4. Low complaint rate may be explained by this effect if fishers give up on complaining when they see that they cannot protect a large enough area for the complaining to be worth the effort, in the case of a low approval rate. There is also a special case of 0 approval rate in which agents will never be able to protect any area and the simulation will always collapse. Since the approval rate at Frøya so far has been 0 with two out of two complaints being rejected, there has so far been no positive indication that complaining works. Therefore it is equivalent with a government which has a complaint approval rate of 0, which always causes collapse. In order to make the situation more fair for fishers, the government may be encouraged to approve more complaints.

Changing the payoff matrix

According to the game theoretic interpretation, the government, if assumed to be rational, acts in such a way to maximize its payoff. One way to change its behavior to approve more of the fishers' complaints, is to alter the government's payoff matrix. More explicitly, a policy needs to be constructed under which the government receives benefits if it approves fair complaints, or receives a penalty if it does the opposite. Another way of stating this is that the government needs to align its payoff matrix with that of the fishers, so that the government receives a benefit when doing something that's beneficial for the fishers, like approving their complaints. Applying a policy that changes the payoff matrix of the government in order to approve more complaints will according to the simulation results allow larger areas to be protected by the fishers' complaints. Allowing for a non-zero complaint approval rate may also cause fishers to perceive their complaints as useful, and cause complaining behavior to be learned as in stabilized runs of the simulation.

A way to alter the payoffs to the regional government is to punish them for environmental damage. Environmental damage is caused by aquaculture facilities being built, and according to the fishers in the workshop (chapter 7), the damage significantly hurts the ecosystem. The environmental damage can therefore be seen as a tragedy of the commons-type of situation, where aquaculture companies use the common resource of healthy waters, leaving unhealthy ocean behind. This tragedy is negative for the regional government in the long run, but there may be a lack of short-term consequences that truly motivate anti-damage behavior. Introducing a penalty for environmental caused to the ocean may help rise approval rates of fishers' complaints about aquaculture establishment, because the fishers' and government's payoff matrices align. By approving complaints they prevent aquaculture expansion which reduces environmental damage, and results in less penalty.

9.1.3 Factoring environmental damage into complaint rates

So far we have discussed prevention of environmental damage as a tool for aligning the government's and the fishers' priorities. However, environmental damage is important for the fishers themselves, and varying amounts of it may affect their complaint rates. Aquaculture facilities that pollute less may be more welcome to the fishers than those who pollute more, which may result in different outcomes. Experiment 4 was conducted in an attempt to chart the relationship between environmental damage caused by aquaculture facilities and fishers' complaining behavior.

The experiment yielded some surprising results. Looking at table 8.6, there are three different configurations that represent varying levels of environmental damage: medium damage (50 m radius, 1 proportion), high damage (100 m radius, 1 proportion) and low damage (50 m radius, 0.5 proportion). However, the collapse rate is lower in the high damage scenario. This may be a result of fisher agents more quickly realizing that the aquaculture facilities cause damage that is important to their payoffs, and managing to protect areas more often. However, the amount of open cells in the stabilized world is the lowest in the lowest-damage scenario, which indicates that protection is more successful when the damage is low. This may be explained by the fishers spreading over a larger area when more cells are viable to harvest from, and the collective votes of the fishermen are therefore able to protect a larger area, rather than different fishermen voting for the same cells, beyond the necessary threshold to protect it.

9.2 How to Remedy False Complaints

The simulation has no explicit notion of false complaints because the fishers' complaining decisions are made by a neural network that is tuned through payoffs which are not influenced by honesty. Measuring honesty would anyway be difficult, since the opinion that a location is worthy of protection is more or less subjective. Deciding if a fisherman complaints with good reason or not is difficult because the fishers in the simulation will always complain if it gives an advantage, or even if there is even a tiny chance that it gives an advantage. This effect may reflect real situations where fishers can complain endlessly about the establishment of aquaculture facilities without being punished for "false" complaints.

9.2.1 Complaints during the stabilized state

Looking at fig. 8.4, there are no unworked cells that are still being protected by the fishermen through complaints. Fishers use all the open cells in a cycle. Especially If there were cells which were never used, the complaints about them may be regarded as "false", since aquaculture there would not prevent fishers from fishing. On the other hand, environmental damage may affect the surrounding area and this may be predicted by fishers and become a factor in their decision making. Since different agents may estimate the risk of damage form a facility differently, the classification of complaints as false is not very straight-forward since complaints are based on subjective decisions.

Sub-optimal distribution of fishers may also be a cause of false complaints. During the movement when the fishers have reached a stabilized outcome, there are some situations where more than one fisher occupies a single slot while other slots are free of fishers. Fishers move to a new spot when they experience that the current location is bad, even if it is only bad because two fishers are sharing it without them being aware of it. This is an inefficient use of the resources, since a larger spread of fishers would lead to more optimal harvesting. If a more optimal harvesting plan was conducted, some cells may never be fished and aquaculture facilities could establish there. This would in turn lead to fewer complaints overall, reducing the number of false complaints.

One problem with this deduction is that fishers in real life might be aware if they share a fishing resource, either through observing each other directly, or the experience of less fish might be enough to understand that another fisher is using the same location. The simulation model assumes that fishers do not know about the other fishers' location and they can only determine the fishing quality of a location through experience, by using it. The new experience will be recorded as the current performance of that cell, even if it was better before. The overall distribution of fishers is therefore self-organized as a global distribution emerging from the local interactions of fishers experiencing slots and moving around. The system may end up in sub-optimal attractor states consisting of imperfect fisher distributions. During the workshop, the fishers said that they respect other fishers' areas by staying away from ones they know about that belong to others. However, if aquaculture expansion pushes the fishers away from their normal areas, they might end up sharing a space. If they are aware that two fishers share the same location, they might coordinate and one fisher moves somewhere else. However, fishers may not be able to deduct that two are sharing the same location. In this situation, the organization of fishing may benefit from a fishing plan that distributes fishers to locations in an optimal fashion.

9.2.2 When complaining takes place

Experiment 3 studied how complaints were distributed among hearing rounds. In the beginning of the simulation of a stabilized run, complaining occurs at all three hearing stages. However, complaining in round 3 disappears when the level of complaints stabilizes. Figure 8.6 shows how all complaining is done in the first two rounds when the level has stabilized. This is because the first two hearing-planning cycles are sufficient to stop any problematic aquaculture expansion, as all cells are classified as reserved zones after only two rounds.

Chapter 10

Future Work

The project was conducted and software created with the goal in mind that both the topic and the simulation may be expanded and further explored in the future. Several areas of improvement and further research have been identified, and they are described in this chapter.

10.1 Focusing on Other Decision-Making Agents

Since the project focused on fishers' decisions, it largely ignored other stakeholders identified by Tiller et al. (2014). Especially tourist and civilians are in section 4.1 identified as important stakeholders with influential opinions on aquaculture expansion. The inclusion of tourists and civilians is important both to see how their actions can influence the decisions and situations of fishers, as well as shifting the focus to studying their actions. Making tourists and civilians more intelligent, including their work and creating more relevant priorities are important to fleshing out the simulation and making it more realistic.

In addition to tourists and the generic "civilian", other stakeholders that are interested in the development in Frøya may be simulated. Both high schoolers, enthusiasts, foreign workers and academics were identified by Tiller et al. (2014) as important stakeholders. In Frøya the high school offers programs for education within both the aquaculture and fishing industries. High schoolers could therefore for instance be simulated as growing up and finding a work place, and their priorities would involve finding work that fits their education.

Another option for the simulation is to expand it to fit with more situations than the one in Frøya. Currently the simulation is built for modeling only the situation in Frøya, and even if generalizing it was a goal, more work could certainly be done in this direction. Other places could be explored, especially those with different attitudes toward aquaculture expansion, in order to modify the simulation so it could cover more general situations.

In real life, the government appeal process is also much more complex than the simulated complaint approval process. The hearing is divided into three stages that are treated by separate organs. During subsequent hearing rounds agents cannot propose new complaints like they can in the simplified simulation; they can only present new and old arguments in different fashions. Attempting to simulate this process was beyond the scope of this project, but it is certainly a possible topic for future extensions. The algorithm used by the government to approve or deny complaints is also very simple, and not subject to any learning process. By introducing more complex decision making that actively used the priorities of the government one could get a better idea of how to understand, work with, and potentially create policies for the government organs.

The simulation was created with two different complaint decision mechanisms, neural networks and a rule-based approach, but only the neural network was used for experiments. Using an evolved artificial neural network has the advantage that relatively few assumptions need to be made about the decision making process of the agents, but understanding its process becomes more difficult for the same reason; the implicit assumptions made by the network are abstracted away. Other complaint decision algorithms could be implemented in future extensions, and compared against each other and with the neural network-implementation. Case-based reasoning is a promising algorithm that could be applied for this purpose, since it also requires few assumptions. Case-based reasoning is explained in section 2.4.3 and an advantage is that it would make the reasons for fishers' decisions more explicit, giving a full representation of the closeness of the decision situation to other decision situations.

10.2 Decision Support System

During development of the software, it was theorized if it could be used as part of a decision support system for policy makers regarding fishers and aquaculture expansion. The CORMAS-based fisher simulation developed by Cleland et al. (2012) and the pest-control simulation developed by Rebaudo et al. (2011) were both used as teaching tools to educate stakeholders about the situation. If this project is continued toward more realistic representations of the environment, the intelligent agents and their interaction structure, the software may become useful as a teaching tool both for policy makers and stakeholders. A simulation can give an immersion into the problem that's richer than what a description can do. Providing insights into the roles of the local government, the aquaculture owners and the fishers in the local community, as well as clarifying the value of the fishing industry, as a relatively environmentally-neutral industry compared to aquaculture, to stakeholders who underestimate it, could be valuable functions of the software. As such the software could be helpful to both fishers, the local government and the local community in general.

Another use for a realistic simulation could be testing policies before putting them into practice. For example, a policy distributing fishers' fishing areas optimally as discussed in section 9.2.1, could be proved valuable in the simulation before being applied to the real fishers, ultimately saving resources.

Insights gathered from using the simulation can help when designing policies.

Chapter 11

Conclusions

This project was based on scenario where real fishers don't complain about the establishment of aquaculture which competes with them, even if they have to opportunity to. Aquaculture is both beneficial as well as pose a threat to the fishers, so the decision to complain is not straight-forward. At the same time, false complaints become an issue when the fishers gain too much power in the coastal planning, where they can completely lock-out new aquaculture facilities. This thesis used simulation and analysis to see the issue from two sides: fishermen experience that their complaints don't work, while the aquaculture-friendly local government and aquaculture companies themselves face the risk that fishers overstate the danger of aquaculture.

The simulated fishers often behave similarly to those at Frøya; fishers learn to not complain. This similarity gives the opportunity to draw insights into the real-world situation from the simulations.

It was confirmed through simulation that the cost of complaining is high for fishermen. Even if the monetary cost of complaining is zero, fishers experience a more subtle cost of ending up sharing fishing spots with others when they complain. The expected return of complaining is at the same time low, and fishers learn that complaining have no benefit, similarly to the Frøya fishers' experience.

Since fishers can end up harvesting from the same locations, the resources may be used sub-optimally. Ensuring good organization and distribution of fishing efforts can prevent false complaints and lead to more efficiency, simultaneously allowing more aquaculture and more profits for fishers.

Charging fishers a fee for complaints do not seem to be a good solution to false complaints because the complaint level is already often very low. However, investigation by a third-party into the state of coastline resources may be a promising tactic since it can chart environmental damage caused by aquaculture facilities in order to discourage false complaints. This information can be used to penalize a government which causes the environmental damage by releasing too many aquaculture licenses. In this way the government's payoff matrix can be changed to promote the fishers' priorities as well as the aquaculture organizations' priorities.

A general decision support system based on the software and insights from this project can lead to a more fair division of coastal resources. If adapted to other areas and extended to reflect the stakeholders' situations more realistically, the system may be applied in places and situations beyond Frøya's fishers.

Bibliography

Cleland, D., Dray, A., Perez, P., Cruz-Trinidad, A., Geronimo, R., 2012. Simulating the dynamics of subsistence fishing communities: REEFGAME as a learning and data-gathering computer-assisted role-play game. Simulation & Gaming 43 (1), 102–117.

Cruz, F., Pereira, A., Valente, P., Duarte, P., Reis, L. P., 2007. Intelligent farmer agent for multi-agent ecological simulations optimization. In: Progress in Artificial Intelligence. Springer, pp. 593–604.

Davidsson, P., January 2002. Agent based social simulation: A computer science view. Journal of artificial societies and social simulation 5 (1).

Floreano, D., Mattiussi, C., 2008. Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies. The MIT Press, Ch. 1: Evolutionary Systems, pp. 1–100.

Gintis, H., 2000. Game theory evolving: A problem-centered introduction to modeling strategic interaction. Princeton University Press, mathematical introduction to evolutionary game theory, including dynamic fields etc.

Groner, M., Cox, R., Gettinby, G., Revie, C., 2013. Use of agent-based modelling to predict benefits of cleaner fish in controlling sea lice, lepeophtheirus salmonis, infestations on farmed Atlantic salmon, Salmo salar L. Journal of fish diseases 36 (3), 195–208.

Heylighen, F., 2001. The science of self-organization and adaptivity. The encyclopedia of life support systems 5 (3), 253–280.

Janssen, M. A., Ostrom, E., 2006. Inequality, Cooperation, and Environmental Sustainability. Princeton University Press, Ch. 4: Adoption of a new regulation for the governance of common-pool resources by a heterogeneous population, pp. 60–96.

Laland, K., Rendell, L., 2010. Social learning: Theory. In: Breed, M. D., Moore, J. (Eds.), Encyclopedia of Animal Behavior. Academic Press, Oxford, pp. 260–266.

Le Page, C., Becu, N., Bommel, P., Bousquet, F., 2012. Participatory agent-based simulation for renewable resource management: The role of the cormas simulation platform to nurture a community of practice. Journal of Artificial Societies and Social Simulation 15 (1), 10.

Mitchell, T., 1997. Machine Learning. McGraw-Hill International, Ch. Chapter 8: Instance-Based Learning, pp. 230–248.

NumPy, 2014. Numpy. Accessed 09.05.2014.
URL `http://www.numpy.org/`

Pereira, A., Duarte, P., Reis, L. P., 2004. Agent-based simulation of ecological models. In: Agent-Based Simulation.

Pereira, A., Reis, L. P., Duarte, P., 2009. Ecosimnet: A multi-agent system for ecological simulation and optimization. In: Progress in Artificial Intelligence. Springer, pp. 473–484.

Rebaudo, F., Crespo-Pérez, V., Silvain, J.-F., Dangles, O., 2011. Agent-based modeling of human-induced spread of invasive species in agricultural landscapes: Insights from the potato moth in Ecuador. Journal of Artificial Societies and Social Simulation 14 (3), 7.

Russell, S., Norvig, P., 2010a. Artificial Intelligence: A Modern Approach, third edition Edition. Pearson, Ch. 2: Intelligent Agents, pp. 34–63.

Russell, S., Norvig, P., 2010b. Artificial Intelligence: A Modern Approach, third edition Edition. Pearson, Ch. 16: Making Simple Decisions, pp. 610–644.

Tiller, R., Richards, R., Salgado, H., Strand, H., Moe, E., Ellis, J., 2014. Assessing stakeholder adaptive capacity to salmon aquaculture in Norway. Consilience: The Journal of Sustainable Development 11 (1), 62–96.

Wooldridge, M., 2002a. An introduction to MultiAgent Systems. John Wiley & Sons, LTD, Ch. 2: Intelligent Agents, pp. 16–46.

Wooldridge, M., 2002b. An introduction to MultiAgent Systems. John Wiley & Sons, LTD, Ch. 8: Communication, pp. 163–188.

wxPython, 2014. wxpython. Accessed 09.05.2014.
URL `http://www.wxpython.org/`

Appendix A

Documentation of the program configuration

Configuration of parameters, initial conditions and other settings is done through a single *JSON*[1] file. There are 9 top-level objects, one for each of the six agent types: *fisherman*, *aquaculture*, *tourist*, *civilian*, *government*, *municipality*, and three more for general configuration: *global*, *world* and *interface*. The configuration objects for the agent types all specify the priorities for that agent, but only the fishermen's priorities are used in the simulation. Listing A.1 shows an example configuration file used to setup the program. Below is a complete description of all fields.

Listing A.1: Example of a configuration file that uses a 15x15 grid, a 1 cell blocking radius and a neural networks for the voting decision.

```
{
    "global": {
        "num max complaints":                10,
        "max hearing rounds":                3,
        "aquaculture blocking radius":       25,
        "aquaculture damage radius":         100,
        "aquaculture damage proportion":     1,
        "aquaculture in blocked":            false
    },
    "world": {
        "structure": {
            "class": {
                "type":  "class",
                "class": "FisherSimulation.world.GridStructure"
            },
            "width":                         15,
            "height":                        15,
            "cell width":                    25,
            "cell height":                   25,
```

[1]For a complete specification of the (very) simple *JSON* file format, see `http://json.org`

```
                "neighbourhood type":          "von_neumann"
            },
            "good spot frequency": 0.1
        },
        "interface": {
            "print messages": true
        },
        "fisherman": {
            "num":         20,
            "priorities": {
                "OwnProfits":                    10.0,
                "CommunityWealth":               2.0,
                "WildFishPrice":                 2.0,
                "FishingIndustryExisting":       3.0,
                "NaturalFishHealth":             4.0,
                "AquacultureIndustryExisting":   1.0
            },
            "learning mechanism": {
                "class": {
                    "type":  "class",
                    "class": "FisherSimulation.ga.Evolution"
                },
                "phenotype class": {
                    "type":  "class",
                    "class": "FisherSimulation.ga.FishermanVotingNN"
                },
                "genotype class": {
                    "type":  "class",
                    "class": "FisherSimulation.ga.FishermanNNGenotype"
                },
                "elitism":              3,
                "selection mechanism":  "rank selection",
                "crossover rate":       0.005,
                "mutation rate":        0.005,
                "genome mutation rate": 0.00005

            },
            "voting mechanism class": {
                "type":  "class",
                "class": "FisherSimulation.ga.FishermanVotingNN"
            }
        },
        "aquaculture": {
            "priorities":       {},
            "work efficiency": 10,
            "voting mechanism class": {
                "type":  "class",
                "class": "FisherSimulation.vote.AlwaysApprove"
            }
        },
        "civilian": {
            "num":         0,
            "priorities": {},
            "voting mechanism class": {
                "type":  "class",
                "class": "FisherSimulation.vote.AlwaysApprove"
            }
```

```
    },
    "tourist": {
        "num":        0,
        "priorities": {},
        "voting mechanism class": {
            "type":  "class",
            "class": "FisherSimulation.vote.AlwaysApprove"
        }
    },
    "government": {
        "priorities": {},
        "decision mechanism class": {
            "type": "class",
            "class": "FisherSimulation.dm.ApproveProbability"
        },
        "complaint approval probability": 0.5
    },
    "municipality": {
        "priorities": {},
        "planning mechanism class": {
            "type":  "class",
            "class": "FisherSimulation.dm.EverythingAquaculture"
        }
    }
}
```

A.1 Preprocessing

The configuration is sent through a basic processor that performs some conversion operations.

A.2 Python Classes

Python classes are detected by looking for the `type` field in objects. When the field is detected with the value “class”, the whole object is assumed to describe a class to be plugged-into the system. The field `class` is expected to be in the same object, with a string value that dictates the complete Python import path specification for the class to be used. The module with the class is then imported, and the object in the configuration dictionary is replaced with a reference to the loaded class. For example, in listing A.1 in the `fisherman` object the `learning mechanism` object has a `class` field which is defined in this way. When converted, the `class` field of `learning mechanism` will refer to the Python class `Evolution` in the GA module in the **FisherSimulation** package.

A.3 Priorities

Priorities are also converted to their respective objects, which are defined in the PRIORITY module of the **FisherSimulation** package. The available priorities documented in

Table A.1: Priorities names and description

Priority Name	**Description**
OwnProfits	The capital of the agent.
WildFishPrice	Price of fish caught by fishermen.
SalmonPrice	Price of fish farmed by aquaculture companies.
CommunityWealth	Average capital of the community.
FishingIndustryExisting	The relative strength of the fishing industry, calculated by how many fishermen are active.
AquacultureIndustryExisting	Strength of aquaculture industry, calculated by number of aquaculture facilities.
NonintrusiveAquaculture	For agents that have a set of cells they care about for environmental reasons, this priority is a representation of whether those cells are "polluted" by aquaculture, and in that case to what degree.

table A.1. Whenever the processor sees a field named `priorities`, it converts each dictionary key in that object to a priority object. The weights for each priority are specified by a number.

A.4 Fields

This section gives a complete description of all the fields and their expected values. This simulation uses all of *JSON*'s value types: object, array, string, number (integer or float), true, false and null. Keys are strings, so they contain spaces.

`global` Defines settings that apply to the whole system, or cannot be fit in any other configuration objects. There are three global settings.

- **`num max complaints`** The maximum number of complaints a single agent can issue in a single hearing is specified as an integer.
- **`max hearing rounds`** The maximum number of cycles of (re)planning, hearing and government decision that can be held for a plan is specified as an integer.
- **`aquaculture blocking radius`** The blocking radius, which is an area in which fishing and potentially building new aquaculture is not allowed, is specified as a number of meters.
- **`aquaculture damage radius`** The radius of the environmental damage that is caused by aquaculture facilities is specified in meters, as a number.
- **`aquaculture damage proportion`** The amount of damage inflicted on the cell aquaculture is placed on is specified as a number between 0 and 1. The damage declines linearly from the center until 0 damage is inflicted at the radius' perimeter.

aquaculture in blocked The setting of whether aquaculture should be allowed to be built in blocked cells or not, expects a boolean value of *true* or *false*.

world The `world` object contain settings that apply to the world, which is the top-level entity that organizes physical position.

structure The first member of `world` is an object describing its structure, which is the entity that deals with the implementation of space in the simulation. It expects several settings that are custom to the implementation. The only structure defined is a grid structure.

class The class that defines the structure is defined in the standard way of defining classes, with an object with `type` and `class` attributes, as described in appendix A.2.

width The number of horizontal cells in a `GridStructure` is defined by an integer.

height The number of vertical cells in a `GridStructure` is defined by an integer.

cell width The cell width and height in a structure is important for blocking radius around aquacultures, which is specified in meters. The cell width is given as a number.

cell height The cell height in a structure is given as a number of meters.

neighbourhood type The neighbourhood type of `GridStructure` is specified as a string of either "von_neumann" or "moore", which corresponds to Von Neumann and Moore neighbourhood types. The difference is that Moore neighbourhood considers diagonally touching cells as neighbours, whereas Von Neumann does not.

good spot frequency The other member of the `world` object is a setting that applies regardless of the structural limitation; the frequency of good spots. Since there is a base assumption that the world is divided into slots or cells, this frequency expects a floating-point number regardless of the structural implementation.

interface The `interface` object contains settings that apply to the graphical (or otherwise) user interface.

print messages Since there are quite a lot of messages produced by the system, showing them all may slow down the interface. Therefore there is an option to turn them on or off, which accepts a boolean value.

fisherman The `fisherman` agent, like all other agents, has a list of priorities, but also has voting and learning mechanisms. The fisherman is the central agent of this implementation of the system.

`num` Since a number of fishermen is initialized at the beginning of the simulation, the `fisherman` object receives an integer to represent how many are created.

`priorities` Fishermen are prioritizing agents, and they are initialized with a list of priority values, specified as described in appendix A.3.

`learning mechanism` An important part of the simulation is how fishermen adapt and change their behavior by experience. The object contains settings to customize the process. A user-defined class can be specified to handle the learning, and this object will be received by the instances, so any custom configuration must be specified in this object.

- **`class`** A Python class must be specified to process the learning itself, as described in appendix A.2. The default learning mechanism is the `Evolution` class in the `GA` module, which uses phenotypes and genotypes and a custom configuration class.
- **`phenotype class`** The `ga.Evolution` class requires the specification of the phenotype it is being applied to, which is the same as voting mechanism class for the agent. The class specification works as described in appendix A.2.
- **`genotype class`** The `ga.Evolution` class also requires the specification of the genotype, which is built in unison with the phenotype. The class specification works as described in appendix A.2.
- **`elitism`** The `ga.Evolution` class can utilize elitism, which is a number of the best-performing individuals whose genotypes are preserved for the next round. The degree of elitism is specified as an integer.
- **`selection mechanism`** The `ga.Evolution` class can utilize different selection mechanisms, specified as a string. The different selection mechanisms are:
  - **"rank selection"** Linear rank selection sorts the genotypes by the fitness values of their corresponding phenotypes. One lottery ticket is assigned to the lowest ranking member, two to the second-lowest and so on. The selected individual is randomly chosen based on the weighed probabilities described by the amount of lottery ticket each phenotype receives.
- **`crossover rate`** The crossover rate is how often the crossover operation is performed on any pair of genotypes. It is specified by a number between 0 and 1.
- **`mutation rate`** The mutation rate is how often any one genotype is selected for mutation. It is specified as a number between 0 and 1.
- **`genome mutation rate`** The genome mutation rate is how often any bit in the genome is flipped of a genotype that has been selected for mutation. It is specified as a number between 0 and 1.

`voting mechanism class` As fishermen are voting agents, a voting mechanism class must be specified, in the way described in appendix A.2. The default implemented voting mechanism classes are:

`FishermanVotingNN` (from FISHERSIMULATION.GA A neural network that iterates the cells in the plan and for each one decides to vote or not depending on the inputs: *a*) the quality of the current “home” resource; *b*) the quality of the targeted resource, if known; and *c*) the distance between the two resources . The output is one neuron that decides to complain when its value is above average. The corresponding genotype is `FishermanNNGenotype`(from FISHERSIMULATION.GA).

`FishermanVotingRules` (from FISHERSIMULATION.GA A rule-based decision mechanism that has three forms, where one is selected randomly: *a*) complain or not as a probabilistic decision based on the distance between the home cell and the targeted location; *b*) always complain about 10 random cell in the plan; or *c*) never complain . Its corresponding genotype is `FishermanRulesGenotype` (from FISHERSIMUATION.GA).

`AlwaysApprove` (from FISHERSIMULATION.VOTE The simplest mechanism is by default applied to agents other than fishermen: never complain. It has no corresponding genotype and can thus not be used with the `Evolution` learning mechanism from the GA module.

`aquaculture` The settings for aquaculture agents include priorities and voting mechanism. A learning mechanism may be either applied or omitted. Unlike other non-single agent types, aquacultures are not initialized as a group at the beginning of the simulation, so no `num` parameter is expected.

`priorities` Aquacultures are prioritizing agents who are initialized with a list of priorities, as described in appendix A.3.

`work efficiency` The efficiency of working an aquaculture facility is given as a number. This number is equal to the amount of capital harvested by aquaculture agents.

`voting mechanism class` Aquacultures are voting agents, so they are initialized with a voting mechanism. This field takes a voting mechanism class, as described in appendix A.2. The default class is `AlwaysApprove` from VOTE in the **FisherSimulation** package.

`civilian` The civilian agents are prioritizing and voting agents, so they require the specification of priorities and a voting mechanism class. A learning mechanism class can also be defined, as described in appendix A.2.

`num` A number of civilians are created at the start of the simulation, and the number of them is defined by an integer.

`priorities` Priorities for civilians are specified as described in appendix A.3.

`voting mechanism class` A voting mechanism is defined for civilians as a class, as described in appendix A.2.

`tourist` Tourists are prioritizing and voting agents, so they require the specification of priorities and a voting mechanism class. A learning mechanism class can also be defined, as described in appendix A.2.

`num` A number of tourists are created at the start of the simulation, and the number of them is defined by an integer.

`priorities` Priorities for toursits are specified as described in appendix A.3.

`voting mechanism class` A voting mechanism class is defined for civilians as a class, as described in appendix A.2.

`government` The government is a single object, so it has no number parameter. It does have priorities, but by default they are not used. A configurable decision mechanism is applied when deciding to approve complaints or not.

`priorities` Government are initialized with a set of priorities, as described in appendix A.3.

`decision mechanism class` A decision mechanism needs to be specified in order for the government to approve and disapprove complaints. The class needs to fulfil the `GovernmentDecision` interface in the ENTITIES module, and is specified as described in appendix A.2. The default classes are:

`ComplaintApproveMoreThanOne` (from DM) Approve complaints for cells that have two or more votes behind them. Respects the `max hearing rounds` configuration.

`ApproveProbability` (from FISHERSIMULATION.DM) This implementation makes the government approve each complaint from an agent on a cell with a configurable probability.

`complaint approval probability` The probability of a single complaint being approved is a part of the `ApproveProbability` government decision mechanism and is given as a number between 0 and 1.

`municipality` The municipality is responsible for the planning, which is configurable through selecting a class. Municipalities also have priorities which are unused by default.

`priorities` Municipality is initialized with a set of priorities, as described in appendix A.3.

`planning mechanism class` A planning mechanism must be specified for the municipality to create and review coastal plans. The class is specified as described in appendix A.2, and the planning mechanism needs to implement the `PlanningMechanism` interface in the ENTITIES module. The default class is `EverythingAquaculture` from FISHERSIMULATION.ENTITIES.

Appendix B

Documentation of modules and classes included in the system

Most of the source code is contained in the **FisherSimulation** package, which is responsible for all the simulation code. Other packages are **config** for configuration, **gui** for graphical user interface, and **cli** for command-line interface. Table B.1 shows an overview of the modules and classes contained in the **FisherSimulation** package. Table B.2 shows an overview of the modules and classes contained in the **gui** package.

B.1 Modules in the FisherSimulation package

AGENT The agent module defines various abstract classes for agents to be based on. The functionality is kept separate, and can be combined with multiple inheritance.

`IdentifyingAgent` Agents identify with a unique ID that is provided by this class. All agents in the system inherit from this class.

`CommunicatingAgent` Provides methods to communicate with other agents, through the directory. When an agent that inherits from this class is created, it should call the *register* method to be registered in the central directory. All agents in the system are communicating agents.

`VotingAgent` Provides methods for voting during hearing rounds. Voting agents include fishermen, aquaculture agents, civilians and tourists.

`WorkingAgent` Working agents are agents that do something in the Fishing and Working phase. The *work* method is called for each working agent during that phase. Working agents include aquaculture agents, fishermen, civilians and tourists.

`PrioritizingAgent` Prioritizing agents are agent that care about things in the world, which is expressed through priorities. Priorities are configured using

Table B.1: Overview of modules and classes in the simulation process.

Module	Classes
SIMULATION	`Simulation`
DECISIONS	`PlanningMechanism`
	`GovernmentDecision`
ENTITIES	`AgentFactory`
	`ProducedAgent`
	`AquacultureSpawner`
	`EverythingAquaculture`
	`Municipality`
	`ComplaintApproveMoreThanOne`
	`License`
	`Government`
	`Fisherman`
	`Aquaculture`
	`Tourist`
	`Civilian`
WORLD	`Map`
	`Slot`
	`AbstractStructure`
	`FishingStructure`
	`GridStructure`
	`TorusStructure`
GA	`Phenotype`
	`Genotype`
	`FishermanVotingNN`
	`FishermanNNGenotype`
	`FishermanVotingRules`
	`FishermanRulesGenotype`
AGENT	`IdentifyingAgent`
	`CommunicatingAgent`
	`VotingAgent`
	`PrioritizingAgent`
PHASES	`Round`
	`StepResult`
	`Step`
	`DecisionStep`
	`CoastalPlanning`
	`Hearing`
	`GovernmentDecision`
	`Fishing`
	`Building`
	`Learning`
NN	`LabeledNeuralNetwork`
	`Neuron`
DIRECTORY	`Directory`
VOTE	`Vote`
	`AlwaysApprove`
MARKET	`Market`
UTIL	
PLAN	`CoastalPlan`
	`PlanEntity`
	`Complaint`
	`Decision`
PRIORITY	`Influences`
	`Priority`
MESSAGES	`MetaInfo`
	`BroadcastMetaInfo`
	`Message`
	`Reply`
	`Inform`
	`AquacultureSpawned`
	`PlanHearing`
	`VoteResponse`
	`VoteResponseInform`

Table B.2: Modules and classes for the graphical user interface.

Module	Class
GUI	`PhaseResultEvent`
	`StopEvent`
	`WorkerThread`
	`Info`
	`Messages`
	`Window`
GRAPHS	`Graphs`
CONTROLS	`Controls`
WORLDMAP	`WorldMap`

the configuration file, as explained in appendix A.3. All agents in the system are prioritizing agents.

DECISIONS The decisions module contains abstract interfaces for decision making mechanisms. Decision making mechanisms are applied to voting, the government decision and planning.

`VotingDecisionMechanism` During each hearing round, the *decide_votes* method in its voting decision mechanism is called for each voting agent. An implementation needs to override this method, which takes arguments for: *a*) the agent; *b*) the coastal plan being voted on; *c*) the current state of the world (a `world.Map`); and *d*) the maximum number of complaints an agent can make. It should return a list of `Vote` instances (from the VOTE module). It also needs to implement the *new* method, which is a class method that should create an instance of the implementing class and attach it to the given agent with the *add_voting_mechanism* method that is defined for `VotingAgent` instances. The *new* method takes arguments for: *a*) the agent; *b*) the configuration object for the agent class; and *c*) the world instance (`world.Map`).

`GovernmentDecision` When the government decides whether to approve or deny complaints, and thus indirectly deciding if the whole plan is to be reviewed or not (by one or more complaints being approved), it utilizes its government decision mechanism that is an implementation of this interface. The constructor of a class implementing this interface needs to receive a single argument, which is the configuration object for the agent class (`Government`). The interface defines two non-constructor methods. The first is *round_reset* which is called every time a new round is started. It takes no arguments. The second is *decision* which is called when the government makes a decision, and it receives a single argument which is a dictionary that maps `world.Slot` instances (map cells) to `plan.Complaint` objects. The expected return value is the dictionary with complaints approved or not.

`PlanningMechanism` The municipality uses a planning mechanism during both initial creation and subsequent revision of the plan. Both are done through a single method in a class that implements this interface. The constructor should accept a single argument which is the configuration object for the agent class (`Municipality`). The *create_plan* method receives three arguments when called: *a*) the world map, as a `world.Map` instance; *b*) coastplan, which is the previous coastal plan if there was one (a `plan.CoastalPlan` instance); and *c*) a list of `plan.Complaint` objects to review the plan based on, or **None** if there was no previous plan.

DIRECTORY In the simulation, agents communicate using messages which are sent through a central repository. The directory module contains the implementation of this functionality.

`Directory` The directory class is the implementation of the messaging hub. It records and propagates messages between agents, and this recording can be displayed for every step in the user interface. Whenever communicating agents are created, they register to the directory using the *register_communicating_-agent* function. Later on messages can be sent as both broadcasts to a group of agents, typically either to all voting agents, or to all fishermen, or as single-target messages with a specified recipient.

DM This module contains some implementations of decision making mechanisms.

`ApproveProbability` This implementation of a `GovernmentDecision` (in DECISIONS) works by simply approving complaints with a configurable probability.

`ComplaintApproveMoreThanOne` This implementation of the government decision interface (from DECISIONS) works by approving a complaint for each cell that receives more than one complaint about it.

`EverythingAquaculture` This implementation of the planning mechanism interface (`PlanningMechanism` in DECISIONS) works by assigning everything as aquaculture, and removing cells with approved complaints when reworking the plan.

DO In order to make the GUI replaceable and possibly work in a different thread, or even process, from the simulation, the user interface communicates with the simulation using simple direct-object representations of data. In Python, this means that the objects are straightforward and "pickable". They contain merely a number of attributes and a static factory method.

`Complaint` Contains fields for coordinates (a 2-tuple of integers) for the cell that is being complained about, and the agent that complains.

`InterfaceConfig` The configuration file is processed by the simulation and not the user interface, so the parts that are relevant to the interface needs to be sent back. It contains one field to toggle printing (all) messages sent in the system.

`Simulation` General information about the initial state of the simulation, with fields for map, a list of fishermen, aquaculture agents, civilians, tourists, interface configuration and the maximum number of complaints a single agent, for a whole round of simulation (maximum number of rounds multiplied by maximum number of complaints per agent).

`WorkingAgent` Non-government, non- municipality agents are considered to be working agents, and their information is transmitted using specific classes derived from this, with no extra functionality. Their fields include an identifier string and their capital as a number.

`Map` The map mentioned for the simulation object is described using this class, which contains a single attribute which is a 2-dimensional array of Slot objects.

`Slot` The cells contained in the map are described by an instance of this class, which contains attributes for whether the cell has spawning fish or not, if there is aquaculture, if there is a fisherman fishing there, if it is land, if it is blocked, a list of fishermen fishing in the spot, the number of fishermen fishing there, and a quality field which measures the amount of natural fish at the location.

`Message` Messages are the communicating entity between agents, and they are represented by a sender ID, recipient ID, list of recipients in case of a multiple-recipient broadcast, the contents summarized as a string, and a string indicating whether the message is a broadcast (by "broadcast"), or a single-target message (by "single"). This class also implements a to-string function that typesets the message in a human-readable format.

`PhaseReport` For each step or phase, information about what happened needs to be sent back from the simulation to the user interface. The information is sent as a phase report, which includes fields for the name of the phase, a list of messages sent, the map with `Slot` (from `DO`) instances for changed cells, a boolean indicating if it is a new round, a generic dictionary data field, a string indicating the name of the next phase, a number indicating the current round, and a list of complaint issued in the phase.

ENTITIES This module contains the definitions of all the agents used in the system, as well as some factories for producing them.

`AgentFactory` The agent factory is initialized with the full configuration file, and provides methods for creating agents with the proper configuration object.

`ProducedAgent` This abstract class implements common functionality for the fisherman, aquaculture, civilian and tourist agents. All these agents have some knowledge of cells, a capital and a voting decision mechanism. The *add_-voting_mechanism* method needs to be called by the voting mechanism class. During hearing, the *hearing* method calls the *decide_votes* method in the voting mechanism instance.

`AquacultureSpawner` This class is used as a singleton that is responsible for finding a cell for aquaculture to occupy. The cell is chosen randomly from all aquaculture sites in the coastal plan.

`Municipality` The municipality has methods for collecting and distributing taxes from workers to community members, and for creating and distributing the coastal plan. It uses a planning mechanism to create the coastal plan.

`License` The license object is an empty placeholder object used by the government to indicate that an aquaculture company has been given a building license.

`Government` The government distributes licenses to the municipality, and collects complaints and decides whether to approve or rework the coastal plan based on them. It uses a government decision mechanism class to decide based on the votes.

`Fisherman` Fishermen do fishing in two steps; first they find a spot using the *find_fishing_spot* method, and second they work that spot in the *work* method.

`Aquaculture` Aquaculture agents work at a set, configurable rate.

`Tourist` Tourists are initialized with a set of prioritized slots on which they can base their votes. They do work, but do nothing.

`Civilian` Civilians work, but do nothing.

GA The genetic algorithm module implements the artificial evolution that by default is used for learning by the fishermen, as well as phenotypes and genotypes for decision making mechanisms the fishermen can employ, and learn.

`LearningMechanism` The learning mechanism interface provides an abstract interface for learning mechanisms. It contains a single method *learn*, which takes no arguments. The constructor takes a list of agents that the learning mechanism applies to.

`EvolutionarySelectionPhenotype` This simple data structure provides a straightforward coupling of phenotype and fitness for applying the tuple of these in various evolutionary selection mechanisms.

`Evolution` This learning mechanism calculates the fitness of each individual agent and applies a selection mechanism to select which phenotypes should be bred for the next generation. Then it applies mutation to these genotypes, and creates new phenotypes based on the genotypes. Lastly it distributes these phenotypes to the agents as decision making mechanisms. The only selection mechanism that is defined is rank selection.

`EvolutionConfig` The `Evolution` mechanism can be configured, and this class processes the configuration.

`Phenotype` This abstract phenotype class provides a factory method to produce from genotype.

`Genotype` This abstract genotype class provides a factory method to create a random genotype with the class-specified length, as well as mutation and crossover operations.

`FishermanVotingRules` The first defined phenotype is a voting mechanism that decides based on one of three modes. Either it decides to complain or not with a probability based on the distance from the agent's home cell or not, or it always complains its maximum number of allowed complaints, or it never complains.

`FishermanRulesGenotype` This genotype class is of length 3, and only provides a method to convert the bits into meaning.

`FishermanVotingNN` The second defined phenotype is a voting mechanism that decides based on a neural network. The network has four input nodes: distance, home conditions and conditions in the threatened cells, and it applies these inputs for each threatened cell. There is one hidden layer with six nodes, and one output node which it interprets as a complaint when the activation is more than 0.5.

`FishermanNNGenotype` This genotype class is of a length based on the nodes in `FishermanVotingNN`. It contains a method for converting the genotype to a list of numbers, representing the weights in the network.

LOG All the logging in the simulation is done through this module, by writing text files.

`LogMessage` A message in the log is recorded using this class, with fields for date, timestamp and text, as well as a to-string method.

`LogEvent` This abstract base class for events is empty.

`RoundStatistics` This log event captures a list of statistics every round: number of the round, average fisherman fitness, average number of fisherman complaints, number of unblocked cells and number of aquacultures.

`VoteFitnessRelation` This log event captures the relation between number of complaints and the fitness of a single agent.

`StatisticsLogger` The plots shown in the interface are produced by data gathered in the PHASES module. The same data is sent to the statistics logger, which writes the data in CSV format. The data format is specified in the PHASES description. The logger attempts to write continuously, but if it encounters a new kind of data point for the first time, it has to rewrite the file with a new heading. Therefore it is better if new data points are added as soon as possible, and new rounds for them are added later.

`Logger` The logging itself is done through methods defined in this class. A number of methods is defined for each log event; enough to capture the necessary data to add to each event. The recordings are saved by calling a method for each event that writes to a separate text file for each.

MARKET The market is static in this simulation. It has predefined prices for wild fish and farmed salmon.

`Market` The market class provides methods for getting the preconfigured fish prices.

MESSAGES Messages are the form of inter-agent communication found in the system. The messages contain a meta info object, which describes the sender, recipients and time of sending. When a message is received by an agent, it calls a method (a *reaction* method) in that agent that is custom for each message type. This way agents that expect to receive a certain kind of message need to implement that method.

`MetaInfo` The single-target message meta info class contains fields for sender (source), recipient (target) and time of sending (timestamp).

`BroadcastMetaInfo` Broadcasts are sent with this class as meta info, which instead of a single target contains a list of recipients.

`Message` The basic message only contains meta info.

`Reply` A reply to a message contains a reference to the original message as well as its own meta info.

`Inform` The general information message only contains a string, and evokes no reaction of the recipient.

`AquacultureSpawned` This message is sent whenever an aquaculture is created, and contains the location. It evokes no reaction.

`PlanHearing` This message contains the coastal plan, and is sent whenever the municipality initiates hearing. It evokes the *plan_hearing_notification* reaction in its recipient.

`VoteResponse` A vote response is a reply to the plan hearing message, and it contains a list of votes that indicates the agent's response to the hearing. It evokes the *vote* reaction in its recipient.

`VoteResponseInform` This message is a reply to a vote response, and its intent is to broadcast to other agents the contents of an agent's votes. It evokes the *vote_response_inform_notification* in its recipient.

NN The neural networks module contains the implementation in the neural networks, employed by fishermen when deciding how to vote.

`LabeledNeuralNetwork` This class implements the neural network itself. It takes parameters for a list of neurons in each layer (one input, one hidden and one output), and contains methods for updating the input, processing and retrieving the output.

`Neuron` Neurons are implemented through a class that stores the value and type of the neuron.

PHASES The simulation is split into phases or steps, and the sequence and actions of each step are implemented in the phases module. Each step is implemented as a class with a *do* method where the processing for that step takes place. They all inherit from a common interface. In each step, statistics may be gathered and sent to the step result through a data dictionary. This object is sent to both the interface (through the "generic dictionary data field" in `PhaseReport`) and the `Logger` instance. Statistical data, which is the data displayed by the GUI and recorded by the `StatisticsLogger` and written as CSV is specified in the "statistics" key in the data dictionary. Here a new dictionary is expected, with a mapping from *data point name* to the value. The data point name is a string that has to be the same every time a new data point is added to that plot. The value is specified as another dictionary, with three expected keys: "mode", "value" and "plot". The mode can be either "set" or "add". Normally, data points are added once each round and recorded per round, but if they are added more than once, the mode parameter describes how it is recorded. If it is "set", the old data is overwritten for that round, and if it is "add' the new number is added to the old data. For instance, the total amount of complaints in a round has the "add" mode because complaints add up in a round, but the number of planned aquaculture sites has the "set" mode because otherwise one could end up double- and triple-counting cells. The “value” field in the value dictionary holds a floating-point number. The “plot” field is a boolean field (flag) that is sent to the GUI to indicate whether the data should be plotted or not.

`Round` The sequence of events is implemented as a simple state machine, with a next-parameter for each state. The class for the whole round takes care of the stepping through the sequence with the *next* method.

`StepResult` The results of a step is stored in an object of this class, which contains fields for phase name, messages sent, cells changed, world map, other data, round number and votes cast.

`Step` The common base abstract class implements the state-switching functionality for simple steps, which only have a static next-parameter. Subclasses overload the *do* method with the desired actions. Access to objects in the simulation world is done through the info field, which contains a named “SimulationInfo” tuple object.

`DecisionStep` Decision steps are a subclass of `Step` which have a next table according to a predicate decided after the step has processed. It performs the same kind of *do* computation as `Step`, but expects a 2-tuple with both a `StepResult` and a value for looking up in the next table as result.

`CoastalPlanning` The coastal planning phase is a simple step where no cells can change, and the statistics collected is number of planned aquaculture sites.

`Hearing` The hearing phase is a simple step where no cells can change, and the statistics collected is the average number of complaints.

`GovernmentDecision` The government decision is a decision step that can have values according to the government decision: approve or review the plan. If it is approved, the next step is fishing 1, otherwise it is coastal planning again. Cells cannot change and no statistics are collected in this step.

`Fishing` The fishing phase are two simple steps, fishing 1 and 2, which happen before and after building, respectively. Cells can change and average fisherman capital is the statistic collected here.

`Building` The building phase is a simple step where cells can change when aquaculture is built on them, or is built nearby so the cell is blocked. The statistics total fish quantity and number of aquacultures are collected.

`Learning` The learning phase is a simple step where no cells can change, and the average fitness of fishermen is stored.

PLAN The plan module contains classes that simplify dealing with the coastal plan.

`CoastalPlan` This class subclasses the standard Python dictionary, and is used to map cells to plan entities.

`PlanEntity` Items in the coastal plan are plan entities, and these have a description. There are two entities implemented.

AQUACULTURE_SITE The aquaculture site represents a location that has been approved for industrial aquaculture development. Once a site has been approved and the plan has been finalized, an aquaculture company with a license may establish there without inference from other parties (like fishermen).

RESERVED_ZONE A "Reserved Zone" is an area that has been reserved from industrial development, and is open for fishing.

`Complaint` A complaint is an entity that gathers negative votes about a certain cell. It has an approved-flag for the government to set.

`Decision` The decision class is used as a name space for the overall plan decisions the government can make: approve, or review.

PRIORITY Agent's motivations are driven by their priorities, which are implemented as calculable entities. Priorities are calculated by picking attributes in a common object for influences.

`Influences` All influences that can have an impact on an agent's priorities are included in this class. It has field for the agent, all agents, community member agents, the market, fishermen, the world map and aquaculture agents.

`Priority` A single priority is implemented by giving a calculating function to this class.

SIMULATION The simulation module creates, configures and runs the simulation. It represents the top-level of the simulation process itself, and user interfaces communicate with this module directly.

`Simulation` The simulation class provides methods for setting up configuration, initializing all the objects in the simulation, and stepping through it, one phase at the time. The information used by the steps is stored in a named "SimulationInfo" tuple, which has the following fields:

map A reference to the `Map` instance from the WORLD module, representing the world.

cfg A reference to the complete configuration object.

directory A reference to the `Directory` object (from DIRECTORY) that handles inter-agent communication.

market A reference to the `Market` instance of the MARKET module that handles pricing.

agent_facotory A reference to the `AgentFactory` object (from ENTITIES) that is used to initialize agents.

aquaculture_spawner A reference to the `AquacultureSpawner` (from ENTITIES) object that chooses the cell aquacultures are spawned at, and creates them.

learning_mechanisms A dictionary mapping agent types to learning mechanisms.

logger A reference to the `Logger` instance from the LOG module that handles logging.

UTIL Utility library for miscellaneous functions. It provides *smart_line_sep*, which joins words by a separator, but limited by a line limit, and *update_map* which applies a set of updates to a `do.Map` grid.

VOTE The vote module contains classes that are concerned with the voting part of the hearing phase.

`Vote` The vote class is connected to a cell and has one of two values: approve, or disapprove (complaint).

`AlwaysApprove` This class is a simple implementation of the abstract interface `VotingDecisionMechanism` in the **decisions** package. It returns no complaints, so it always approves all plans with no interruptions.

WORLD The world module contains implementations for elements regarding the geography and contents of the world. This includes structure, and contents of map cells.

`Map` The map is the top-level entity, and it contains a structure which controls positioning. It defines many methods for accessing information stored in the structure, and for manipulating it.

`Slot` The contents of a map cell is stored in the slot class. It has fields for occupants, a spawning flag, a blocked flag, a land flag, an aquaculture flag, and a field for fish quantity. It also defines methods for manipulating and retrieving this information. There is also a *get_fishing_efficiency* method which is used by fishermen when working the slot.

`MapStructure` This interface provides method stubs for each function that needs to be implemented by a structure.

`AbstractStructure` The abstract structure class serves two purposes. Firstly it is an interface that defines method stubs for functions that need to be implemented by concrete structures. Secondly it is an abstract implementation of methods that are common to all structure implementations.

`FishingStructure` The two implemented structures are based on this common abstract implementation. It contains methods for initializing the slots and populating fishing spots.

`GridStructure` The first implemented structure is a grid structure. The grid structure has a given width and height, and its edges are final.

`TorusStructure` The second implemented structure is a torus. It functions as a grid, except for its edges which wrap around.

Appendix C

Extending and Altering the Program: Interfaces and Plug-ins

During configuration, the user can change classes that are used in the system. These classes are specified in appendix A and specified by a fully qualified class path name. The built-in classes that can be used all implement certain interfaces that describe which methods they require. These interfaces can be implemented by user-defined classes that wish to change the behavior of the program without changing the original code, and then referred to from the configuration file. The interfaces are documented here.

In addition to interfaces, more priorities can created by adding instances to the PRIORITY modules in the **FisherSimulation** package. Those priorities can then be referred to by their name in the configuration file.

C.1 Map Structure

The map structure is described by an interface and its implementation controls how the geography of the world acts. An implementation should create the world and populate it with `Slot` instances and distribute good fishing spots according to the given frequency.

The map structure interface is the `MapStructure` class in the WORLD module in the **FisherSimulation** package. Two implementations of the interface, `GridStructure` and `TorusStructure`, can also be found in the WORLD module.

The methods that requires implementation, with summary of expected functionality, arguments and expected return values, are detailed below. All the methods are ordinary functions that receive **self** as the first argument; a reference to the instance, unless otherwise is stated (like *create* which is a class method).

create This class method creates the map structure. It may either be overridden, or the implementing subclass can have a constructor that takes the same arguments as this function.

Parameters:

cfg The configuration object for the structure. This is an instance of the `Config` class from CONFIG in the **config** package.

good_spot_frequency A floating-point number between 0 and 1 representing the frequency of "good" fishing spots. A good fishing spot is twice as good as a regular one. The implementing class has to use this parameter when building the world.

Expected return value is an instance of the implementing class.

get_cell_distance Finds the distance in meters between two given cells.

Parameters:

a A `Slot` (from WORLD) instance representing the first cell.

b A `Slot` (from WORLD) instance representing the second cell.

Expected return value is a floating-point number representing the distance in meters.

get_distance Finds the distance in meters between two given positions.

Parameters:

pos_a A duple of two integers representing the coordinates of the first position.

pos_b Another duple of integers representing the second coordinates.

Expected return value is a floating-point number representing the distance in meters.

get_size Gets the size of the structure.

Expected return value is a duple of integers representing width and height (like (w, h)).

get_all_slots Gets all the slots that makes up the map.

Expected return value is a standard Python list of `Slot` instances (from WORLD).

get_aquaculture_blocking Gets all the cells that will or would be blocked by aquaculture expansion in the given cell. The aquaculture blocking radius is given by the `aquaculture blocking radius` field in the `global` configuration settings.

Parameters:

cell The `Slot` instance (from WORLD) where the aquaculture is being built.

Expected return value is a list of `Slot` instances (from WORLD).

get_aquaculture_damage Gets the cells damaged by aquaculture expansion in the given cell, and the amount of environmental damage sustained to them. The aquaculture damage proportion is given by the `aquaculture damage proportion` field in the `global` configuration settings. The aquaculture damage radius is given by the `aquaculture damage radius` field in the `global` configuration settings.

Parameters:

cell The `Slot` instance (from WORLD) where the aquaculture is being built.

Expected return value is a dictionary mapping cells to a floating-point number between 0 and 1, representing the amount of damage, where 1 is completely destroyed and 0 is untouched.

get_cell_position Gets the position of the given cell.

Parameters:

cell A `Slot` instance (from WORLD).

Expected return value is a duple of integers representing the coordinate position of the cell.

get_radius Gets all cells within the given radius of the given position.

Parameters:

r A number representing the radius in meters.
pos A duple of integers representing the coordinates of the position.

Expected return value is a list of all `Slot` (from WORLD) instances within the radius.

C.2 Learning Mechanism

Learning mechanisms represent learning functionality that can be defined for the different agent types. A learning mechanism needs to implement the `LearningMechanism` (from GA) interface. The interface defines a constructor that receives two argument which is the list of agents it applies to and the configuration, as well as a single method that is invoked whenever the agent learns. Both methods are also standard Python functions that receive a **self** reference.

__init__ This method creates the learning mechanism. It has implemented functionality that stores the passed agent list in a field called “agents”.

Parameters:

agents A list of agents that the mechanism applies to.
cfg The configuration object for the mechanism.

Expected return value is the instance.

learn This method is called once for each learning mechanism during the learning phase. It needs to be implemented by subclasses. It takes one argument, but no return value is expected.

Parameters:

fitnesses A dictionary mapping all agents to their fitnesses, which is a measure of priority satisfaction.

The only default implementation of a learning mechanism is `Evolution`, which itself uses instances of two new interfaces, `Phenotype` and `Genotype`.

C.2.1 Phenotype

Phenotypes represent the objects which are subject to evolution. They are defined by the interface `Phenotype`, and there are two implementations: `FishermanVotingNN` and `FishermanVotingRules`. The interface defines a constructor signature with a single line of code, and a classmethod for creating.

__init__ This constructor does not need to be overridden, but may be called by subclasses. It takes a single argument, which is the genotype and returns the instance.

Parameters:

genotype The genotype fulfills the `Genotype` interface.

Expected return value is the created instance.

from_genotype This method creates launches the constructor with its single argument. Returns the created instance.

C.2.2 Genotype

Genotypes are the description of their corresponding phenotype. They are defined by the interface `Genotype` which also provides some basic functionality. Subclasses *must* override the **length** class property with the length of the genome, so this implementation expects static genome lengths. Genomes are assumed to consist of a list of 0 and 1 bit characters. Implementing classes define functions and methods that are used by the phenotype.

__init__ The constructor takes a single argument which is the genome and saves it to a field "genome".

Parameters:

genome A list of strings "0" and "1".

Returns the instance.

random This class method creates a random instance of the genotype of a length equal to the length given by the class attribute "length". Returns the instance.

mutate Mutates the genome based on the given probabilities.

Parameters:

mutation_rate The probability of the genome being touched.

genome_mutation_rate If the genome is touched, the probability of any one of the bits being transformed.

__len__ Returns the class parameter length.

crossover This class method accepts two instances and a probability, and performs the crossover operation if it happens.

Parameters:

first The first instance to be crossed-over.

second The second instance.

crossover_rate The probability of the crossover operation taking place on this pair.

Returns a tuple of new instances created with the constructor based on the transformed genomes.

C.3 Voting Mechanism

Voting mechanisms are configurable and used by agents to decide whether they want to complain or not about aquaculture expansion. Voting mechanisms need to fulfill the `VotingDecisionMechanism` interface (from DECISIONS). There are three default implementations of the interface: `AlwaysApprove` from VOTE, `FishermanVotingNN` from GA as well as `FishermanVotingRules`, also from GA. The interface defines a factory method that is called for creating new instances, as well as a method that's called when agents vote. Voting mechanisms are individual for each agent. An implementation needs to implement both methods.

decide_votes This method decides whether to complain or not for each vote. As well as the described parameters, the standard self-reference "self" is included.

Parameters:

agent The agent deciding the vote.

coastal_plan A `plan.CoastalPlan` instance showing the plan and all its cells.

world_map A `world.Map` instance showing the current state of the world.

max_complaints An integer giving the maximum number of complaints allowed by a single agent in a single round.

Expected return value is a list of vote.Vote instances with APPROVE or DISAPPROVE values for cells with planned aquaculture in the plan. There is a maximum number of complaints allowed, so they have to be prioritized.

new Adds an instance of this mechanism to the given agent. Agents have a method *add_-voting_mechanism* for this use, which accepts a single argument that is a reference to the object implementing this interface.

Parameters:

agent An agent to add the mechanism to.

config The configuration object for the given agent,

world The `Map` instance giving the world.

C.4 Government Decision Mechanism

The government decision involves considering all the complaints and either approving or rejecting them, ultimately deciding if the coastal plan needs to be reviewed or not. Government decision mechanisms need to implement the `GovernmentDecision` interface from DECISIONS. There are two default implementations: `ApproveProbability` which approves each complaint with a configurable probability (a float between 0 and 1), as well as `ComplaintApproveMoreThanOne` which approves a complaint about a cell if more than one agent complained about it. Both are located in the DM module. The interface defines two methods as well as outlining a constructor. The constructor takes the configuration object which is the whole object for the agent type (`Government`).

round_reset Round reset is called when before a new round is started.

decision The decision function is called on the set of all complaints from a single hearing round.

Parameters:

complaints A dictionary of `world.Slot` instances mapped to `plan.Complaint` objects.

The expected return value is the dictionary of complaints, with approved flags set or not. For approval, the *approve* method can be used directly on complaints.

C.5 Coastal Planning Mechanism

Coastal planning is done by a planning mechanism. A planning mechanism needs to implement the `PlanningMechanism` interface in the DECISIONS module. There is a single default implementation of this interface, which by default plans everything to be aquaculture and removes items that have approved complaints, `EverythingAquaculture` from DM. The interface defines a single method for creating and reviewing a plan. The constructor accepts the configuration object for the agent type (`Municipality`).

create_plan The create plan method is called in each coastal planning phase.

Parameters:

world_map A `world.Map` instance representing the current state of the world.

coastplan (Default is None) The previous plan, a `plan.CoastalPlan` instance.

complaints (Default is None) A list of `plan.Complaint` objects to review the plan based on, it is None if there was no previous plan.

Expected return value is a `plan.CoastalPlan` instance. The coast plan class is a subclass of the standard Python dictionary, and it is initialized with a dictionary sent to the constructor. The dictionary should map world `Slot` instances to `PlanEntity` instances. There are two default plan entities: aquaculture site and reserved zone.

Appendix D

Software and Project Details

Table D.1 shows an overview of various project details related to the software.

Table D.1: Project details

Project code license	MIT
Project code hosted at	`https://github.com/ingfy/FisherSimulation`
External libraries	NumPy (2014)
	wxPython (2014)
Data and figures at	`https://github.com/ingfy/FisherSimulation`