Modelling collective dynamics in climate change agreements

Hugo da Silveira e Lorena Salgado Braz

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Francisco João Duarte Cordeiro Correia dos Santos

Prof. Fernando Pedro Pascoal dos Santos

Examination Committee

Chairperson: Prof. Miguel Nuno Dias Alves Pupo Correia

Supervisor: Prof. Francisco João Duarte Cordeiro Correia dos Santos

Members of the Committee: Prof. João Carlos Serrenho Dias Pereira

November 2017

Acknowledgements

Where do I start…? So many people made my work and thesis possible.

I want to thank my parents and family for making me study when I wanted to play and to always supporting my decisions, even when I was wrong.

My computer science professors, who were passionate about their fields of work and transmitted their passion to us, students. A very special thank you to my supervisors, professor Francisco Santos and Fernando Santos who always helped me with their boundless knowledge.

My friends and colleagues, who taught me to think outside the box, making me a more complete person.

Finally, to all that showed a genuine interest in this work, that supported me and made me continue and constantly improve my thesis.

Hugo Braz.

Resumo

Podemos ver a cooperação como sendo o objetivo principal de qualquer acordo de mudanças climáticas. No entanto, nem sempre é fácil alcançar este objetivo devido à tentação de não contribuir enquanto se tenta recolher os benefícios do trabalho dos outros. Usando um sistema multiagente, onde os agentes jogam um "jogo de bens públicos" e aprendem a jogar através de aprendizagem por reforço, simulamos como países podem não cair na Tragédia dos Comuns. Estamos interessados em investigar em que condições a cooperação entre agentes emerge e como esta se torna estável. Exploramos várias configurações possíveis, configurações de aprendizagem e o papel de vários tipos de fatores externos. Neste documento mostramos que em grupos mais pequenos a cooperação é mais fácil e que as contribuições para o bem comum aumentam.

Também analisamos o comportamento dos agentes quando estes são sujeitados a objetivos incertos e a erros. Verificamos que, ainda assim, a cooperação é sustentável, bastando que a variância de objetivos e a taxa de erros sejam baixas. Finalmente, demonstrámos que quando os agentes são criados com diferentes níveis de riqueza, as decisões de cooperar ou não passam a incidir principalmente no grupo de ricos. Admiravelmente, mesmo não comunicando entre si e recorrendo a um simples algoritmo de aprendizagem por reforço, os agentes, conseguem convergir na adoção do mesmo conjunto de ações, originando perfis de estratégias que se constituem justos e equitativos.

Palavras-chave: Sistema multiagente, Aprendizagem por reforço, alterações climáticas, tragédia dos comuns, cooperação, jogos de bens públicos.

Abstract

Cooperation can be seen as the ultimate goal in any climate change agreement. However, this is not always easy to achieve because there is a clear temptation to free-ride and acquire all the benefits of the deals, without any individual investment or effort. Using a multi-agent system where agents play a Public Goods Game and revise their strategies through reinforcement learning, we simulate how countries may cooperate and avoid falling into the Tragedy of the Commons. We are interested in investigating in which conditions cooperation between agents emerge and become stable in the long run. We will explore several types of group configurations, learning configurations and the role of several kinds of externalities. In this report, we show that cooperation emerges easily within smaller groups. We were also able to conclude that agents can cooperate when subject to an uncertain objective or to errors, provided that variance/errors are low. Finally, we demonstrate that, when agents are created with different levels of wealth, the decision of cooperating shifts to rich agents that have the power to achieve the goals almost by themselves. Remarkably, even though the agents do not have any communication system, they were able to converge to the same small set of actions, showing that agents can learn independently to play the same small set of strategies.

Keywords: multiagent system, reinforcement learning, climate change, tragedy of the commons, cooperation, public good games

Contents

List of Tables

List of Figures

Chapter 1. Introduction

One of the greatest challenges that can have a tremendous impact on the global economy, and therein affect the wellbeing of millions of persons around the world, is Climate Change. However, countries historically avoided finding a solution to this problem, because it would both require global cooperation that incur short-term economic losses. Not reaching the goal, however, will make the whole world face a risk of a huge human, ecological and economic losses [1]. This is a prime example of the Tragedy of the Commons and of a Social Dilemma. We face the Tragedy of the Commons when we have a shared-resource (in this case, the environment) and the users of this resource act only on their self-interest. This situation results in the depletion of the resource, that is contrary to the interest of all. The same can happen on several other shared-resources, like for example on intensive fishing, exploitation of water resources or overlogging in forests. To avoid the Tragedy of the Commons, large-scale cooperation is required between all countries and ideally, nobody should free-ride in an agreement. A free-rider is an entity that does not contribute to a solution of a problem but collects the benefits from other entities' contributions. Free riding is often considered one of the main risks to human cooperation, which can drive the population to the Tragedy of the Commons [2, 3].

The lack of cooperation on the issue of climate change is also explained by it being a Social Dilemma: an individual benefit from being selfish unless all other individuals also choose to be selfish.

This is a very real and current problem. Countries need to address the climate change problem soon rather than later, to avoid bigger economic costs in the future. In this thesis, we explore how simple agents can learn to avoid the Tragedy of the Commons and cooperate in the right situations given the correct incentives. This incentives study will be made by analysing the problem using Game Theory.

Game Theory is the study of social interactions between individuals and provides a very well-studied framework to analyse the incentives and predict outcomes for our problem, first-time introduced by *Von Neumann and Morgenstern* [4]. On section 2.1.1 we will discuss the details and practical examples of applications of the Game Theory.

Game Theory has been used to study many different games, one of them being the Ultimatum Game. The Ultimatum game is a two-person game, where one plays the role of the proposer, and the other plays the role of the responder. Before each play, the proposer is endowed with some resource and must propose a division with the responder. The job of the responder is to reject or accept the offer. This game is better explored later in section 2.2.8.

Game Theory can also be extended to support an N-person game instead of the traditional two-person game. This introduces a new complexity layer because we need to analyze the behaviour of a group that is playing against a player, or even against another group.

One possible generalization of the N-person game is to make what is called a Public Goods Game, where traditionally we have a group of *N* players that can contribute to a common pot with a percentage of their wealth. At the end of each game round, the amount in the pot is multiplied by a factor and distributed evenly among all the players, even with those that do not contribute to the pot. Players face a dilemma: they are better off individually if they do not contribute to the pot because they still would get a reward with zero effort. However, if all players chose to proceed in the same way, by not contributing to the common pot, then nobody would receive any reward.

Another interesting use of Game Theory is when we take the concepts of Charles Darwin's Evolution Theory and mix it traditional Game Theory. In Evolutionary Game Theory (EGT) each player has a fitness value (average reproductive success) and all players are competing to get the largest share possible of descendants [5]. This type of application of the game theory will be discussed in section 2.2.1.

For our project, we will create a model that is inspired by all these three types of games: The Ultimatum Game, the N-person games and the Public Goods Game. We will have a game where a group of players decide to contribute a percentage of their wealth to a public pot. If the common pot by the end of the round reaches at least a certain threshold, then agents can keep whatever amount of wealth they have left. If the agents are not able to reach the goal, then they face the risk of losing all their remaining wealth (with a certain probability). Then, instead of reproducing the most successful agents, we will use a reinforcement learning algorithm that will allow agents to learn when they play against other players. There are many different reinforcement learning algorithms available, but all have the same basic functionality: if an action received a positive (negative) feedback, then that action will have a bigger (lower) probability to be picked again in the future [6, 7].

We must consider that humans are not fully rational, nor do they will always play the best possible action on every game, so we will also introduce several mechanisms that try to close the gap between fully rational virtual players and humans. These different mechanisms and their implementation will be detailed later.

With this project, we expect to be able to gain new insights on human behaviour, when people face a social dilemma in which overall wellbeing is at stake given the self-interest propensity of individuals. We will study how can we avoid the Tragedy of the Commons, what are the conditions necessary to do so and how can we help groups of people work together to reach a common goal.

1.1 Outline

In chapter 0 we introduce our work and its scope. In chapter 0, we discuss some background theory with examples that have the goal to help to understand the basic concepts of Game Theory and Multi-Agent Systems. The related work will be presented in chapter 2.2. Thereafter, in chapter 0 we will discuss the model, system architecture and implementation of our software. The next four chapters will be dedicated to exhibiting our results: in chapter 0 we will discuss the impact of risk on cooperation, in chapter 0 we analyse the effect of wealth inequality on cooperation, on chapter 5.2 we discuss the results on cooperation of an uncertain (variable) goal on each game and finally, on chapter 0 we try to put together all previous results we extract a common conclusion of the players behaviour. On chapter Chapter 6 we will have a conclusion and on chapter 0 we have all the references used in this document.

Chapter 2. Background and Related Work

Before we study the related work on the next section, it is important to begin to understand some basic concepts that we will use in this document, using a few straightforward practical examples.
We start to explain the basis for our work (Game Theory) with a popular example, the Prisoner's Dilemma. We also explain what is an Agent and a multi-agent system and finally, we explain some theoretical concepts related to Game Theory.

2.1 Background Theory

2.1.1 Game Theory and Cooperation

Game Theory is a mathematical framework that studies the social interactions and strategic behaviours between agents [4] and gives us a systematic method to study the decision process of rational agents.
One of the most well-known game it is the 2-person Prisoner's dilemma.
The Prisoner's dilemma is a simple game where two agents can only make two possible actions: cooperate or defect [8]. The payoff of each agent is given in Table 1 where $T > R > P > S$.

Table 1. Payoff of the row player in the Prisoner's Dilemma

	Cooperate	Defect
Cooperate	R	S
Defect	T	P

Because of $T > R$ and $P > S$, there is a big incentive of an individual agent to defect because his payout will always be bigger, independently of the other agent's play.

Another type of game is the Public Goods Game, which corresponds to a multiplayer Prisoner's Dilemma. This game is played with more than two agents that decide how much of their wealth they want to contribute to a common pot. The sum of all endowments constituting the pot is then multiplied, and all agents (even if they did not contribute) receive an even share of this final value minus their contribution. Again we see that agents that do not contribute can collect larger rewards than agents that contribute, so there is a strong incentive to defect, that is, to not contribute at all. But how can we study how the (rational) agents will behave in this game? One easy way is to identify the game's **Nash Equilibrium(s)**.

The Nash Equilibrium concept is a crucial one in Game Theory. We say that we have a Nash Equilibrium in a game when no single player can get a bigger payoff by changing their strategy (play) if the other players keep their current strategies.

By replacing the variables on Table 1 with real values, we obtain the Table 2.

Table 2. A concrete example of a payoff matrix for two players of the Prisoner's Dilemma. The red value indicates the Nash Equilibrium strategy for this game.

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

In Table 2 we have the payoff matrix for two players (A and B) and two possible actions (cooperate and defect).

Suppose that both players are cooperating (and getting a payoff of 2), any of the two players can change their strategy to "defect" and get a payoff of 3 (and the other gets a payoff of 0). So, we can say that the strategy pair (Cooperate, Cooperate) is not a Nash Equilibrium. The same can be said regarding the strategy (Defect. Cooperate) and the (Cooperate, Defect as, in both of these cases the agents cooperating are better off if they defect. However, if both agents are defecting (Defect, Defect), no player can improve their payoff of 1, if just one of them start to cooperate; therefore (Defect, Defect) is considered a Nash Equilibrium. Interestingly, the Nash Equilibrium in the Prisoner's Dilemma is not a **Pareto Optimal** Strategy. Thus, the Prisoner's Dilemma is a so-called social dilemma [9]).

We can say that we have a state of **Pareto Efficiency** (also known as Pareto Optimality) when no player can increase their payoff without decreasing the payoff of any other player. Again, we use the example of the Prisoner's Dilemma to illustrate this concept.

Table 3. A concrete example of a payoff matrix for two players of the Prisoner's Dilemma. All green values indicate the Pareto Optimum strategies for this game.

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

With the example of Table 3 it is trivial to see that, when playing this game, the only non-Pareto efficient strategy is the (Defect, Defect) because there is one strategy that will improve both players reward. The strategy that improves both players reward is the (Cooperate, Cooperate) strategy. Notice that if the players choose to play the (Cooperate, Defect) or the (Defect, Cooperate), one of the players will get a better reward, but one of them will see their compensation decrease from 1 to 0.

Until this point, we have discussed the basics of Game Theory through a very specific example, namely how can we predict the behaviour of rational agents if facing such challenge. On a simple game like the Prisoner's dilemma, finding the Nash Equilibrium and the Pareto Efficiency strategies is trivial, and yet, linking those concepts with actual human behavior is often a challenge: first, there are more complex games and advanced players where studying and predicting the player's behaviour it is a much more challenging task. Second, people often use heuristics to adopt behaviors that deviate from what rational choice theory – resorting to equilibrium concepts such NE – predicts. For example, in the real world, humans like to explore new strategies, even if they are not the optimal ones and even if they try to make the best possible play sometimes they make errors.
We could set up a real-world experiment with humans, but this would be slow and expensive, or we could make a "virtual environment" with several virtual players (called "**Agents**") in a **Multi-Agent System** (virtual world where several agents interact).

2.1.2 Agents and Multi-Agent Systems

An agent is an autonomous entity, in our case a virtual entity, that makes an action, using actuators, according to given inputs (percepts) in its sensors and what the agents think it is the state of their environment and other internal state information. An agent may or may not have any artificial intelligence implemented; the simplest possible agent is a reactive agent. A reactive agent is one that reacts directly to some stimuli on one of its sensors.
A more sophisticated variant of an agent is a learning agent. The learning agents receive percepts in their sensors; they analyse the received information and runs it through the learning component that suggests an action to be realised in the environment. Ideally, the agents can receive immediate feedback from the environment and learn from the action that was done.
A multi-agent system is a system where several agents interact with each other in the same environment [10]. The interaction between agents may be direct, for example using communication, or indirect, by acting on the shared environment where the agents are inserted. Multi-agent system can be applied with different agendas and goals. Our agenda is often called a "descriptive" one by the literature because we are interested in how agents learn in the context of other learners [11].

With this work, we have the goal to apply a formal model and algorithms that agree with people's behaviours in laboratory experiments within the context of our problem [11]. There were several social experiments involving **threshold games**, cooperation and game theory. We will discuss some of these experiments and their most important results and conclusions.

A Threshold game is a game where a set of players make a contribution to a common pot and their final earnings are conditioned on achieving a given total sum. We will illustrate with a similar but simpler version of a well-known game Threshold game [3].

We can have three players playing a ten round game. Each of them starts with 40€, and on each round, he or she can decide to donate to the common pot anonymously 0€, 2€ or 4€. If by the end of the game the common pot is at least 60€ (this value is called *threshold*), every player can keep what they have for themselves, otherwise, there is a 50% chance (often called the *risk*) of them losing everything. Notice that the contributions already made are not recoverable, the players do not know who contributes with what, and there is no communication between agents [12].

To achieve the threshold in these games, the players have to be motivated to contribute, and typically this means that either the reward of reaching the threshold is significative or they face a penalty if they fail to reach the threshold.

2.2 Related Work

Much work has been done about how players (both humans and virtual agents) interact with each other when facing several types of threshold games. In this chapter, we will discuss previous work related to this subjects that are relevant to our work. We start by discussing how multi-agent systems are used in the research and commercial environments. We will also talk about the impact of risk on human decisions, the role of fairness, wealth inequality and local institutions on cooperation.

2.2.1 Evolutionary Game Theory

Evolutionary Game Theory mixes two familiar concepts: The Charles Darwin's Evolution Theory and the Game Theory.

In this context, we have a population of individuals, each one with a fitness value (average reproductive success) and all players are competing to get the largest share possible of descendants [5].

If we have some individuals with a trait X that are more successful than the individuals with the trait Y (have the biggest fitness) then natural selection will lead to an increase of individuals with the trait X within the population. If the individual's fitness with the feature X does not depend on the individuals with trait Y, then all population will gradually be entirely composed of these individuals, but "if the success of a trait is frequency-dependent, then its increase may lead to a composition of the population were other variants do better; this can be analysed by means of game theory" [5]. One practical example of this situation in nature is the prey/predator relationship. If the prey is abundant, then the population of predators will increase, reducing the prey population. When the prey population

is critically low, predators cannot feed themselves, and their population starts decreasing while the prey population increases until an equilibrium between both populations is met [5].
Like in Biology, we can also introduce mutations to our population. With each game, players will have a small probability from suffering from a mutation to their game strategy. This mutation mechanism can create new population dynamics to the game, by creating randomly new strategies that were not initially designed by anyone [5].
For our problem, Evolutionary Game Theory is impractical to implement, because there are too many actions that an individual can do (too many traits) that would require a significant population that would need to play all together at the same time, thus requiring a vast information exchange between individuals, consuming far too many resources. A more appropriate solution is to use a set of learning virtual agents in a multi-agent system.

2.2.2 Virtual Agents and Multi-Agent Systems

Learning in multi-agent systems is almost as old as the game theory itself. Has an example, in 1951 a learning algorithm was proposed for calculating equilibria in games, so the concept of virtual or fictitious players is nothing new [11]. Since then, hundreds of articles have been published on this topic. In this section, we will make a short overview of how multi-agents systems have been used and what they try to accomplish.
The Artificial Intelligence is a vast field of work, and as such, it is impossible to state here all the related work done with multi-agent systems [10, 13]. However, Multi-Agent systems literature was pointed to fit in one of five distinct agendas and goals [11]:

1. Computational
2. Descriptive
3. Normative
4. Prescriptive, cooperative
5. Prescriptive, non-cooperative

The first one views a multi-agent system as a mean to obtain solutions concepts (like the Nash Equilibrium) of a game [11]. This is useful if someone is only interested in the possible solutions to a game (considering only fully rational agents play it).
The goal of the descriptive agenda is to investigate formal models of learning that agree with people's behaviour and to study how agents learn when interacting with other agents [11]. In this work, we have this agenda, we are not only interested in the game solutions or at its Nash Equilibriums, but we also want to see how agents interact, how they can cooperate. To study these interactions, we will try to get a virtual environment where we can extrapolate behaviours to the real world and obtain relevant information about how we should behave to accomplish the same objectives as our agents.

The normative agenda focus on determining which sets of rules are in equilibrium with each other. For other words, we are interested in discovering which repeated-game strategies are in equilibrium. This is important because most repeat games strategies have a learning-rule associated [11].
The last two agendas have the goal to analyse how agents should learn. The prescriptive, cooperative agenda is used mainly in team games because it decentralises the decision process to all agents, hence the agents must cooperate to achieve a goal.
Finally, the Prescriptive, non-cooperative agenda looks to study how an agent should act to get a high reward, considering their competition (other agents) and the environment where they are inserted [11].
In many of these agendas, agents must interact with each other, to achieve an objective. There are many interaction networks and structures that can increase or decrease the probability of cooperation between agents (and humans). In the next chapter, we discuss several of these structures that are known to increase cooperation, their advantages and disadvantages.

2.2.3 Mechanisms for Cooperation

There are several interactions structures (often called mechanisms [14]) that affect how agents interact with each other and can increase the chance of the emergence of cooperation. These networks will be a little bit discussed below.
Humans cooperate, even in laboratory experiments where they play one-shot anonymous experiments (totally deprived of the mechanisms that help the emergence of cooperation) between people that never met before and could, because of that, develop a natural tendency to cooperate with each other. There are several possible explanations for this phenomenon.
All people have a cultural and social context and might have learned that many social interactions may be repeated in the future, so it is useful to cooperate now, in order to collect the benefits from others in the future, this is called the “social heuristic hypothesis” [15].
These and other human behaviours are necessary to comprehend in order us to construct the best computer model possible. All this can be summarised in a single sentence: “Humans are not entirely rational”.
There are five main interaction structures: direct reciprocity, indirect reciprocity, spatial selection, multilevel selection and kin selection [8].
However, the real world is extremely complex, and many of these networks may occur at the same time, even with some degree of variations. Having also in account that humans are not entirely rational, makes these networks, even more, harder to study
When there is a high probability of two agents interacting more than one time, then cooperation can emerge [16]. This can happen because these individuals can act based on previous interactions: “If you helped me in the past, I will help you now.”
Even, after introducing errors (where a play selected by an agent is switched without his control), cooperation continues to prevail [8, 17].

One of the most popular implemented strategies that simulate **direct reciprocity** is the Tit-for-tat strategy. It works by simply copying the previous action of the other agent (except in the first play, where there is no previous play, in that case, Tit-for-tat always chooses to cooperate). However in an n-person Prisoner dilemma (public good game) the key difference is that in a public good game, interactions cannot be targeted to a single individual [8], so if an agent does not contribute (or contribute much less) than a second agent, this agent cannot penalize the first agent directly. This problem can be solved by allowing every agent at the end of a game, to be able to adjust the payout of every other agent, so any agent that does not cooperate will see other agents to reduce his payout.

Another way to solve this problem in an n-person game with repeated interactions is to let the agents adopt an All-Or-None strategy [18]. This strategy consists in "cooperating only after a round of unanimous group behaviour (cooperation or defection)" [18] and have one big advantage that is to be robust against errors in the population. When following this strategy, if one or more agents defect (due to an error), in the next round, all agents will defect, and because we obtained a unanimous group behaviour, in the round after that, all agents will choose to return to cooperation [8, 18].

However, in order to direct reciprocity to work effectively, it requires one person to know what was the other's past behavior, for example, it doesn't work if a meet someone for the first time because I don't know if a can trust this person due to the lack of information available.

In **indirect reciprocity**, each agent has a "reputation", and in each interaction between two agents, a 3rd party (may be another individual not directly involved in the game) awards or removes reputation of an agent based on his action.

If an agent cooperates, his reputation increases, if defects then the reputation decreases. Of course, this system only works if all individuals have a communication system where they can obtain the reputation value of other agents.

This way, before playing, an agent can check the other player reputation and play accordingly. "My behaviour towards you depends on what you have done to me and others" [8].

Humans are particularly sensitive to reputation, infants as young as six months take into account other's actions when making social evaluations [8].

Experiments made with human subjects shows that subjects condition their behaviour based on the reputation and if they were cooperative or not in the past, thus making a good reputation a very valuable asset because it is worth paying the cost of being a cooperator to be able to get the benefits in the long-run and because of this, cooperation is sustained [8, 19, 20]. We can see examples of this system implementations in commercial today. Several e-commerce websites like Amazon or eBay, buyers rate sellers, maintaining large reputation systems between totally anonymous users using indirect reciprocity since they allow any potential buyer to check the seller reputation and decide if they want to engage in a trade based on this information [21].

Direct and indirect reciprocity often assume that everyone can interact with everyone, but in the real-world, we usually interact with only a handful of people. To model this, we can have an interaction network called **spatial selection** that assume that for every player, it only interacts (and influences) it's direct neighbours.

When populations are structured some of the previously mentioned dynamics change because agents can only interact with other agents near them. This means that when an agent makes an action it will impact all its neighbours which can be a problem. If for example, one of the many neighbours of an agent defects, the agent can not punish the defector individually, it must choose whether to make an action that will punish all its neighbours, or opt to ignore the single defector and continue to cooperate.
Besides that, the population structure also affects payouts and has the potential to create both cooperators and defectors clusters [8, 22]. In evolutionary games, agents copy the action of others with the highest payout and because cooperators clusters tend to have bigger payouts than clusters of defectors, cooperators tend to prevail.
Until now we have discussed only the interactions structures of individuals, however, in the real world, we have bigger structures involving teams, in which selection can also act. This is called **multilevel selection**.
As the name implies, multilevel selection operates when there is an extra layer of competition, that is, not only there is competition between agents, but also between groups of individuals. When there is a reward given for a winning team, instead of a winning individual, cooperation tends to increase substantially inside each cluster [8] because now there is a powerful incentive not to compete within a group, but with others.
There is ample evidence from real-world studies with humans that intergroup competition can drive intragroup cooperation [8, 23–26].
Another factor that can drive intragroup cooperation is when individuals have bigger interests that simply get a big payoff.
Kin selection is a process of cooperation where the individual's interest goes far beyond the payoff they receive by doing a certain action. Often, helping someone who is related, as a kin, is more important, even if it implies being altruistic and receiving a lower payoff.
This kind of selection has not been as extensively studied as the other types of cooperation because is deemed uninteresting by the scientific community because kin selection is already expected in human behaviour [8]. There is also evidence that humans cooperate with a much higher rate than expected with non-kin individuals, thus making this a uniquely human feature [8] showing that humans have a natural tendency to cooperate, even if gives them a lower payoff by choosing a non-optimal action, just to avoid creating conflicts with other people.

All of these interactions mechanisms are likely to be involved (or absent) from contemporary problems that our society faces and whose escape requires global cooperation.
One of the most preeminent challenges is climate change. In the real-world human interaction are incredibly complex and can vary significantly.
In the context of fighting climate change, we need to take into account these interactions to get better results, whether be in a local city meeting where only a few people get together in order to decrease their carbon footprint, or in a global meeting in the United Nations where all the world leaders are present and try to reach an agreement to reduce the global emissions of greenhouse gases.

However, reaching a consensus is not always easy, in the next chapter, we will analyse the dilemma that these participants face and how they get around them.

2.2.4 Risk impact on the Tragedy of the Commons in repetitive game

Regarding the climate change, there is a dilemma: emissions reductions will be required by all countries and entities, but this will have a short-term negative economic impact. The failure to meet these reduction quotas will make the whole world face a risk of a massive human, ecological and economic losses [1]. This is a classic example of the "Tragedy of the Commons".

Because climate change agreements exist to avoid the concretization of these risks, we shall make a small modification to the general public goods game until now discussed where some agents contribute to a public pot and then, if a minimum amount is reached, the pot is shared evenly to all agents. This method does not fit the climate change agreements model because countries contribute to a public pot to avoid being affected by climate-related disasters.

One common model used in collective-risk dilemmas is one where agents will still choose if they want to contribute or not to the public pot and they will always receive a reward minus their contribution (independently of the pot value or if the minimum amount was reached or not). However, if the pot failed to reach to minimum value threshold, there is a risk of all agents losing their wealth.

In the particular case of climate change, the efforts to prevent future losses depend on how likely these losses are perceived to be. This way, it is imperative to talk about risk.

In this context, we will speak of risk has the probability that agents lose all their wealth due to the failure of meeting the contribution threshold to avoid the tragedy of the commons. However, this social dilemma has some features that are different from other dilemmas [1]:

1. Agents must make decisions before they know the outcome of their action.
2. Contributions are lost, does not matter if the goal (contribution threshold) was reached or not.
3. The effective value of the public good (in this case, prevention of climate change) is unknown.
4. The remaining agent wealth is at stake with a certain probability (risk) if the contribution threshold is not met.

Manfred Milinski, et al (2008) [1] did an experiment with this dilemma with 180 university students. Thirty groups of six people were organised where each student would start a 10-round game with 40€, and in every round, each player would decide to invest in a public pot, 0€, 2€ or 4€ anonymously. If by the end of the ten rounds, the public pot had at least 120€ (average of 2€ contribution per player per round), all the players would take home their remaining money, otherwise, they would lose all the money they kept with a certain probability (the tested values were: 10%, 50%, and 90%) [1]. The more he or she invests, the higher probability the group successes the reach the target amount (120€), but the less money remains in his or her account. This social dilemma adds an extra trade-off: the more others invest, the less a player needs to contribute to the public pot [1].

In this experiment, as expected, the lower the risk, the bigger percentage of groups that failed to meet the 120€ minimum pot: half of all groups with a 90% risk did not reach the minimum pot, and 90% of all groups failed to reach the minimum where the risk was 50%. No group was capable of reaching the minimum threshold when risk was 10%.

However, in the groups subject to 90% risk, most of the groups that failed to reach the minimum pot value, only failed by a marginal value, but this value was much bigger in groups where risk was 10% and 50% (average sum of contributions at the end of the ten rounds was 92.2€ and 73.0€ respectively).

This experiment showed us that these kinds of the social dilemma are very sensitive to risk. Higher risk causes bigger pressure to act and contribute to the public good because of the danger of losing all endowment causes the expected value of the game to drop below the cost of contributing. Knowing that it is “cheaper” to contribute than to face the consequences of failing.

2.2.5 Risk impact on an evolutionary game

A different approach from the previous subsecrtion is modelling an evolutionary game – a game where individuals tend to copy others when they seem to be more successful, in this case, had gathered a bigger payoff [3].

In this scenario, a population of *Z* agents engages in an *N*-person dilemma where each agent can only make two possible actions, to cooperate or to defect.

A cooperator will donate a portion of their endowment, on the other hand, a defector donates nothing. If a group of size *N* does not contain at least *M* cooperators, all members will lose their remaining endowments with a certain probability (similar to the previous experiment).

In this framework, the presence of risk leads to emerging of cooperation. With finite risk *r,* both cooperators and defectors became disadvantageous when rare [3]. The fraction between cooperators and defectors is 0 when the risk in non-existing (no cooperators). However, when the *r* value increases, so do the number of cooperators [3].

This experiment also concluded that are other two major factors that enable cooperation: group size *N* and the value of *M*.

With smaller groups, it is easier to see collaboration between individuals than on larger groups. Intuitively is also easy to understand that a lower *M* barrier enables an easier emergence of a stable cooperation because each agent can contribute less on average to achieve a positive payout.

If we bring together these two results, we can conclude that, in the real-world, it can be beneficial make smaller and local agreements, where there is a high-risk evolved, in opposition to the global summits.

2.2.6 Risk curves and preferences

In 2016, *Kristin Hagel, et al* [27] made several simulations where they considered the risk as a variable that depends on contributions, instead of being a fixed constant in the game. Their simulations

consisted of a single game with *d* players, each one of them started with an endowment *E* and these players decided to contribute a certain portion *c* of their endowment to reduce the risk of losing all their endowment [27]. Unlike other studies, there is no threshold to be met; this means that the risk that the players face to lose everything only depends on the risk function and from their contributions.

They then observed how the payoff changed with each studied risk curve. All risk curves are decreasing functions with respect to the number of contributions. This means when players decided to contribute more, the risk of losing their endowment was guaranteed never to be larger than the risk they had before.

The authors also analysed for each curve where the Nash Equilibrium was situated. This is important to understand better what would perfectly rational players play.

However, in the real-world, there is a vast literature that shows that humans consider not only their expected payoff but also their risk preferences [4, 27–29]. Taking these risk preferences into account, the authors repeated the same scenario but this time with two new player groups, one more risk adverse (willing to contribute more to reduce the risk substantially) and other more risk-taking (contribute less but are subject to a bigger risk to lose their endowment). They concluded that risk-seeking individuals have essentially the same strategy as a risk-neutral individual. They are not willing to contribute much less than a risk neutral player, whereas a risk-avoiding individual can have a contributions pattern that is very different from neutral players because they tend to contribute a much bigger portion of their endowment [27].

A risk is usually the probability of an external event in the real-world, that is very hard to control [27], however in certain conditions, we can have some influence on the risk management by our actions. One scenario where risk can be successfully managed is the climate change, for example, where a group of people must work together to minimise the risk of future consequences.

In this paper, the authors showed indeed that the players were indeed willing to cooperate between themselves and contributed to reduce the risk of losing all their endowment. This showed us that even without any threshold mechanism, like the ones referred in subsections 2.2.5 and 2.2.6 players still cooperate towards a common goal.

2.2.7 The importance of Institutions

Until now, we analysed previous experiments without any institution. Beyond cooperators and defectors that already were discussed and defined in the preceding sections, we will also have punishers. Punishers are agents that not only cooperate by giving a portion of their endowment to a common pot (like any regular cooperator) but also contribute by paying a “punishment tax” to an institution that with enough funding will punish defectors by a certain amount [30].

Two types of institutions may exist, global and locals. GIobal institutions are institutions that supervise all interactions between all groups and all punishers in the population contribute to them; local institutions are institutions that exist only in a particular interaction and are only supported by punishers present in that interaction [30].

A paper published in 2013 by *V. Vasconcelos, et al.* [30] using both local and global institutions in small groups interactions, concluded that local institutions are easier to emerge (there is a stronger incentive to punish defectors where punishers are directly involved) and therefore more effective to stop free-riders (defectors) in interactions. Local institutions have an added benefit; they enhanced cooperation when is most needed: with low risk and high requirements to avoid a collective disaster [30].

Global institutions were found to provide, at best, a marginal improvement when compared to not having any institution in place at all.

With these results, we can see further evidence that may be counterproductive to make a global summit with a global institution (like the United Nations) to try to reach an agreement. To obtain a bigger percentage of cooperators, it is preferable to have a small group size in negotiations and to have supervision by local institutions capable of punishing defectors.

2.2.8 Fairness and learning

Fairness plays a significant part in human behaviour; it sometimes forces humans to make non-optimal decisions and creates unexpected outcomes [21].

An effortless way to observe this phenomenon is by watching people play the Ultimatum Game. The Ultimatum game can be a two-person game, where one plays the role of the proposer, and the other plays the role of the responder. Before each play, the proposer is endowed with the same resource and must propose a division with the responder. The job of the responder is to reject or accept the offer.

If the proposal is rejected, then none of the players earn anything, otherwise, if the proposal is accepted, then each player will split the resource per the proposal.

In this context, only an equal resource division is considered to be fair.

If humans were totally rational, the responder would accept any offer, because any offer, does not matter how small it is, is better than ending up with nothing, as a consequence the proposer would not fear to have his or her proposal rejected, so he would offer the minimum amount possible [21].

However, this is not what happens when we see people playing this game; humans tend to reject low offers and by doing that, manage to get better and even fair offers [21]. Regular equilibrium analysis neglects this typical human behaviour, so it is necessary to use other techniques. A paper published by *F. P. Santos et al (2016)* [21] tried to do exactly that, to study how humans would behave in a Multiplayer Ultimatum Game.

The authors of this study decided to use the Multiplayer Ultimatum game, where instead of a single responder, there is a group and collectively decide whatever to accept or reject a proposal by the proposer.

When this group of $N-1$ responders receives a proposal, each of the players decides to accept or reject the offer. If the number of individuals is equal or greater than a threshold M, then the proposal is accepted; otherwise it is rejected.

The payoff distribution also suffered some changes. There were two different formulas studied, one where, if a proposal were accepted, all responders would get an equal amount of payoff and other where only the individuals who accepted the proposal would get a payoff. If the offer were rejected, none of the responder or the proposal would get any payoff, like in the regular two-person ultimatum game.

The groups of N players were picked randomly at the start of each round.

How to model the human behaviour previously discussed? One possible solution is to let the agents learn by themselves by playing thousands of games against each other. This learning process can be achieved by using the Roth-Erev reinforcement learning algorithm, which has been successfully used in modelling human learning in well-known interaction paradigms such as this one [31].

This algorithm works in an intuitive way. When an individual *k* chooses an action to play, it will reinforce the use of this strategy depending on the payoff gathered in that play. Higher gains in a game will increase the probability of this action being picked later. Over time, all other strategies and their past payoffs will be forgotten, the speed of which previous plays are forgotten is determined by the forgetting rate constant.

Other features that can be implemented with this algorithm like for example, an error rate or the local experimentation rate. The error rate is exploration mechanism and is defined as a probability of an agent to pick a random strategy, ignoring his experience. Local experimentation is another exploration mechanism that allows agents to play strategies like the ones already played. These features capture some essential properties of human learning, the Law of Effect [32] and the Power Law of Practice [33].

The Law of Effect simply states that humans tend to reinforce the use of previously used successful actions and the Power Law of Practice states that learning curve for a given task in a human is initially steep by over time becomes flat [21].

This paper concludes that this learning process indeed makes the proposals fairer. For example, for a group size of eight (one proposal and seven responders), found that for a low value of *M* (*M* = 1), the responders only got, on average, a reward of 0.2 (maximum possible reward is 1), oppositely, with an *M* of 7 (all agents must agree to accept the offer) the reward obtained was much bigger, on average, the responders received 0.7, this is surprising considering that the game equilibrium analysis predicted that the responders would get close to zero.

2.2.9 Wealth Inequality

Until now we only analysed studies where all agents had similar wealth, and they needed to share a portion of it to the common good. However, in the real-world exist a huge gap in countries or individuals' wealth.

Previous failures to reach a consensus in several climate summits have been attributed, in part, to conflicting policies between rich and poor [34].

In 2014, *Vitor V. Vasconcelos et al* [34] published a paper where they analyse the impact of wealth inequality in a repeated threshold public good game using computer simulations.

All games featured groups with a fixed size (six agents each) where all individuals belonged to one of two possible wealth classes: “rich” or “poor”.

The population was constituted by 80% poor and 20% rich, values that were chosen based on the idea that 20% of the world’s wealthier countries produce approximately the same gross domestic product as the remaining 80% [34].

One of the goals of this study was to see the impact of homophily (tendency of agents to copy only the actions of the agents with the same wealth status) and concluded that whenever the rich and the poor are evenly influenced by anyone else (no homophily), cooperation and group achievement are greater for any risk value experimented. However, when in the presence of homophily, the chances of success are generally below the changes without any wealth inequality mechanism.

In summary, homophily generally impels the rich to compensate for the poor [34], but given that contributions of the poor are vital to solving the climate change problem, it is critical that homophylic behaviour is avoided at all costs. Of course, that when the poor contributions are widespread, the rich will refrain from contributing.

This paper also predicts that the changes of cooperation appearing and being sustainable are bigger when: (i) high perception of risk, (ii) negotiations are between small groups with short-term goals, (iii) agents are influenced by the most successful peers, whether they are rich or poor. Many of these conclusions are the same ones that the ones previously discussed.

Wealth inequality adds a very interesting layer to a simulation because in the real-world this type of differences between countries can have a significant impact on the negotiations of climate deals.

Chapter 3. Model and System Architecture

In order to comprehend human behaviour and the reasons behind their actions and decision processes we have two options: do real experiments with humans (see section 2.2.4) or simulate human behaviour by modelling our problem using evolutionary game theory (section 2.2.1). Both approaches have problems. By doing experiments with real humans we would face the following problems:

1. We would need to provide them with rewards and penalties (most times it is used monetary incentives – making large-scale experiments very expensive)
2. Most times humans are not entirely rational, making harder to understand some behaviours through a limited amount of data.
3. The experiences would need simple rules, to allow our players to learn and understand the game within a short period of time.
4. Experiences take much more time when compared to a computer simulation, primarily when using big groups of players.

On the other hand, Evolutionary Game Theory (EGT) would solve some of these problems. However, when applied to the timescale of learning in social systems, EGT is particularly useful to model situations in which fit behaviors spread faster, for instance through observation and imitation. This said, it can be hard to argue that, in some complex games, social learning through imitation is, *de facto*, the behavioral update process of humans. In these situations, all agents would need to share information about their internal state, like for example, obtained rewards, strategy played and the whole decision process that got them to contribute what they contributed (this could either a very simple or a very complex and computationally expensive process).

To get around this problem, we came to the natural conclusion to let the agents learn by themselves what they should and shouldn't do when faced a specific problem using Artificial Intelligence, more specifically Reinforcement Learning. By using this method, we only need to model our challenge

(rewards, penalties, agent's wealth, etc.), to define a reward function to the agents (this will define what will agents try to optimize) and just let the agents learn by experience during several hundreds of games. Playing such big quantity of games is almost impossible to be feasible in a reasonable amount of time with real people.

In this section, we start by making a general description of the project and then we will discuss how our system was designed and finally we talk about the most relevant implementation details of this work.

3.1 Reinforcement Learning

From the several available reinforcement learning algorithms, we chose the Roth-Erev Reinforcement Learning Algorithm for several reasons:

1. Was successful in modelling human learning in well-known interaction paradigms [31].
2. Captures two important human learning properties: The Law of Effect [32] and Power Law of Practice [33]. The Law of Effects states that humans tend to reinforce the use of previously used successful actions and the Power Law of Practice states that the learning curve for a given task in a human is initially steep and over time becomes flat [13].
3. Works in an intuitive way.
4. Easy to implement.

The Roth-Erev algorithm basically says that, when an individual *k* chooses an action to play, it will reinforce or not the use of this strategy depending on the payoff gathered in that play. If the reward is positive, then this action will tend to be repeated in the future, otherwise (reward is negative) the action will be less played. In the case that the reward is zero, the play will be neither reinforced neither discouraged. The sum of these rewards for each action is kept in a propensity vector. This propensity vector must have the same size as the number of available actions that are possible to be made by each agent. More formally, for each agent *k,* we have a propensity vector *Q* of size *V*, for any given timestamp *t*. The agent on timestamp *t + 1* will make an action *i* and receive a payoff *P* for that action.

$$Q_{ki}(t+1) = \begin{cases} Q_{ki}(t) + P_i, & k\ played\ i \\ Q_{ki}(t), & otherwise \end{cases} \quad (1)$$

The above formula 1 states that the unique value of the propensity vector that will be updated corresponds to the position of the played action.

The values contained in this vector are not probabilities, but they must be transformed into one because the propensity vector represents the score of each action and actions where bigger scores must have a bigger probability of being played in a game. This can be done by normalizing the propensity vector [21]. The chance that individual *k* picks the strategy *i* at time *t* is given by:

$$\rho_{ki}(t) = \frac{Q_{ki}(t)}{\sum_n Q_{kn}(t)} \quad (2)$$

This formula has a problem: the sum of values in the propensity vector must be bigger than zero. For our purpose, we considered that if the sum is indeed zero, then the action to be played will be the first one on the propensity vector.

The two formulas above (1 and 2) are the basis of the Roth-Erev algorithm: the first one tells us how each played action change our knowledge (represented by a propensity vector) and the second formula how we can transform that knowledge into probabilities for future actions. However, one problem remains: how should we initialize the propensity vector? There are many possible solutions, but for our project, we only considered two main ones:

1. Random Values: initialize the vector with random values in the range of [0, 1[using a uniform distribution.
2. Single Value Bias: initialize the whole vector with zeros (or with very low value), except in one single position that is initialized with a value bigger than zero, forcing the agent to start their game by playing that strategy.

The Roth-Erev algorithm guarantees that positive (negative) feedback will make the probability of repeating the same action higher (lower). However, until now we are ignoring several important properties of the human behaviour.

Simulating perfectly the human brain and behaviour is a whole research field by itself and out of the scope of this project however, we can integrate some important characteristics:

1. Errors. Humans sometimes make mistakes or like to try new unknown strategies [35].
2. Forgetting. With time, less practised strategies will slowly be forgotten.
3. Local Exploration. When receiving a positive (negative) reward for playing a strategy, will make similar strategies more (less) likely to be played.

All three characteristics above help us to obtain more realistic results and solve some technical problems.

Errors are a fact of life, humans make mistakes and sometimes they know what is the best action to a situation, but implement a completely different one. Sometimes, from these mistakes, new and better strategies can be found. Implementing an “error mechanism” is also extremely simple: before any given agent makes a play, they have a small change (for example, 1%) to ignore their propensity vector and to make a random valid action.

We will also introduce two more learning features, the forgetting rate (γ, $0 \leq \gamma \leq 1$) and the local experimentation rate (ϵ, $0 \leq \epsilon \leq 1$) [31].

The forgetting rate is the rate that any given position in the propensity vector decays at each time step. This feature is important because avoids the values of the propensity vector to grow without a bound [21].

The local experimentation states that if a given played strategy is successful (positive reward), then similar strategies will also get a (smaller) reward, the same is applied if a play is unsuccessful. This introduces an interesting feature, a spontaneous trial of novel strategies (yet close to the ones tried before). This is a characteristic that is very human-like and it improves the performance of the agents [21, 36].

The complete update formula for the propensity vector is:

$$Q_{ki}(t+1) = \begin{cases} Q_{ki}(t)(1-\gamma) + P_i(1-\epsilon), & k\ played\ i \\ Q_{ki}(t)(1-\gamma) + P_i\frac{\epsilon}{4}, & k\ played\ i \pm 1 \\ Q_{ki}(t)(1-\gamma), & otherwise \end{cases} \quad (3)$$

This will be the final formula that will be used in this work. We can see that the previous values of the propensity vector are multiplied by $(1-\gamma)$ that is our forgetting rate, making the previous values less relevant for future decisions. If $\gamma = 0$, then we have no forgetting, and our agents will consider all past interactions to choose a strategy, in the other hand, if we have $\gamma = 1$, all past games are ignored and the decision is only based on the immediate past. However, we allow any γ value that is in the range of [0, 1]. Forgetting rate introduces a technical feature, as well: it prevents the values Q_{ki} to grow without bound as, for later stages of the simulation, the value forgotten has the same scale of the ones being added.

The local exploration factor distributes the reward of the played action to its neighbours. Can also be any value in the range of [0, 1]. When $\epsilon = 0$, then the only position on the propensity vector that receives the reward is the position that represents the action played; when $\epsilon = 1$, is the opposite, the reward is exclusively attributed to the neighbors of the strategy played in the propensity vector.

For simplicity sake, we may refer from now on the exploration factor as "ϵ" and the forgetting rate as "γ".

The only element from our propensity vector update function that we did not yet fully analyze is the payoff function. In Artificial Intelligence, the payoff function is one of the most important elements that must be defined and have a very significant impact on the algorithm performance. This payoff function must capture all the incentives and penalties that the agents are subject to. The wrong set of incentives will create abnormal behaviours and misaligned strategies with the goals of our game.

3.2 Payoff Function

Our payoff function is a function that given 1) an agent's action *a*, 2) the collection of actions of the other players and 3) our game threshold (*M*), outputs a float number that represents the reward attributed to the agent that played *a*. This function is extremely important to our simulation because it has a significant impact on the values of the propensity vector of the agents. We must develop a

function that considers all incentives and penalties of the agents, but it must also be balanced. An unbalanced payoff function may produce agents that display a strong risk-taking behaviour (if we give too little weight to the penalties of failing to meet our game goals), or the opposite – creates agents that are too conservative and will always meet our goals that are unrealistic in the real-world. This function it also needs to be fast to compute its output because the payoff function will be called in the worst-case *N* * *T* times on each simulation run[1].

We are going to base our payoff function on similar previous work on this field [3, 21] and will attribute a fixed positive reward and then provide penalties according to the agent's behaviour. Remembering the previous section: we have a set of agents that start with a certain amount of wealth and then choose to contribute a percentage of it to a common good. If the total amount in the pot, is equal or bigger than the threshold *M* than the agents can keep the remaining amount of their wealth, otherwise they risk losing everything with a given probability.
Fortunately, our game scenario is easy to translate into a payout function because it doesn't have many different variables that directly affect the payout. It is only affected by the contribution of the agent, the risk and the threshold. The payoff for an agent *i* is given by:

$$P_i = \begin{cases} b - c_i, & if \ \sum_{k=0}^{N} c_k \geq M \\ -c_i + (1-r)b, & otherwise \end{cases} \quad (4)$$

Where *r* ($0 \leq r \leq 1$) stands for the risk that a group of agents looses everything when not achieving the threshold, *b* is the fixed positive reward that an agent always receives and c_i is the contribution made by agent *i*.
This formula 4 is very fast to be computed because it only uses simple arithmetic operations. However, this apparently simple function can create some incredibly complex games, for example, we can manipulate the risk and threshold values directly or create scenarios that will impact indirectly these values, like by introducing an uncertain threshold, that changes every game.

In the previous sections, we discussed the theoretical details behind our simulation system. On the next section, we will introduce the practical side of our system and discuss its implementation details.

3.3 System Architecture and Implementation

Our project will perform a lot of computations on each game, so we must take this into consideration. As a result, we will implement our system in C++11 with the time-tested GSL (Gnu Scientific Library).
Because our game definition involves a certain amount of randomness, we will perform several runs/executions for the same scenario and do a statistical analysis of the data collected.

[1] N = group size; T = maximum time-steps on the simulation

Let us define a run *R* with *T* timesteps with a population of *Z* agents. Each agent contains a propensity vector *Qk(t)* of size *V* where each entry corresponds to a different possible strategy that can be played in a game.

In every match, we select a random subset of *N* different agents to play without any preference or order. Each of these selected agents must choose a strategy to play per his propensity vector.

We considered only discrete strategies, where every strategy is rounded to the nearest multiple of *1/ V* because otherwise, we would have to deal with an infinite number of possible actions that would not be computationally possible to handle [21].

All plays will be collected, and if the sum of the contributions (a portion of their endowment) is bigger or equal than the threshold *M*, every agent will receive a reward. Otherwise, no reward is given. After every game, each of the *N* participants (players) will have their respective vector updated with the payoff they gathered (based on their contribution and if *M* was reached or not). Positive (negative) payoffs will increase (decrease) the likely probability of the strategy played to be chosen again in the future.

We expect to see all the agents converge to a small set of similar strategies a few thousand timesteps [3, 21].

Another goal of this project is to make its code base as reusable as possible to allow us to extend or create new functionalities with ease. Ideally, our code will also be abstract enough to be used on completely different types of simulations and reinforcement learning algorithms.

We decided to design our code as a library that has an interface that encapsulates our system's behaviour that can be invoked by a client.

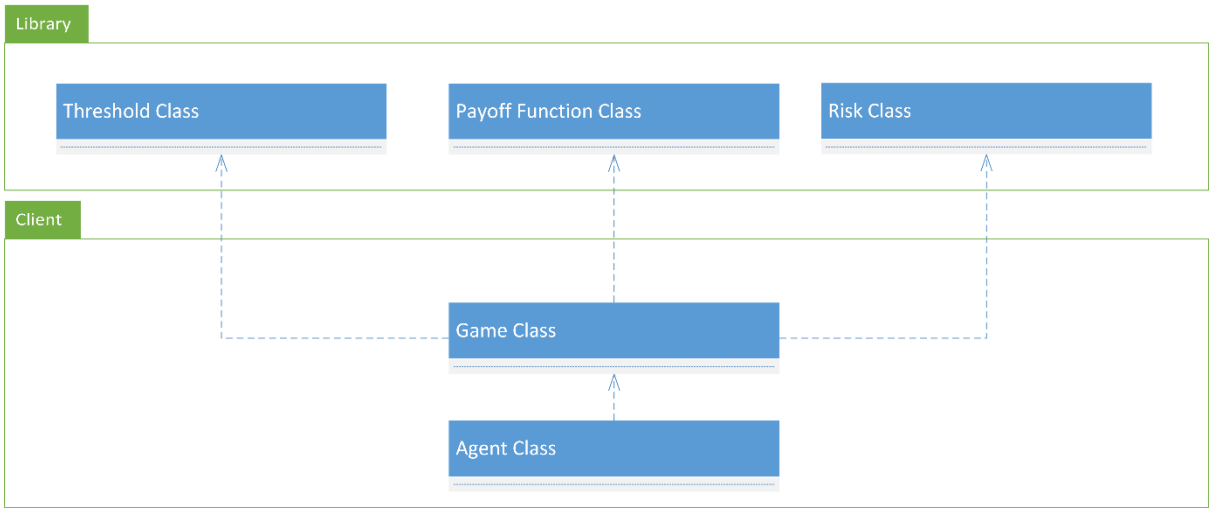


Fig. 1 Representation of the classes that our library implements and the classes that must be implemented by the client. This UML diagram is simplified, it does not contain the classes attributes or methods.

Our Library implements the Threshold Class, that represents our objective that we want the agents to accomplish; the Payoff Function that defines the rewards that will be attributed to the agents according to their actions; and the Risk Class that implements several risk strategies that can be applied to the

game. On the client side, it must have a class that specifies the Agent and their respective learning algorithm and the Game class that defines how the simulation will be done. Finally, it is also necessary on the client side a class that coordinates the other two client classes – this class is not represented in the figure above because it does not interact directly with the library. Optionally, the client can write auxiliary classes to output metrics related to the simulations. In our simulations, we implemented these output classes that wrote to a file, statistics about the contribution level, percentage of games won, contribution variance and the learning curve of our players. This output file can be processed later to obtain interesting and useful information about the games.

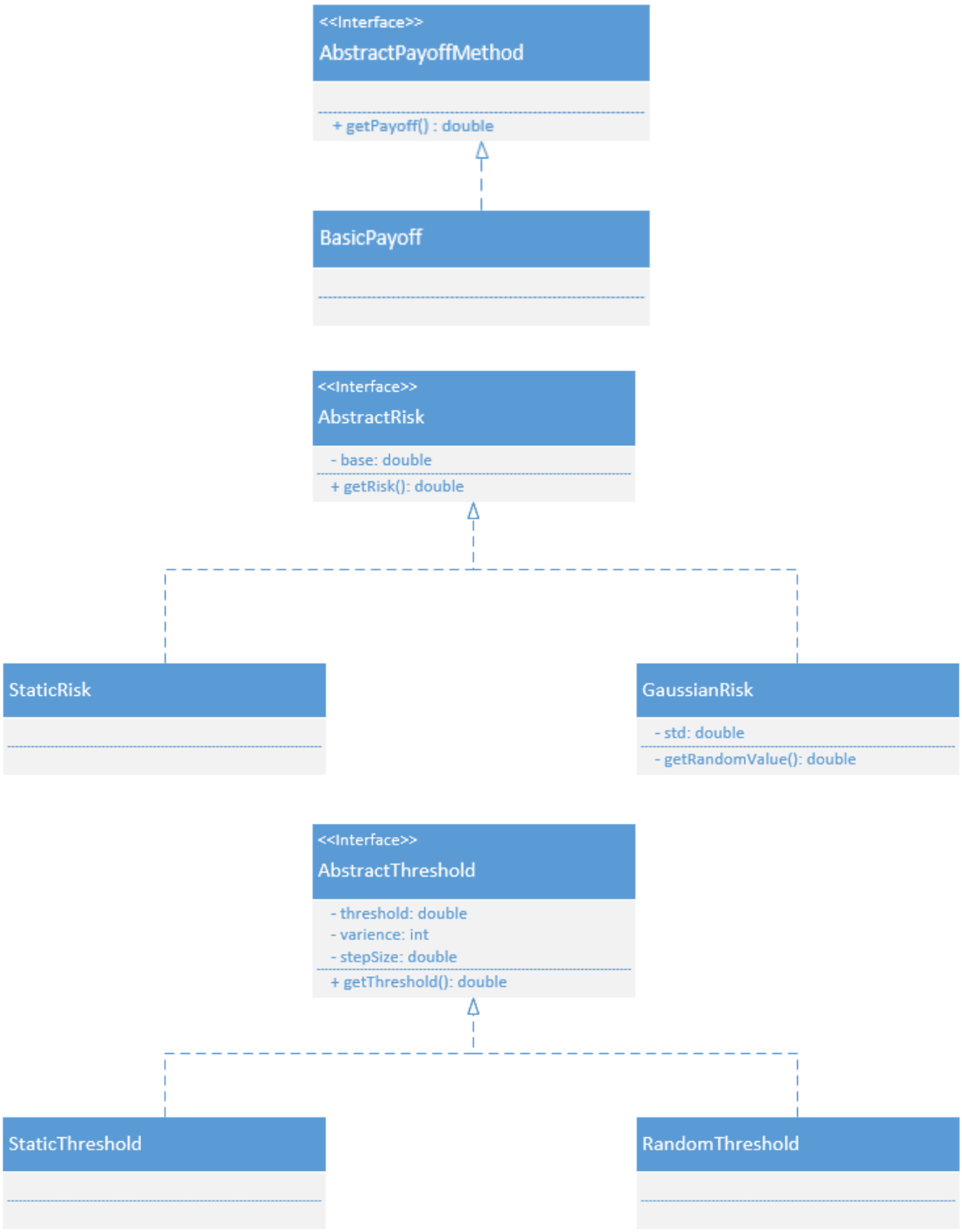


Fig. 2 UML diagrams of the classes present in the library. In the upper pane, we have the *AbtsracPayoffMethod* interface that determines what is the payoff that agents will receive. IN the middle pane, we have the *AbtractRisk* interface that represents the risk value that the agents face and finally in the bottom pane the *AbstractThreshold* determines what is the goal that the agents are trying to achieve. Note: only the most relevant attributes and methods are represented here.

In our library, we have three interfaces that can be used by the client to implement their Games and Agent classes. The client can also extend library's classes if the ones implemented don't fit the intended propose.

The *BasicPayoff* class implements the payoff function that is used in our project, explained in the previous section.

The *StaticRisk* class implements the most basic risk that we can make - a fixed value configured in the beginning of the simulation. The *GaussianRisk* is a class that calculates a random value for risk in each game using a Gaussian (Normal) distribution. This can be useful for simulating agents that are subject to an uncertain amount of risk in each interaction that they have.

StaticThreshold class simply returns a fixed threshold that was also configured at the start of the simulation. *RandomThreshold* returns a random threshold within a range on each game. This range is calculates using a methodology that is explained later in section 5.2.

Our library only has implemented some simple algorithms; however, it can be extended to much more complex classes and introduce new and interesting dynamics to simulations. We consider that this library can be used as a "building block" to be used by the client to implement their own feature-rich simulation system.

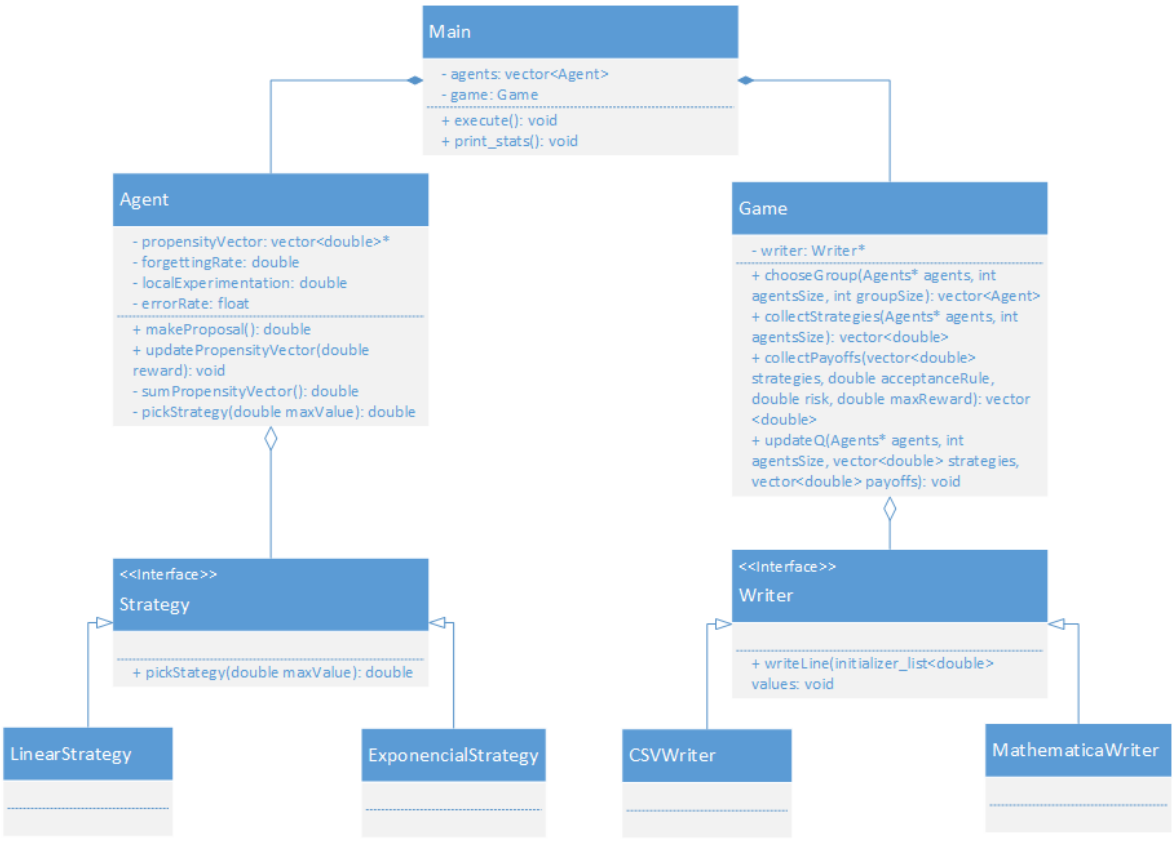


Fig. 3 Client-side implemented classes and their respective main attributes and methods.

This structure was chosen because it allows the code to be loosely coupled, that is, each of the code modules does not have many dependencies (if any) with other modules. Thus, all classes and methods can be easily modified or extended to allow new behaviours to experiment.
Our Main class is the class that binds all the different elements together. Its major goal is to define the game loop.

```
vector<Agent> agents;
Game game = new Game();
for r = 0 to R, number of runs do
   agents.clear();
   for z = 0 to Z, number of agents do
      agents.add(new Agent());
   for t = 0 to T, number of time-steps do
      for z = 0 to Z, number of agents do
         vector<Agent> group = game.chooseGroup(agents, agents.size(), N);
         vector<Strategy> strategies = game.collectStrategies(group, N);
         vector<Payoff> payoffs = game.collectPayoffs(strategies, M, risk, b);
         game.updateQ(group, N, strategies, payoffs);
```

Algorithm 1. Pseudo-code of the main game's basic loop method in the Main class.

In this class, we start by initializing an empty array of Agents and by creating an object of the Game class.
At the beginning of each run *r*, *Z* new agents are created (with new randomized propensity vectors, or, as detailed below, with a single value different than 0) and added to the agent's vector.
Then, for each time-step t, are played *Z* games with different randomly chosen groups.
After selecting a group, their contributions are collected, and a payoff is awarded to each one of them, and finally, each of the agents will get their propensity vector updated based on the payoffs received. The value of b is 1 (one) by default; that is the maximum reward that an agent can get.

How we can see in the Algorithm 1 we call several methods from the Game class.
The *chooseGroup* method receives as arguments the list of all *Z* agents, its size and the size of the group we want to pick (*N).* This method randomly chooses a set of *N* different agents from the *Z* available without any preference or bias. The *collectStrategies* methods get the previously chosen group of *N* players (and the group size) and output each of the player's strategy. Here we decided that agents should output their played strategies, but it could technically be a private attribute of the agents. However, when we have direct access to the strategies, more easily test the method and above all, compute performance metrics, like the evolution of the contributions with time.

The next step is to compute the payoffs for each agent. This is done in the *collectPayoffs* method that needs to get as input the played strategies, the threshold (or game goal) *M,* the risk value and the max reward value *b*. The output an array of rewards for each player calculated with the already discussed payoff function.

Finally, the *updateQ* method receives as arguments the player's group, its size, the played strategies and the rewards vector and updates the propensity vector of the players. Only the agent's propensity vectors that played in this round are updated.

Optionally, we can add several features to this simple code. We implemented two improvements: an early stop condition in the inner "for" loop. This would stop the simulation when we detect that all agents converged (with a small degree of tolerance) to the same strategy; and several monitors that collected data on the behaviour of the agents. For example, statistical data regarding the agent's contributions and rewards. The early stop condition checked every 250 timesteps if the average contributions of the agents changed (+/- 0.025%) when compared with the previous 250 timesteps. If this happens, we stopped the simulation, otherwise, the simulation continued until the end. Because we selected a very small tolerance, we guarantee that the agents indeed converged. This feature was extensively tested to ensure that the simulations with and without the early stop would obtain the same results. This method is, unfortunately, very slow because it requires calculating the average values in the propensity vector of all *Z* agents in the game. We solved this problem by only calling this method every 250 timesteps. We found this value provided a good granularity and a small enough performance penalty.

The Game class also have an implementation of the Writer interface. This is a class that it's the only job it's to write several significant metrics and relevant data into a file to be processed on a later date.

The Game class does its job by calling several Agent class methods. When agents are created, their propensity vector is initialized with random values between [0, 1[unless if it specified a position of the propensity vector that should have a bias. This position would start with a fixed positive value and all other positions on the vector are initialized with zero. This propensity vector is the core of the Agent's learning and is what that defines its behaviour, so it's natural that must of the Agent's methods interacts directly with their propensity vector. The first operation that an agent must do is to make a proposal with the *makeProposal* method according to their propensity vector values or making a random action with a certain probability (error rate). This method returns a double that represents the contribution of the agent to the common pot. The methodology used to pick the strategy was discussed in section 3.1. The private *pickStrategy* method is an auxiliary method that helps to compute the value of the strategy. The *sumPropensityVector* is also a private method of the class that computes the sum of all the values in the propensity vector, this is useful for computing the probability of a given position in the propensity vector is chosen. Finally, the agent must update their propensity vector. The *updatePropensityVector* receives the reward for the previously played action as input and updates its propensity vector. This is done by applying the formulas discussed in section 3.1.

The *pickStretagy* method is implemented on a separated class that must implement the Strategy interface. As an example, we implemented two different classes, the one used in this work the

LinearStrategy that is presented in section 3.1 (formula 2) and the ExponentialStrategy. The probability for a given position in the propensity vector being picked using the exponential strategy picker is very similar to the linear one:

$$\rho_{ki}(t) = \frac{e^{Q_{ki}(t)}}{\sum_n e^{Q_{kn}(t)}} \quad (5)$$

The only difference is that the values in propensity vector are now the exponent of a power of base *e.* This formula (based on the Boltzmann Q-learning) has the advantage of dealing with negative values in the propensity vector. However due to the exponential nature of this formula, reinforcement is much stronger than in the liner strategy picker, this means that the agents would have fewer opportunities to explore all the actions. After several initial experiments, we decided to only use on this project the linear strategy because it gives better and most realistic results. But we still need to solve the problem with negative values in the propensity vector when using the linear strategy picker. Rewards can be negative values that are added to a position in the propensity vector, so it could happen that the result of the operation is negative. If we do not solve this issue, the result of the linear strategy picker could be a negative probability. Our solution is to limit the values in the propensity vector to zero. If a result of the update of the propensity vector is less than (or a value very close) to zero, then we correct this value and force the value of the operation to assume the value zero ensuring that we don't end up with any mathematical inconsistencies.

3.4 Production Environment

To run our experiments, we used a cluster with 100+ CPU cores. To take advantage of this, we set up a Linux bash script that would launch several simulations with different parameters at the same time, each one producing data and outputting it to a file. The next step is to collect all these files (could be over one hundred of them) and processing them.
We used a python script to merge all the relevant data from all these files to a single file, ready to be used in the production of charts and new statistical evaluations.

All the results that are presented in the next sections were produced in this environment using these scripts.
As already discussed in this chapter, one of the most critical parameters that have a direct impact on the agents' reward is the risk value. We started by doing several experiments for several group sizes and risk values and observed the results. Because there is a certain amount of randomness involved in each experiment we repeated each one of them 50 times (50 runs) and plotted the average results. All used parameters in the experiments are stated in the chart's captions.

Chapter 4. Risk and the emergence of cooperation

There are many experiments that were made around the role of risk on cooperation scenarios [1, 3, 27, 30, 37, 38] however we introduce a novel environment with a very different context, so we will start by making studying the impact of risk in our simulation.

We can see in that, the risk value is indeed a huge factor in the emergence of cooperation.

Risk is crucial, as both charts shows. The bigger the risk and smaller the group size (*N*), the easier is for cooperation to emerge.

Another conclusion we can take from the data presented is that big groups require bigger perceived risk before start coordinating between themselves and start cooperating. Groups large enough, in our simulation groups of 12 or bigger, don't even show any signs of cooperation.

At the upper pane of , we can extract another interesting result. For all group sizes, when they decide to fully cooperate with each other, they do it in a fair manner, that is, the average individual contributions are the smallest possible but still large enough to reach the threshold (*M*).

This is a fascinating result because in the real-world, climate change summits usually involve many countries in a single deal. Our simulations show that this is may be contra productive for reaching a lasting group cooperation and a sustainable deal between all involved parties, which agrees with the previous results evidencing the increased efficiency of local and smaller institutions [30].

The best way to obtain sustainable cooperation with relatively small risk values is to have small groups working together to achieve a common goal.

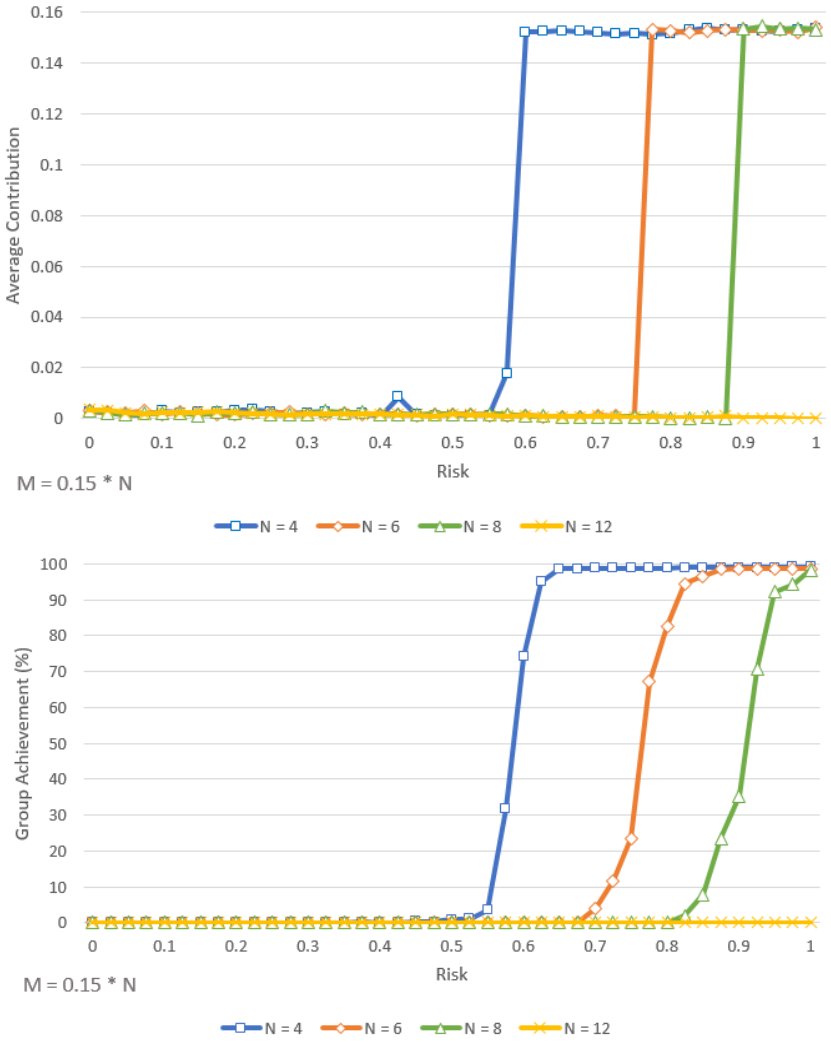


Fig. 4 Simple risk impact on cooperation by varying the risk value. Risk value was changed in steps of 0.025, between 0 and 1 (inclusive). At the upper pane, we can see the average contribution per agent ate the end of the simulations for several group sizes (*N*), and in the bottom pane, we can see the percentage of games won (group achievement) for the same group sizes. Parameters: *M* = 0.15*N*, *N* = [4,6,8,12], γ (*forgetting rate)* = 0.01, ϵ (*local exploration)* = 0.01, *V* (*propensity vector size)* = 20, error rate = 0.01, *agents (Z)* = 100, *maximum time-steps* = 25.000, *runs* = 50.

4.1 Threshold Impact on Cooperation

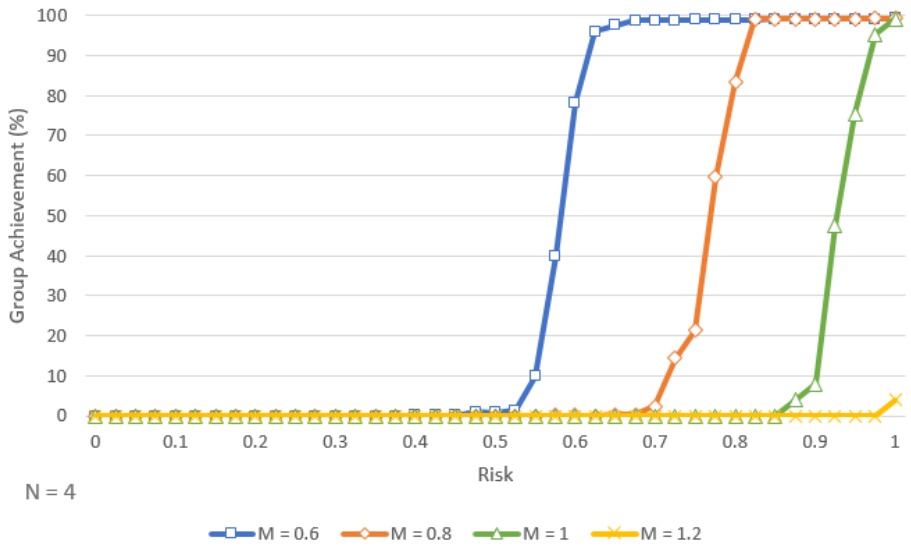


Fig. 5 Simple threshold impact on cooperation. We varied the risk value for a fixed group size (N = 4) for several threshold (M) values. The bigger the threshold, more difficult is to have a big percentage of group achievement. Parameters: M = 0.6 (blue line), M = 0.8 (orange line), M = 1 (green line), M = 1.2 (yellow line), N = 4, γ = 0.01, ϵ = 0.01, error rate = 0, V = 20, agents (Z) = 100, *maximum time-steps* = 25.000, *runs* = 50.

In Fig. 5 we can see a very similar pattern that was already discussed in chapter 0.

When we fixed the group size (N = 4) and varied the risk value using several different thresholds, we can see again that the when the risk values are significant, cooperation is easier to achieve.

Of course, for lower M it is easier to achieve the threshold, than it is natural that the agents need fewer incentives (lower risk) to cooperate. When the threshold is too high, not even very high levels of risk can make the agents to fully cooperate; In this case, we would need new incentives mechanisms to motivate agents to cooperate, like the implementation of institutions [30], for example.

Notice that cooperation here also rises very rapidly when risk is above a certain point, making the percentage of group achievement climb from nearly 0% to almost 100% with a slight risk change in all tested threshold (M) values.

4.2 Learning Process

Until now we only saw what happens to some metrics (average contribution per agent, the percentage of group achievement, etc.) at the end of the simulation. However, it would also be interesting to see how the agents learn and how their contributions change with time (in our case, virtual time or time-steps).

In we can see two important features from both our learning algorithm, the Roth-Erev reinforcement learning and from real human behaviour [31, 39]: the Law of Effect [32] and the Power Law of Practice [33].

The Law of Effects simply states that humans tend to reinforce the use of previously used successful actions and the Power Law of Practice states that learning curve for a given task in a human is initially steep by over time becomes flat [21].

The Power Law of Practice is very easy to see, the line is very steep at the first 1,000 time-steps and becomes progressively flat over the next 9,000-10,000 time-steps.

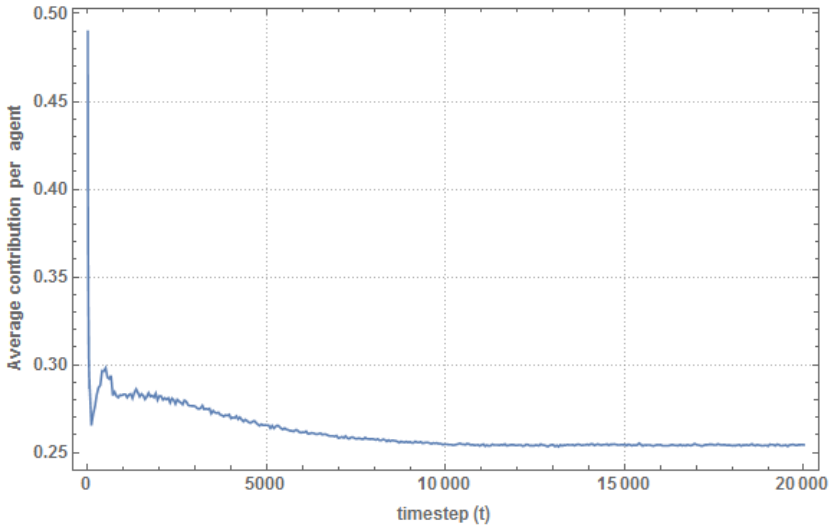


Fig. 6 Evolution of the average contribution per agent in a game. Much of the learning process happens in the beginning of the simulation, on the first 1,000 time-steps. After this period, the agent learns to optimize their contributions. This takes the agent other 9,000-10,000 time-steps. Parameters: *group size (N)* = 4, *threshold (M)* = 0.96, γ = 0.01, ϵ = 0, error rate = 0, *V* = 20, *risk* = 1, *agents (Z)* = 100, *time-steps* = 20.000, *runs* = 20.

The Law of Effect is visible in the last 5,000 to 10,000 time-steps, when the agents have a very solid contribution line: their contributions are on average always around 0.25 with very small variations. We believe that these little variations only happen due to our also small *local experimentation* rate, that allows agents to explore similar actions to the ones that were previously made by them.

At the very beginning of the simulation, at time-step 0, we can see that agents start on average with a contribution very close of 0.5. This is to be expected because agents start with their propensity vector randomly populated using a uniform probability distribution that results in an average initial contribution of around 0.5. Finally, agents after around 9,000 times-steps, converge their contributions to 0.25, which is the minimum contribution possible but without losing the group cooperation.

We picked these simulation parameters from one of the many scenarios that we know that agents would cooperate.

4.2.1 Contributions Variance

We also decided to do some statistical analysis to the contributions of the agents. We picked two scenarios from (upper pane) where we knew that agents would cooperate, and another where they would not.

For the scenario where agents would cooperate we picked a group size of four ($N = 4$), a threshold of 0.6 ($M = 0.15N$) and a *risk* of 0.65. The scenario where agents do not contribute is the same, but the group size is six ($N = 6$) [2].

In this section, we will analyse two different metrics that can help us to better understand the contribution's behaviour of the agents.

1. The average most common contribution by the end of each of the 50 simulations made in each scenario
2. The average variance of the contributions made by the agents in 4 different range of times-steps:
   (a) [0, 1000[
   (b) [1000, 5000[
   (c) [5000, end[
   (d) [0, end[
   (Note: "end" symbolises the end time-step of each simulation)

The ranges above were selected to give us a better view of the variance in each period of the agent's learning, in , the beginning we have a sharp and fast learning, and as the time-steps progresses, the learning becomes much more stable. See Fig. 7 for a visual illustration of these selected ranges.

In Fig. 8 we can conclude that most agents learned to contribute the fair Nash Equilibrium value (0.15). However, a small portion of the agents still chose to contribute 0.1, slightly less than the optimal value and a few outliers, learned to free-ride the other's contributions (around two agents on average in every simulation, out of a population (Z) of 100).

For the scenario where agents would not contribute, all agents in all simulations learned to donate nothing. This is an interesting result that shows that when cooperation is not achieved, then there is a single agent that can make any other contribution, other than zero by the end of our simulation.

[2] All simulation parameters: $M = 0.15N$, $N = 4$ (6), *risk* = 0.65, $\gamma = 0.01$, $\epsilon = 0.01$, *error rate* = 0, *propensity vector size* = 20, *agents (Z)* = 100, *maximum time-steps* = 25.000, *runs* = 50.

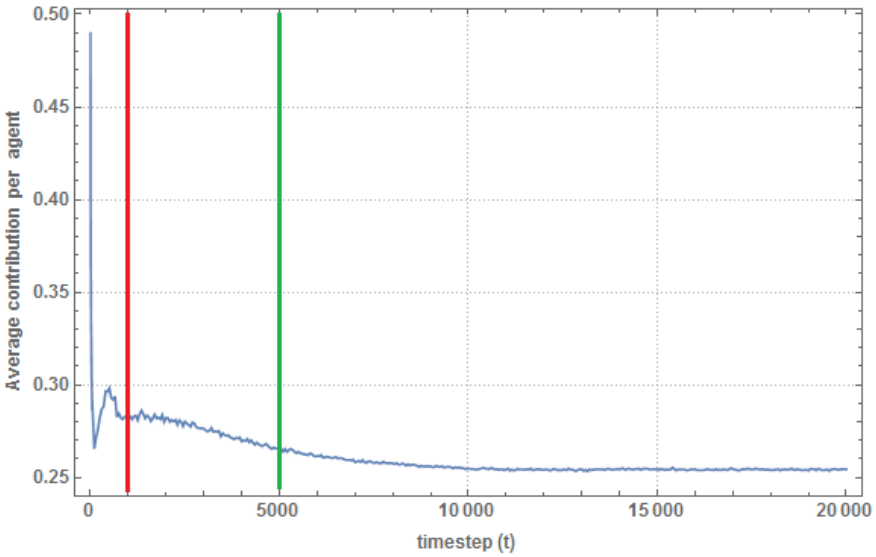


Fig. 7 Visual representation of all ranges where we will measure the contribution variance. (a) from the begin to the time-step 1000 (red line), (b) from the time-step 1000 to the 5000 (green line), (c) from the time-step 5000 to the end of the simulation and (d) from the begin to the end. Parameters: M = 0.15N, N = 4 (6), risk = 0.65, γ = 0.01, ϵ = 0.01, error rate = 0, propensity vector size = 20, agents (Z) = 100, maximum time-steps = 25.000, runs = 50.

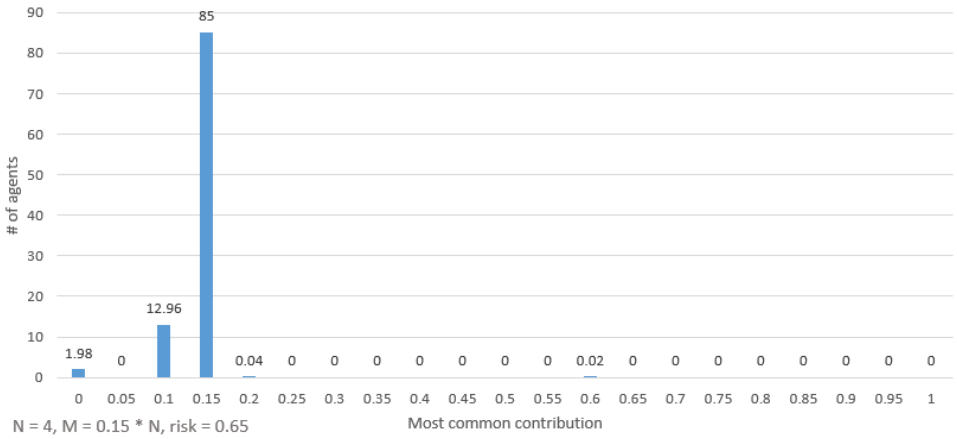


Fig. 8 Average most common contributions of the agents by the end of each of the 50 simulations on the scenario where agents cooperate. We can observe that most agents 85 of the 100 (*Z*) chose to contribute the fair Nash Equilibrium value. However, almost 13 out of the 100, opted to contribute slight less (0.1). Parameters: M = *0.15*N, N = *4 (6),* risk = *0.65, γ = 0.01, ϵ = 0*.01, error rate = *0,* propensity vector size = *20,* agents (Z) = *100,* maximum time-steps = *25.000,* runs = *50.*

Table 4 Average variance values for several time-steps ranges for the scenario where agents cooperate (*N = 4*) and don't cooperate (*N = 6*).

Range	*N = 4*	*N = 6*
[0, 1000[	0.014005	0.009917
[1000, 5000[	0.001916	0.000133

[5000, end[	0.000402	0.000045
[0, end[	0.00246	0.000699

On Table 4, we can see that the obtained results are consistent with the previous study (most common contributions). When the agents do not contribute, they all converge to the same action (contribute zero), and they learn to do this very fast, in only a few time-steps, therefore, in this scenario, the average of contribution's variance in all measured ranges are lower when compared with the same ranges in the scenario where agents do cooperate.
When agents cooperate, they take more time to learn, and we have seen in Fig. 8 that, not every agent learned the same, however, a vast majority of the agents did, in fact, learn the same two actions (0.1 and 0.15) contributing to the low values of variance on Table 4.

To produce the Table 4, we added three different linked-lists on each agent, where we recorded every play for each range. We had a list that contained the plays made during the first 1000 time-steps, another for the next 4000 time-steps and finally another list for the rest of the time-steps.
After having all plays, it is trivial to calculate the variance for each agent for each range. The presented results were computed by making a simple average between of the variances of all agents of each range. In the end, we merged all three lists and applied the same process to calculate variance average from the begin of the simulation until the end. Finally, we present the average results of 50 simulations, to avoid any initial bias that an individual simulation might have.
Fig. 8 was made by using a similar process. By the end of the simulation, we retrieved the weighted average from the propensity vector and rounded up (to the up nearest possible play). The results observed in this figure are the average obtained in 50 different simulations.

4.2.2 Learning Convergence

In section 4.2.1 we explored how the majority of agents converge to a single action, in both scenarios that promote cooperation and scenarios that do not.
We can plot what actions agents prefer playing with time and further validate the previously discussed data. The plots were made by calculating the element-wise average of the propensity vector of the populations every 50 timesteps in two scenarios: One where we know before-hand that agents would not cooperate (*N = 4, M = 0.15N, risk = 0.4*) and other where agents cooperate (*N = 4, M = 0.15N, risk = 0.7*). In Fig. 9 agents start their propensity vectors initialized with random values and then, we can observe that indeed most agents converge to a single equilibrium point, in the upper pane that equilibrium is 0.15 (the fair contribution value) and on the bottom pane is 0. However, it is fascinating to see that agents learn faster not to contribute than the opposite, where agents explore more strategies before converging.

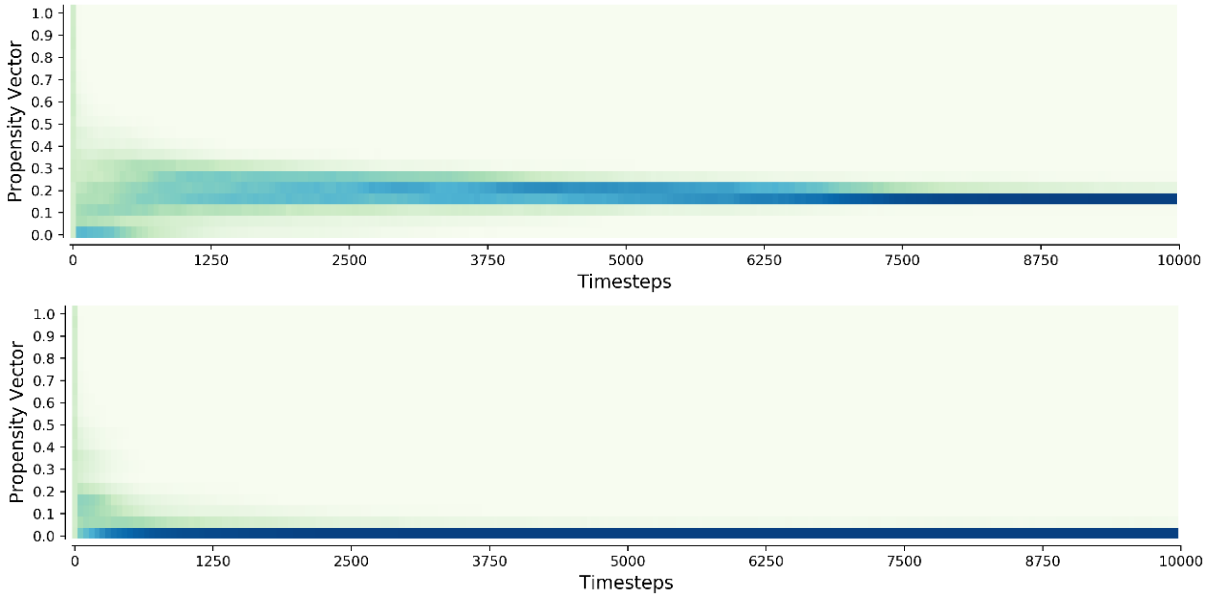


Fig. 9 Learning convergence in well-defined risk scenarios. In the top pane we have a simulation where agents converge to the fair contribution value of 0.15 and in the bottom pane we have a simulation where agents choose not to contribute and rapidly converge to 0. The convergence process in the bottom pane is much faster that in the upper pane. These are density plots, where grey represents a strategy that is not used, green and light blue represent strategies that are moderately used and finally dark blue represents a big concentration of played strategies. Parameters: M = *0.15N,* N = *4,* risk = *0.7* (upper pane), risk = *0.4* (bottom pane), γ = *0.01,* ϵ = *0*.01*,* error rate = *0,* propensity vector size = *20,* agents (Z) = *100,* maximum time-steps = *10.000,* runs = *1.*

A more interesting experiment is to see how agents learn group achievement is bigger than 0% but lower than 100%. For example, in (upper pane), for *N = 4* (blue line), this condition is satisfied when the risk is in the range of [0.56, 0.60]. We chose to experiment when the *risk = 0.56*.

Due to the stochastic nature of our simulation and due to the fact that, by definition, experiments with these settings display a “changing” behaviour (agents start to converge from contributing 0 to contribute the fair value), we obtained slightly different results on each run of our simulation.

In Fig. 10 we show three different results on three different runs. We observed that all results obtained on all executed runs were very similar to one of these three plots presented.

On the upper pane we can see that the clear majority of agents converged to zero, however, a small minority learned to contribute 0.6, in a game where the threshold (*M*) is 0.6. This is very interesting because it suggests that a small fraction of players, decided to by themselves, do “all the work” and allow the whole group to free-ride.

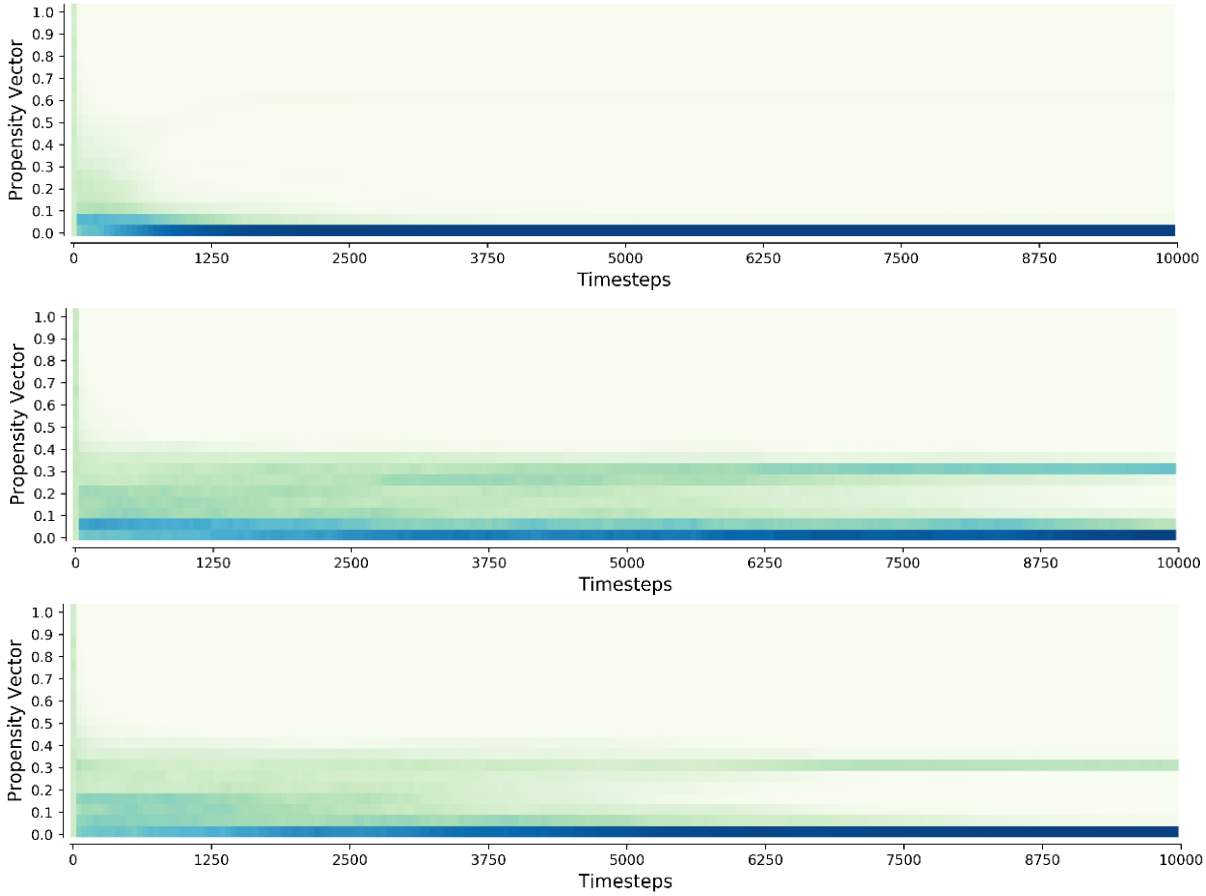


Fig. 10 Learning convergence on a behaviour tipping point on 3 separate runs with the same parameters. On the upper pane the large majority of agents learned to contribute zero, however, a small minority of players chose to contribute 0.6, in a game where the threshold is 0.6. On the two bottom panes, we can observe the emergence of several equilibria contribution points around 0 and 0.3. These are density plots, where grey represents a strategy that is not used, green and light blue represent strategies that are moderately used and finally dark blue represents a big concentration of played strategies. Parameters: M = *0.15N,* N = *4,* risk = *0.56*, γ = *0.01,* ϵ = *0*.01, error rate = *0,* propensity vector size = *20,* agents (Z) = *100,* maximum time-steps = *10.000,* runs = *1*

The two bottom panes display similar behaviours. They show the emergence of two main equilibria, one at 0 and at 0.3. However, at the second pane, it is clearly visible multiple other equilibria points, around these values. This is very interesting because it shows that this is a setting where heterogeneous groups can co-exist. Where we have both free-riders, and agents that contribute to the common pot with different values. We can also note that in these bottom two plots, convergence takes more time and is a lot less “well-defined” that in simulations where agents take all the same action (Fig. 9).

These are fascinating results. They show that when agents start cooperating (but not all of them) we can have groups that cooperate more than their fair share in order to avoid risks and costs to themselves, even if it means allowing other agents to free-ride. However, this only works on a fraction of games, if a group of players want to achieve sustained levels of cooperation in whatever groups they play, they must all cooperate and contribute to the common good.

4.3 Dependence on the Initial Conditions

Another of the first experiments we should do is to investigate what happens to cooperation if agents, instead of being initialized with random values in their propensity vectors, these vectors would be initialized with a very small value (0.05) in all positions of the vector, except in one determined position whose value would be 10. This specific position initialized with the value 10, would be the same for all agents in our simulation.

We considered this to be an interesting topic to study because until now, we have not seen in any of our experiments the impact of the agent's initial condition.

When the agents are created with their propensity vectors filled with random values between 0 and 1, it is expected to have agents that are more "generous" or more "selfish" than the rest, but with a significantly large population, like the ones we have in our program, these small variations even out.

We considered a "generous" agent, an agent that initially has a bigger probability of contributing *M/N* of their endowment or more to the common pot. A "selfish" agent is the opposite; it is an agent that has a bigger probability of contributing to the common pot, values less than *M/N*.

The value of *M/N* is a crucial one; it is the optimal value that each agent on any game, must contribute to achieving the threshold *M* (considering that all agents are equal and contribute equal values, that may not be the case). Any value contributed to the common pot bigger than *M/N*, is not optimal, but it may be needed to compensate any agent that contribute less than the optimal value.

In this experiment, we will set the same parameters to all simulations, changing only the position that is initialized on all agents' propensity vector (which corresponds to a contribution value, that is shown on Fig. 11).

In Fig. 11 we can extract several interesting conclusions. However, before we analyse the results it is important to calculate the value of *M/N* in this experiment*: M/N = 0.6 / 4 = 0.15*.

We picked four attractive risk scenarios: two in which the agents would not cooperate (blue and orange lines), one in which they would, but are very close to the tipping point - point where agents start to cooperate - (green line) and finally, a scenario where all agents are subject to a higher risk value and therefore, shown a very high probability of cooperating between them (yellow line).

In the blue and green lines, we can see that agents still do not cooperate. However, there is a spike in the group achievement on the green line when the initial condition is 0.15, our *M/N* value. This shows that even in scenarios where cooperation expectations are low (due to the low risk, but not too low), cooperation can emerge if all agents are created with a strong bias towards the optimal contribution value. All other starting values for their propensity vector, fail to achieve any cooperation because the agents do not have enough incentives to do so. This spike can be observed when we have a low enough risk to not see a generalized cooperation (like in the yellow line) but sufficiently high to allow agents sometimes to decide to cooperate.

Both green and yellow lines show that if agents start with values lower than our *M/N,* agents have more difficulty to cooperate, with a group achievement percentage that grows with risk, a pattern already seen and studied in previous experiments.

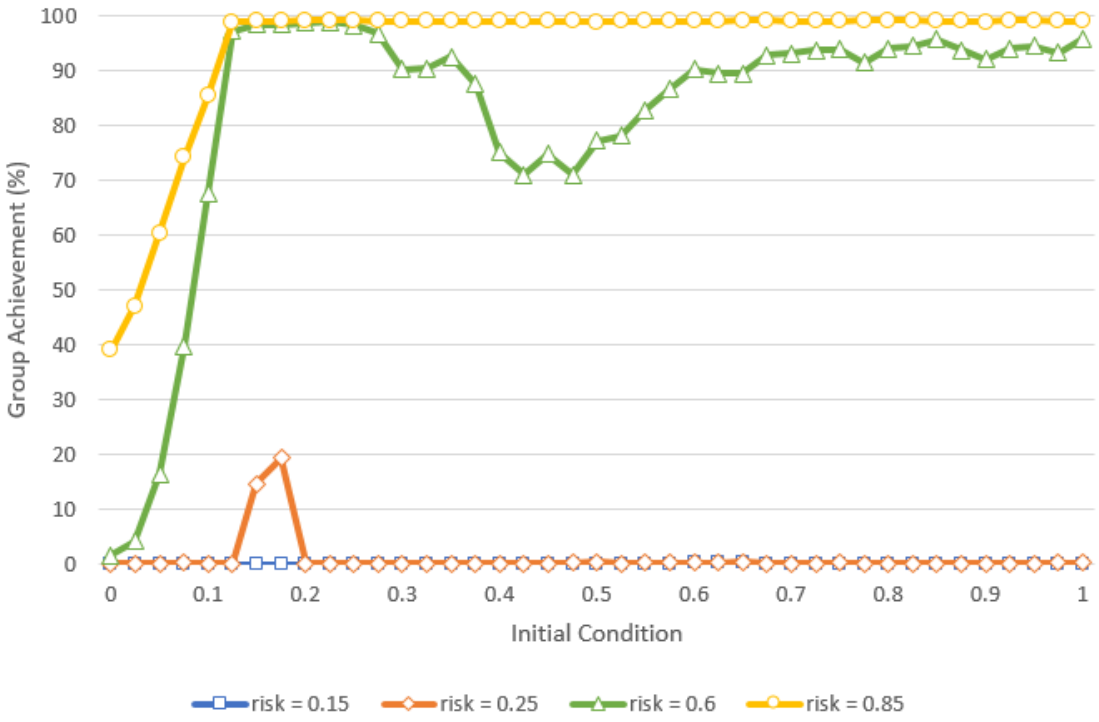


Fig. 11 Initial condition. We can observe a common pattern: if the agents are created with the bias of contributing around the optimal value for the game, than agents have a bigger chance to cooperate between them, even in scenarios where cooperation was not previously possible when agents were created with random values in their propensity vector (orange line). Parameters: N = 4, M = 0.6, γ = 0.01, ϵ = 0.01, error rate = 0, *propensity vector size* = 20, risk = 0.15 (blue line), *risk* = 0.25 (orange line), *risk* = 0.6 (green line), *risk* = 0.85 (yellow line), agents (Z) = 100, *maximum time-steps* = 25.000, *runs* = 50, V = 20.

With bigger risk values (green and yellow lines), when the initial condition is greater than M/N than the agents cooperate more between them.

However, we can see that when the risk is equal to 0.6, cooperation values are not very stable. This is caused by the risk value used in this simulation. Lower risk values can cause agents to sometimes prefer not to contribute to the common pot.

We can conclude that the initial conditions of the agents can have an impact, but only in very specific circumstances, when the risk is not high enough, and all agents have a very strong bias towards the contributions optimal value. In our regular simulations, we assign a random value for all propensity vector positions for every agent, combining this with a big enough population (Z), then we can have a significant degree of certainty than our simulations will not have a strong bias towards any particular strategy.

4.3.1 Learning process of selfish agents

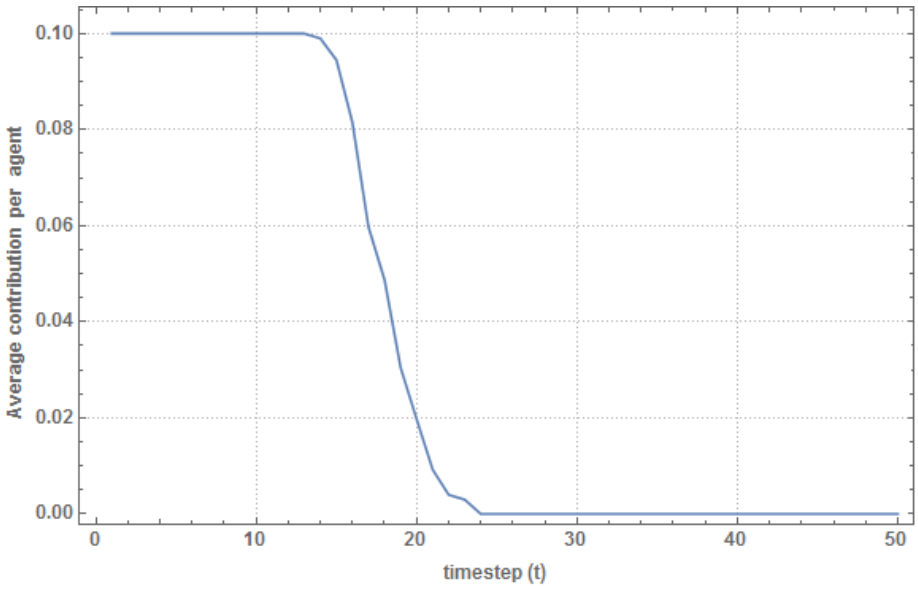


Fig. 12 Evolution of the average contribution of "selfish" agents in a game. All learning process happens in the first 24 time-steps, after that period all contributions stay at 0. Parameters: *group size (N)* = 4, *threshold (M)* = 0.96, *initial condition* = 0.1, γ = 0.01, ϵ = 0.01, *V* = 20, *error rate* = 0, *risk* = 1.0, *agents (Z)* = 100, *time-steps* = 20.000, *runs* = 20.

We can see in Fig. 12 that the initial condition does not affect the observed properties described in section 4.2 (Power Law of Practice and Law of Effects).

All the learning happens in the first couple of dozens of time-steps, and after this initial period, agents start not contributing at all and stay that way in all simulation.

In this scenario, we have agents that start only to contribute 0.1 of their endowment to the common pot, that in Fig. 11 proved to be insufficient to promote long-term cooperation and here we can see the root cause. Agents are unable to learn to contribute more to achieve cooperation, they try to donate the same values for a few time-steps and then, stop trying and choose not to spend more of their endowment on an effort that does not provide any reward.

One way to avoid this behaviour is to allow agents to explore more by increasing their "error rate" – the change for a given agent to make a random action, totally ignoring their propensity vector. This possibility and their results are explored in section 4.4.

4.4 Error Rate

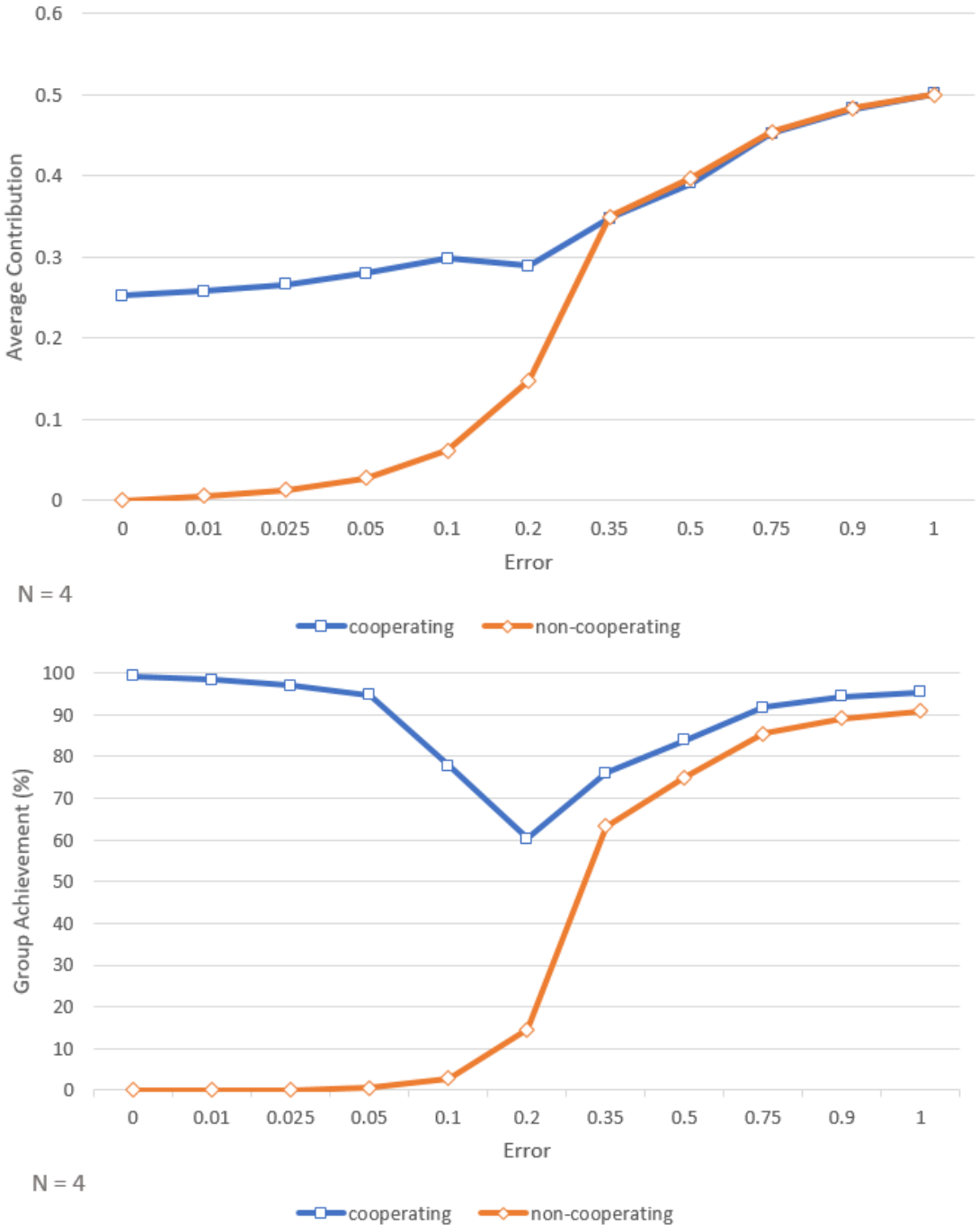


Fig. 13. The impact of the error on cooperation. Varying the value of the "error" parameter (probability of an agent making an entirely random action, ignoring their propensity vector) from 0 to 1 in increasing increments. In the upper pane is shown the average contribution per agent on the last 50 time-steps of the simulation and pane below illustrates the percentage of group achievement in the last 50 time-steps of the simulation. The blue line indicates the impact of the error on a scenario that we previously knew that would converge into cooperation and the orange line a scenario where we know that would not converge to cooperation. The error only seems to help groups where cooperation does not emerge naturally (orange line). Parameters: M = 1 (blue line), M = 1.2 (orange line), ϵ = 0.01, γ = 0.01, V = 20, *risk* = 1, agents (Z) = 100, *maximum time-steps* = 25.000, *runs* = 50.

In our system, we considered the “error” to be a probability (value in [0, 1]) of a given agent to make a random action without taking into account their respective propensity vector. It might be important to give agents an opportunity to error, to let them explore more actions and avoid falling into a local minimum/maximum in their learning. This error should not be confused with the local experimentation error introduced in section 2.2.8.

To test how varying the change of occurring an error impacts the global cooperation, in our environment, we pick two different scenarios, with only one difference between them: one of them with a lower threshold (M = 1) where we know from previous experiments that agents would converge to cooperation (where we expected at the end of the simulation having a percentage of group achievement – where the sum of all contributions in equal or bigger than M – to be very close to 100%), and other simulation with a higher threshold (M = 1.2) where we know that agents would choose not to cooperate (having the percentage of group achievement on the end of the simulation very close to 0%). All other parameters were the same in these two simulations.

In *Fig. 13* we can see that indeed the high cooperation scenario (blue line) when the error is 0, their group achievement percentage is above 98% confirming that without any error, their natural behavior is to cooperate, on the contrary, the low cooperation scenario (orange line, M = 1.2) their default behavior is not to cooperate (group achievement is below 1%).

If we analyse the high cooperation scenario first (blue line), we can see that the error is always contra-productive. Agents learn to cooperate and to optimise their contributions, and the error forces the agents to ignore all their already optimised behaviour. Any amount of error (even tiny) prove to be unhelpful to the agents and the global levels of cooperation.

With the low cooperation scenario (orange line) we see an increase of the cooperation levels, both on the values of the contribution and on the group achievement, however, we cannot see any signs of enhanced cooperation due to learning. The increased change of cooperation and the values of contributions observed in the charts can all be explained by the value of the error.

In conclusion, despite the error rate being a real-world phenomenon, in our system, it seems to provide no advantage to the agents to have it. However, we may still use low levels of the error (around 0.01) in our simulations sometimes whenever it seems fit to allow agents to make non-optimal decisions or to make some exploratory actions (like many Humans in real-life situations [35]).

To sum up, in this section, we discussed the impact of the risk and errors on the agent’s cooperation and analysed how the agents learn.

Until now, we had agents that were all equal (except their propensity vector that is random), and in our experiments, almost all agents converged to the same strategy/action. However, in the real-world negotiations rounds are not that easy and show lots of variability of actions and positions taken by nations. One of the reasons for this variability is the existence of external factors that influence the agent’s decisions in many different ways. Simulating all possibilities of external factors would be impossible due to the number of external factors that exist.

In this thesis, we will focus mainly on one simplified version of these external factors that play a vital role in negotiations between nations and one that we consider to be one of the most relevant – the wealth inequality.

Chapter 5. Wealth Inequality and Uncertainty

In this charpter we will report the results obtained by experimenting with two different external factors that impact real-world climate change negotiations: Wealth Inequality between countries and Goal uncertainty.

5.1. Wealth Inequality

Wealth Inequality is a pervasive reality in the real world; some countries have more resources available than others. We followed the methodology used in *Vitor V. Vasconcelos et al.* (2014) paper [34], and we will have 20% of agents richer (with more endowment) than the other 80%. If we chose a group size where it is not possible to obtain exactly 80% of poor agents, we will round up the number of poor agents to the nearest integer. This is a simplification of the wealth inequality problem because, in the real world, different countries have many more variables that influence their willingness to contribute to a solution for a common problem and have many different interests and motivation. In this thesis, we are more interested in studying if agents with more resources are willing to contribute a bigger portion of their wealth to compensate the lack of resources of the other agents. In this project, we called a “rich agent” to an agent that has more wealth than a “poor agent” and they are both equally available to contribute to the pot.

We also considered that all rich agents have the same wealth as all other rich agents and the same is applied to the poor agents.

For a given poor agent *i* it’s endowment E_i is provided by the following formula:

$$E_i = w.b \tag{6}$$

Where *b* is their regular endowment (in all previous experiments, we considered *b* = 1), and *w* is the new configurable parameter, that we call wealth equality factor (must be smaller or equal to 1) that will determine how the poor agents will be. When *w* = 1, we will consider that there is no wealth inequality present in the simulation, that is, all agents will have the same endowment, and when *w* < 1, we consider those poor agents will have *w* times the original wealth value. For example, *w* = 0.5, means the poor agents will start with half of their original wealth.

For a rich agent *i*, their endowment in a simulation with a group size of *N* and with N_r rich agents and N_p poor agents, is given by:

$$E_i = \frac{N - (w.N_p)}{N_r} \tag{7}$$

By default, $N_p$and $N_r$are given by:

$$N_p = 0.8N \tag{8}$$

$$N_r = 0.2N \tag{9}$$

We picked these formulas because they allow us to keep the sum of all the endowments in a group the same as before, enabling us to make more realistic comparisons in contributions patterns.

An example, for *N* = 4 without any wealth inequality mechanisms, where all agents are given an endowment of 1 is trivial to calculate: 4 * 1 = 4.

The same example, for *N* = 4 with this implementation of wealth inequality the total endowment in a group is given by:

$$\sum_i^N E_i = wbN_p + N_r \frac{N - wN_p}{N_r} \tag{10}$$

So, with *w* = 0.5 and *b* = 1 we get:

$$\sum_i^N E_i = 3(0.5 \times 1) + 1\left(\frac{4 - (3 \times 0.5)}{1}\right) \Leftrightarrow$$

$$\Leftrightarrow \sum_i^N E_i = 1.5 + 2.5 \Leftrightarrow \sum_i^N E_i = 4 \tag{11}$$

We can see that in this example, with Wealth Inequality the poor agents are given an endowment of 0.5 and the rich of 2.5 to keep the total wealth sum of the group the same as before (equal to 4).

With the previous formulas in use, we can compare the results obtained in simulations with the wealth inequality and in simulations without this mechanism.

5.1.1 Impact of the Group Structure

Before wealth inequality, we had groups made by agents picked randomly from our entire population. However, with Wealth Inequality, this presents a problem: if we continue to pick random individuals from our population, in each game, we can have groups composed only but poor agents, or we can even have groups made completely by rich agents, or we can have anything in between. This may

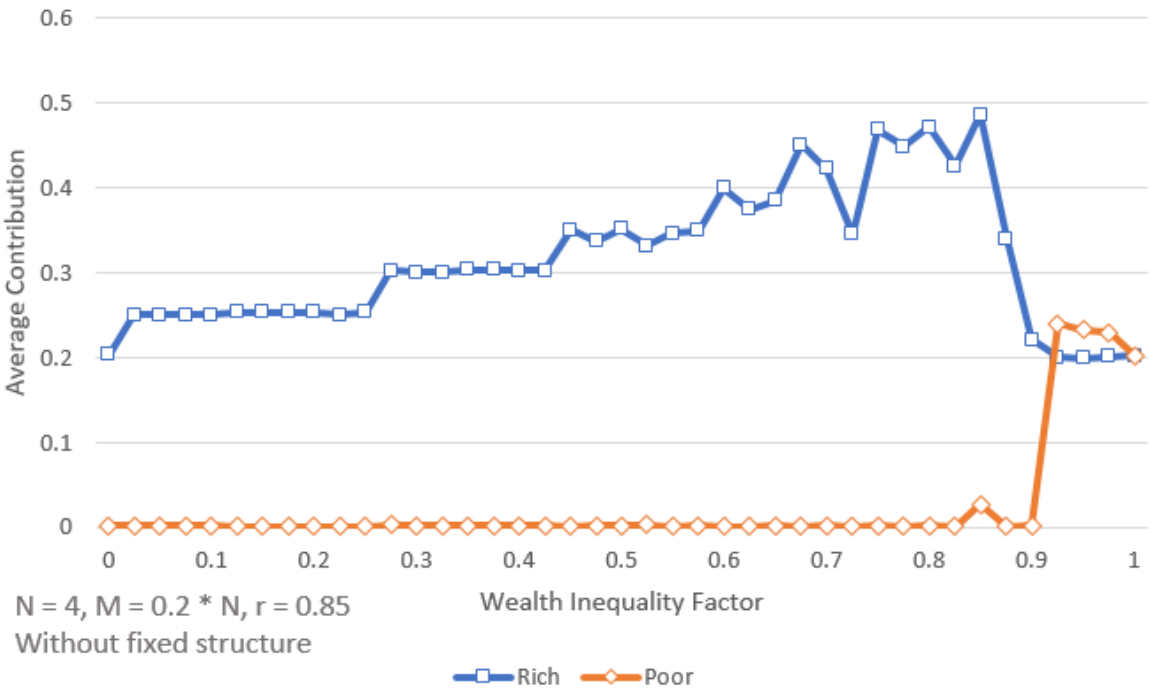
cause a problem related to the fact that agents may learn in a context and act in a completely different one.

Our learning algorithm does not have into account any information about the agent's environment; this means that it does not know who is in the group, what is the threshold, how many agents are in each game, or how the other agents will play. The only way that the algorithm in use can learn is by trying all the actions and see what actions outperform the others; these actions will be picked more and more until almost all actions done by the agent are the same – the action that gets the biggest reward. If the agents are subject to a different environment in each game, the agents will have more difficulties in the learning process, but it is still possible to do it.

In this case, if we pick random agents and every time we can get a varying amount of rich and poor agents, agents must adapt not only to our main cooperation problem but also with the permanent changing structure of their games.

There is also an added benefit of having games with a fixed number of rich and poor agents. It makes it easier to compare the results with previous experiments where the sum of wealth in each game was also fixed.

In the following charts, we will analyze the impact of structured groups and unstructured groups on our cooperation game. Structured groups are groups of N agents composed by 20% of random rich agents and 80% of random poor agents. Example: for N = 4, we will have 1 rich and 3 poor agents in each game. With an unstructured group, all agents are picked randomly, regardless of their wealth status. This means that we can have a variable number of rich and poor agents in each game.



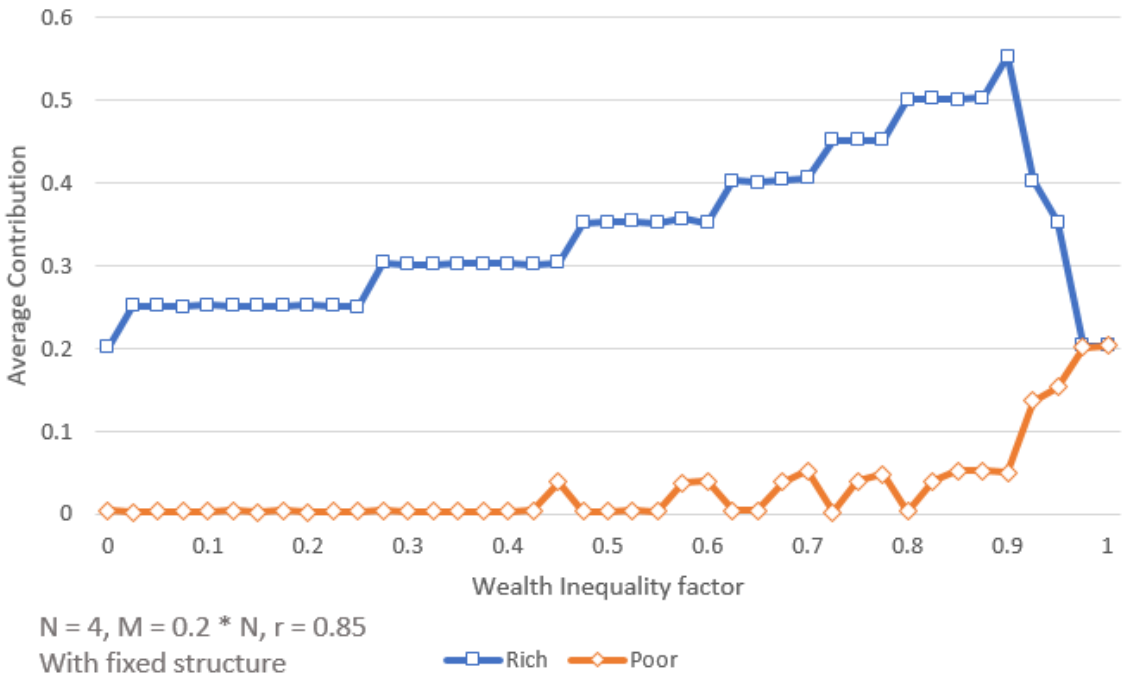


Fig. 14 Contributions patterns of rich and poor agents by varying the Wealth Equality Factor in both unstructured (top pane) and structured (bottom pane) game groups. Both simulations show similar contributions patterns, however, structure groups pattern is more predictable and stable than the unstructured groups. Parameters: N = 4, M = 0.8, γ = 0.01, ϵ = 0.01, error rate = 0, V = 20, *risk* = 0.85, agents (Z) = 100, *maximum time-steps* = 25.000, *runs* = 50.

In Fig. 14 we can see that the structured group offers more solid and sustained contributions by the rich agents to the common pot. The predictability of the group appears to give more confidence to both poor and rich agents to contribute more and to contribute to with lower values of the Wealth Equality Factor, where contributions by the poor are more difficult. When the Wealth Equality Factor is equalled to 1, is the same to say that there is no wealth inequality, so, it makes sense that both structured and unstructured have the same behaviour when this value is very close or equal to 1.

Below we compare in the same chart the sum of contributions from both groups and the percentage of group achievement by each one of them.

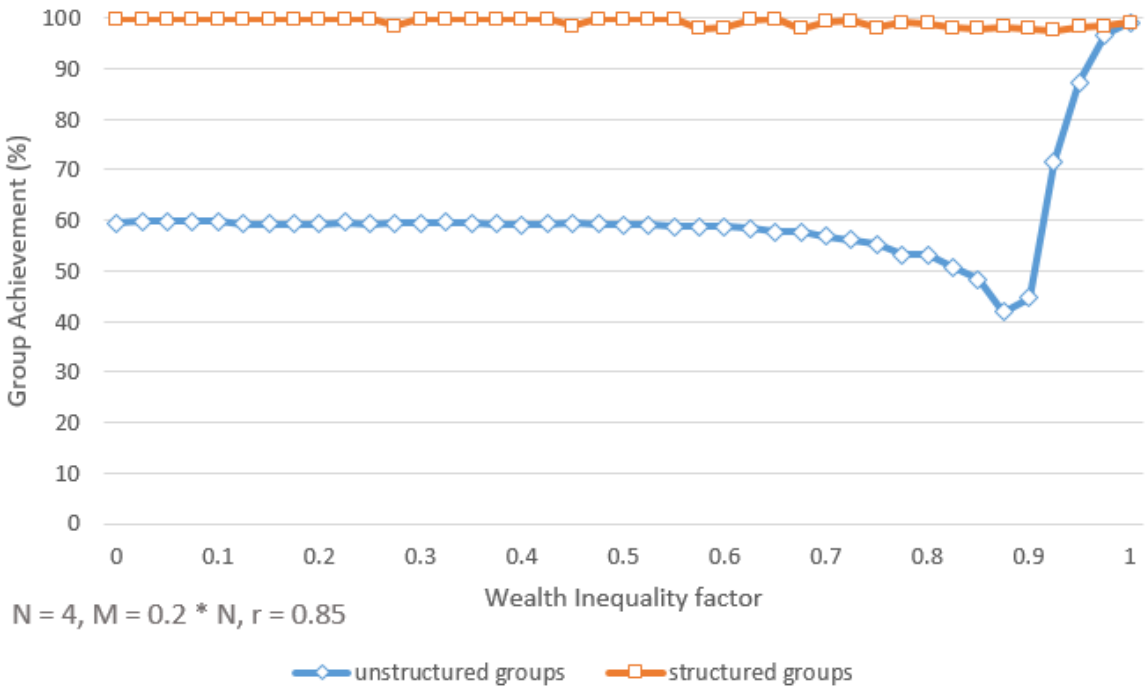


Fig. 15 Percentage of group achievement with both structured and unstructured groups in a wealth inequality scenario. With structured groups (orange line), cooperation is almost always achieved, independently of the wealth equality factor, without structured groups (blue line), the percentage of group achievement is around 40-60% for low amounts of wealth equality factor (< 0.9), however the proportion of the group achievement starts rising between 0.9 and 1. Parameters: N = 4, M = 0.8, γ = 0.01, ϵ = 0.01, error rate = 0, V = 20, *risk* = 0.85, agents (Z) = 100, *maximum time-steps* = 25.000, *runs* = 50.

In Fig. 14 and in Fig. 15 we saw that contributions with unstructured groups were more unstable and made with lower amounts. This had a direct impact on the percentage of the group achievement that can be observed in Fig. 15. It is evident that lower contributions will mean lower amounts of group achievement. Structured groups behaved very well, maintaining high levels of cooperation and group achievement through all tested values for the wealth equality factor.

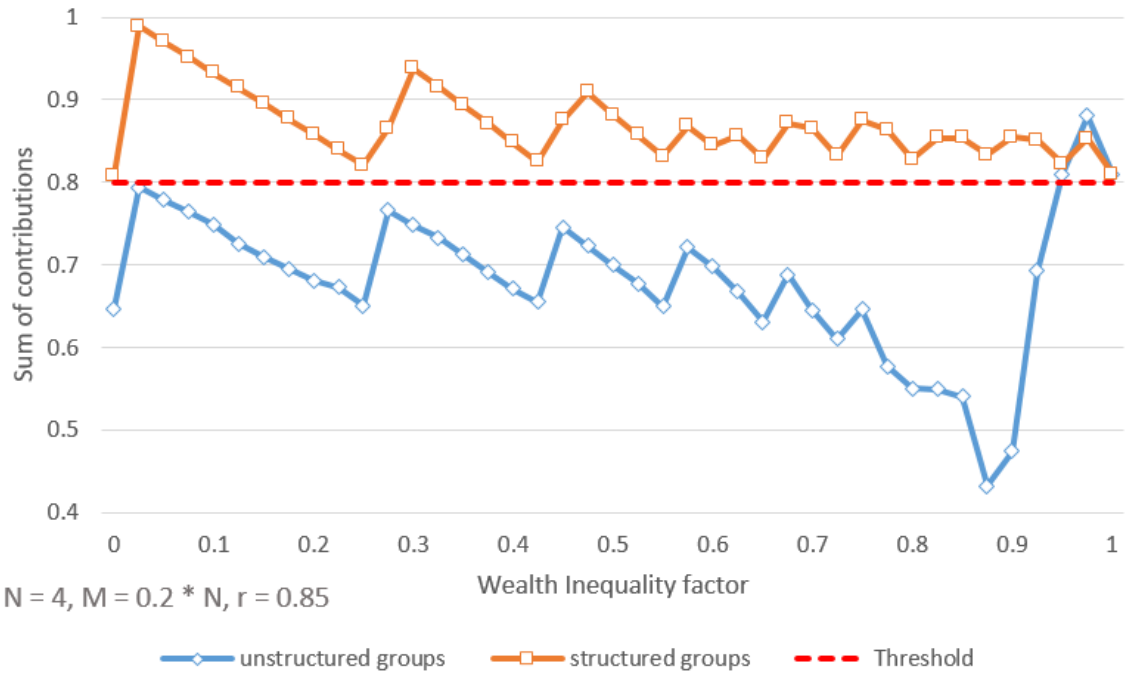


Fig. 16 Comparison between contributions totals with structured and unstructured groups. Structured groups (orange line) systemically had pots above the threshold (red dashed line) unlike unstructured groups (blue line) that were on average close but not above of the threshold, except on high values of wealth equality factor. Parameters: N = 4, M = 0.8, γ = 0.01, ϵ = 0.01, error rate = 0, V = 20, *risk* = 0.85, agents (Z) = 100, *maximum time-steps* = 25.000, *runs* = 50.

In Fig. 16 we can see that on average, unstructured groups, in general, performed worse than the structured groups because their contributions for the common pot were below the threshold, except for high values (> 0.9) of wealth equality factor. This also helps us to understand the data shown in Fig. 15 where the percentage of group achievement for unstructured groups were very low when compared to structured groups.

In this figure, we can see that both groups have a behaviour where there is a significant spike in contributions followed by a steady decline in contributions, followed by a new spike and so on. This response was expected due to the nature of the experiment. By increasing the wealth equality factor, we are decreasing the total amount of capital available to the rich agents (that we already discussed that are the main contributors), however the agents do not change their strategy, causing the sum of contributions to decline until finally, the rich agents decide to contribute a bigger percentage of their wealth causing a new spike in the chart.

Another factor that we must consider is that the rich agents in structures groups play more times than poor agents because there are less rich agents than poor agents. This will help rich agents to learn in fewer time-steps than poor agents. However, this does not play a crucial role in the behaviour of the agents because we give them more than enough time to converge and to learn what are the best strategies.

With this data, we decided that, for future simulations using the wealth inequality functionality, we would use the structured groups in order to get more consistent, stable, and overall, better results. This structured groups can also be easily implemented in the real world by forcing rich countries to work with countries with fewer resources than them. This argues in favor of stable negotiation groups over time, composed by individuals with consistent (over different groups) wealth levels.

5.1.2 Analysis of the impact of the Wealth Inequality on cooperation

In this section, we are going to analyse the effects of Wealth Inequality in our cooperation game. Using structured groups composed by selecting for each game a group consisting of 20% of rich agents and 80% poor agents, as described in section 5.1.1 we will see what is the impact of risk and the threshold in the groups with Wealth Inequality. We will also try to explain the obtained results.

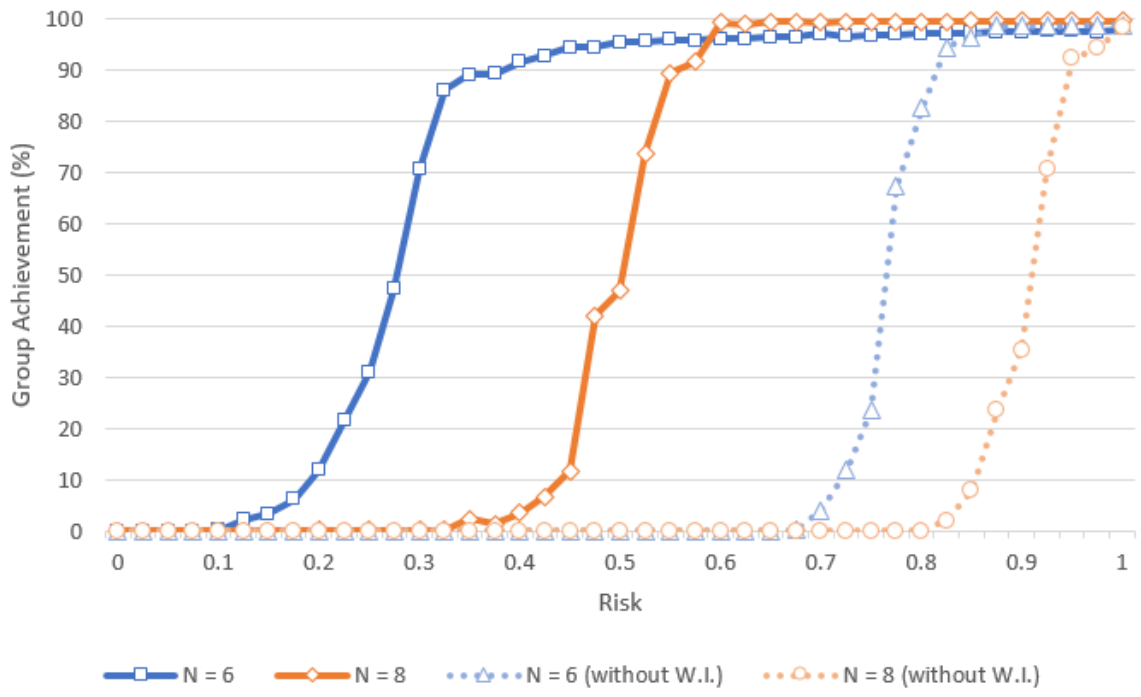


Fig. 17 Effects of varying risk (*r*) levels in groups with wealth inequality (structured groups) for several group sizes (*N*) compared with groups without wealth inequality (W.I.). When compared with groups without wealth inequality, cooperation emerges with much lower levels of risk when compared with groups of the same size. Parameters: *M* = 0.6 (blue line), *M* = 0.9 (orange line), *wealth equality factor* = 0.5, γ = 0.01, ϵ = 0.01, error rate = 0, *V* = 20, agents (*Z*) = 100, *maximum time-steps* = 25.000, *runs* = 50.

In Fig. 17 we can see very similar patterns of cooperation when compared with groups of the same size (*N*) but without wealth inequality (dotted lines). That is, cooperation is easier to achieve within smaller groups than with larger groups and that the cooperation emergence presents a steep curve and ends converging to values of around 98-99% of group achievement (games, where the values contributed to the common pot, is equal or higher than the threshold *M*).

Another fascinating conclusion is that cooperation arises much sooner for all group sizes with wealth inequality. For example, for N = 4, we can see the percentage of group achievement starts to rise when risk is around 0.1. In the same scenario but without wealth inequality, the group achievement percentage only began to grow when risk was around 0.7.

We can also compare what happens to the amount of cooperation when we change the threshold (M) for a given group size N. In Fig. 18 we did this experiment for N = 4 because it was the size of the group that had shown in Fig. *17* that was the best for incentivizing cooperation.

We tried with two different values for *M* = [0.8, 1]. These values were selected because they correspond to all agents contributing with 0.2 and 0.25 of their wealth respectively for each of the mentioned thresholds.

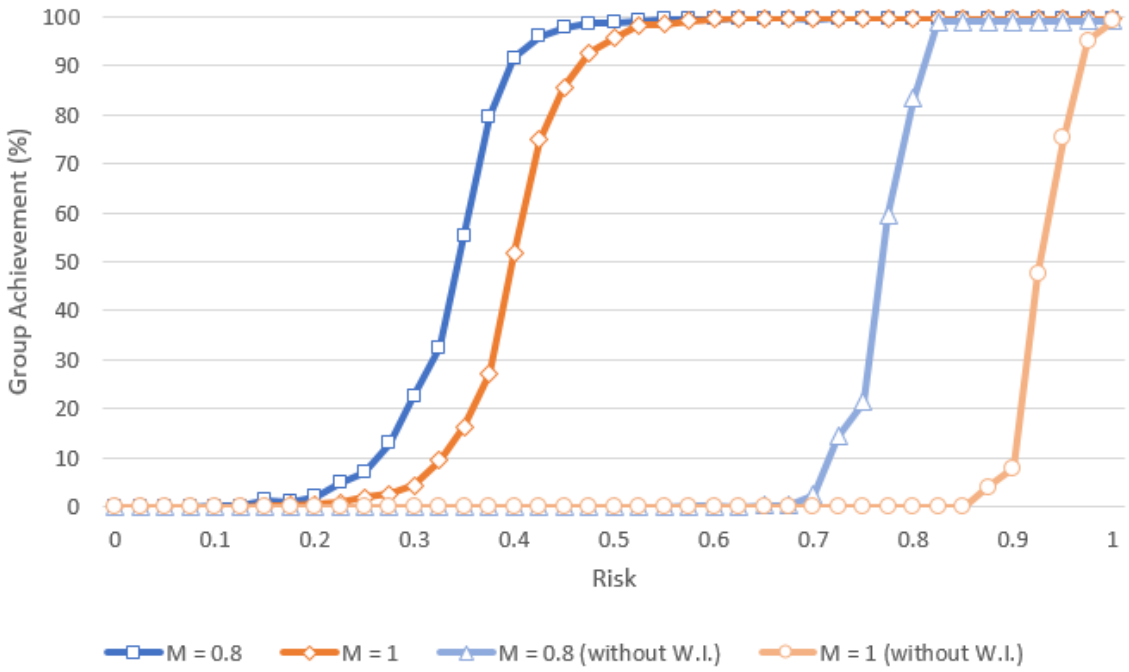


Fig. 18 Effects of varying risk for several thresholds (*M*) levels in groups with *N* = 4 with wealth inequality (structured groups). Behavior is very similar to groups without any wealth inequality. Cooperation is easier to obtain when the threshold is smaller. Parameters: *N* = 4, *M* = 0.6 (blue line), *M* = 0.8 (orange line), *M* = 1 (green line), *M* = 1.2 (yellow line), *wealth equality factor* = 0.5, γ = 0.01, ϵ = 0.01, error rate = 0, *V* = 20, agents (*Z*) = 100, *maximum time-steps* = 25.000, *runs* = 50.

The results obtained were the ones expected and very similar to the previous one: lower thresholds make cooperation easier because agents get a bigger reward by contributing smaller amounts of their wealth.

It is important to analyse the previous results. One of the most surprising results that we obtained is that it seems that wealth inequality is helping agents to cooperate, to accomplish the game's threshold. This result may be contra-intuitive, but it makes sense once we think about how our public game works.

Without wealth inequality, a large fraction of the agents involved in each game would need to contribute in order to win a game. However, with wealth inequality, a significant portion of the games total available wealth (sum of wealth of the agents involved in a particular game) in concentrated in just a few agents, this means that this time we only need a much smaller number of agents (only the rich) to contribute for the total of the common pot contributions is equal or bigger than the threshold.
This also explains why; poor agents do not contribute at all when the wealth equality factor is lower than 0.9: they do not need to contribute. Their contributions have had a minuscule impact on the game.
However, humans are very sensitive to fairness; there are several experiments where humans reject offers just because they are not fair, even if that decision is far from being the optimal choice [21].
This behaviour of having only the rich to contribute, would not be easily accepted in the real-world. On climate submits, all countries are required to contribute to the common good, but with these experiments, we showed that a big portion of the impact could be made only by few, rich and most polluting countries.

One may ask, what would happen if we changed the proportion of rich and poor (currently at 20% rich and 80% poor). If we increased the number of rich agents in each game, that would be the equivalent to increasing the wealth equality factor, because the wealth would be distributed between more agents; the opposite happens if we decrease the number of rich agents. The wealth would be more concentrated in fewer agents, so it would be the equivalent to decreasing the wealth equality factor.
We can conclude that the study of varying the value of the rich/poor proportion, would lead to the same conclusions achieved in this chapter.

5.1.3 Contributions Diversity

Other interesting discussing is to know what happens to the learning behaviour of the agents when they are in an unstable environment, this means, in an environment where the percentage of group achievement is either ~0% or ~100%. In Fig. 17 we can see that many points verify these conditions, we picked one at random. We chose N = 8 (green line at Fig. 17) and risk = 0.5 using groups with a fixed structure (20% rich and 80% poor).
In Fig. 19 we can observe a different pattern from previous sections. In these charts, we can see that the contributions from the rich are very unstable, with a very high degree of variance and we see another contra-intuitive result: the more similar the wealth of the agents is (values of the wealth equality factor closer to 1) the lower the percentage of group achievement and the agents contributions.
The reason behind this is the same discussed in section 5.1.2. When the wealth equality factor is closer to 1, the agents are forced to have a larger degree of cooperation, that is, all agents need to contribute a bigger percentage of their wealth to win the game. When this does not happen (agents do

not contribute more with bigger wealth equality factor levels) than, cooperation is not sustained and the percentage of group achievement drops very rapidly.

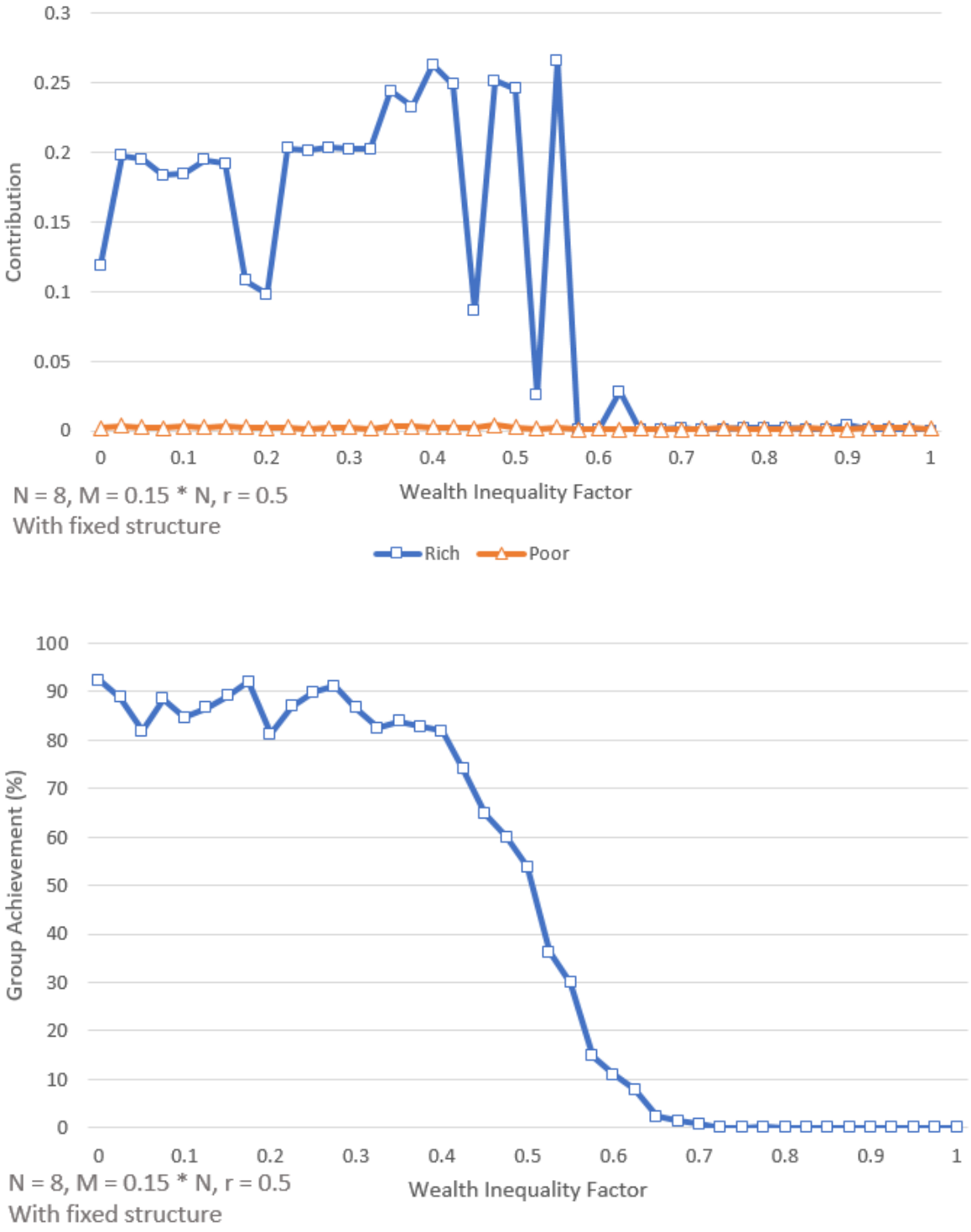


Fig. 19 Average contribution per agent (top pane) and percentage of group achievement (bottom pane) for groups with Wealth Inequality (structured groups) in an unstable environment. Parameters: N = 8, M = 1.2, *wealth equality factor* = 0.5, γ = 0.01, ϵ = 0.01, error rate = 0, V = 20, agents (Z) = 100, *risk* = 0.5, *maximum time-steps* = 25.000, *runs* = 50.

5.2 Uncertain Threshold

Until now we are only using a fixed threshold that is constant during all games in each simulation. However, we could change this and use a variable random threshold within a range of possible values. This is interesting to study because, in the real world, the same country can have multiple interactions with other nations, each of them with different objectives with various levels of difficulty. The reduction effort that will prevent climate disaster is not fully determined, as well.

To recreate this scenario, we will introduce a new variable into our system called “acceptance rule variance” (α).

At the beginning of each game, we will now pick the threshold taking into account the base threshold and the α. The bigger the α, the bigger will be the variance of the calculated random threshold.

The formula to calculate the range of values possible for the threshold during the simulation is the following:

$$minRange = M - \left(\alpha\left(N.\frac{1}{V}\right)\right) \quad (12)$$

$$maxRange = M + \left(\alpha\left(N.\frac{1}{V}\right)\right) \quad (13)$$

Where *M* is the base threshold, α is the acceptance rule variance, *N* is the group size, *V* is the length of the Propensity Vector of the agents, the *minRange* is the lowest possible value for the threshold in each game and the *maxRange* is the opposite, the maximum possible value for the threshold in each game.

In each game, we will pick a number between [minRange, maxRange] in steps calculated by the formula:

$$step = N.\frac{1}{V} \quad (14)$$

These three formulas ensure that each game has a threshold with the same granularity that the agent’s contributions, for example, it does not make sense to have a threshold of 0.175 if the agents can only contribute amounts multiple of 0.05. We also want to ensure that each agent has the opportunity to contribute equal amounts, that is the way we take into account the size group (*N*).

Notice that when $\alpha = 0$, we are in a scenario where we have a fixed threshold. Also, it is important to ensure that $\alpha \geq 0$, $minRange \geq 0$, $maxRange \geq 0$ and $step > 0$.

The algorithm we use to pick the threshold for each game is very simple:

```
1 function variableThreshold (base, arv, step);
  Input  : base: (double) base threshold (M); arv: (int) variance level;
           step: (double) threshold accepted steps
  Output: (double) new threshold for the current game
2 if arv = 0 or step = 0 then
3 |   return base;
4 else
5 |   randomNumber ← rand(−arv, arv + 1);
6 |   return base + (randomNumber ∗ step);
7 end
```

Algorithm 2 Pseudo-code of the algorithm which outputs the threshold for each game in the range of [minRange, maxRange]. For the input, we have the base threshold, the α (acceptance rule variance) and the step (calculated per formula 14). The function *rand(min, max)* used, outputs a random integer number between [min, max[range.

The Algorithm 2 ensures that is compatible with previous versions of the program without variable thresholds. By inputting the $\alpha = 0$, the program will behave exactly the same as before.

For our experiments, we tested the same parameters with 4 different acceptance rule variance values: 0 (same as not having any variance, it will be our baseline for comparison), 1, 2 and 3.

We also used the same parameters used in previous experiments so we can obtain comparable results.

In Fig. 20 we can see that the bigger the variance, the less likely is the cooperation to emerge. When the variance is absent, we start to see cooperation with values of the risk of around 0.525 but with α = 1, then we only see the emergence of cooperation around risk = 0.675, and for α = 2, this value is even bigger, around 0.875. When α is equal or larger than 3, we cannot see any cooperation at all.

This happens because agents have a harder time learning. Our learning algorithm used in all agents do not have any state, that means that it does not know any information about their environments, including what is the current threshold in the game that they are playing. This lack of information forces the agents to contribute more to the common pot that would normally be required to achieve cooperation in order to avoid the risk of losing their wealth, but only to a determined value. When the threshold varies too much, agents prefer not to contribute at all, then making contributions that have a significant change not to be enough to cover the current threshold, even if faced with values of risk very close or even 1.

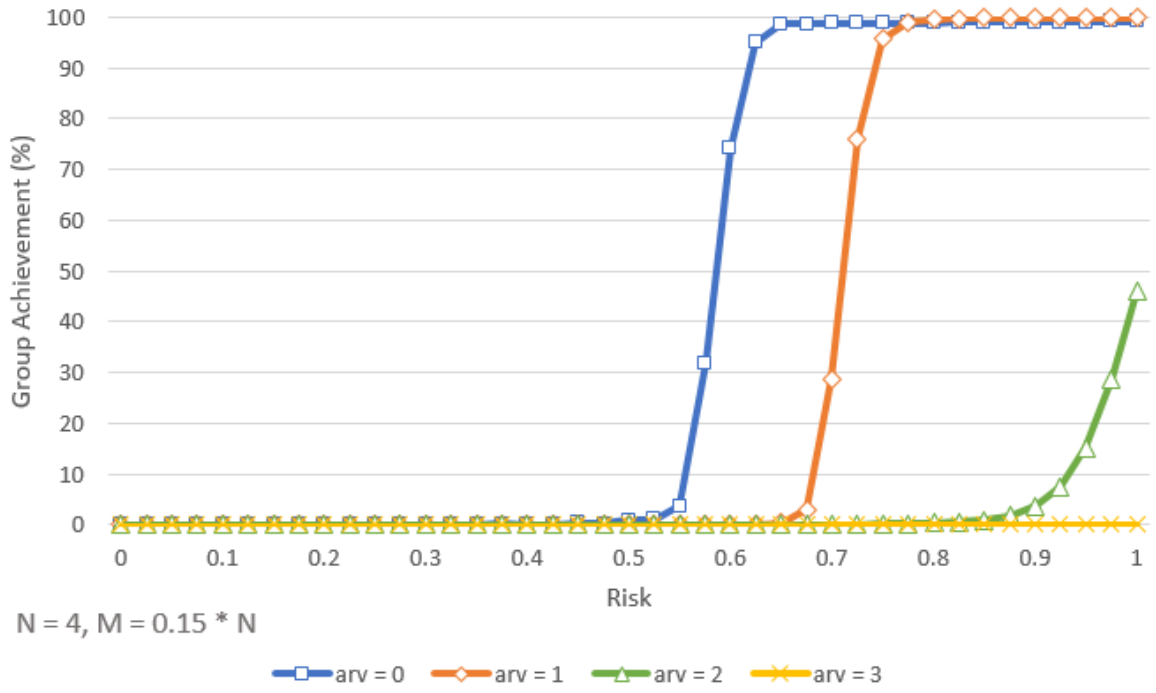


Fig. 20 Variable Threshold. Results obtaining be experimenting with different levels of uncertainty and variance in the threshold *M.* In each new game, we would pick a new threshold within the accepted range based on the α value. This plot shows that the bigger the variance, the more difficult is the cooperation to emerge. Parameters: *N* = 4, *M (base threshold)* = 0.6, *wealth equality factor* = 1 (without wealth inequality), γ = 0.01, ϵ = 0.01, error rate = 0, *V* = 20, agents (*Z*) = 100, *maximum time-steps* = 25.000, *runs* = 50, α = 0 (blue line), α = 1 (orange line), α = 2 (green line), α = 3 (yellow line).

We can conclude that consistent thresholds in all games are the best way to ensure that cooperation has the biggest possible change to emerge. However, in the real-world sometimes this cannot be guaranteed.

In these cases, the best way to get the agents to cooperate is to have a small degree of variance, if possible, to allow the agents to learn better what are the contributions needed to achieve the game's threshold. This could be attained resorting to insurance products, for instance.

Bigger variance in the threshold means that agents will have a much harder job to learn what values they should contribute, that is why the percentage of group achievement drops when the games are subjected to bigger values of α.

Chapter 6. Conclusion

In this work, we tackled the problem of “How can agents learn to cooperate with other agents without communication and given a goal that they do not know?”. The results were impressive.

We used for this work the well-known Roth-Erev reinforcement learning algorithm in a multi-agent environment and allowed the agents to learn by trial-and-error in several hundreds of games.

We found that not only agents demonstrated a remarkable capacity to cooperate when provided with the right incentives, but they do it (most of times) in a fair way, where all agents contribute with the same value with a very low degree of variance. The only exception for this behaviour is when we do not quite provide enough incentives to all agents to cooperate, neither is the risk small enough to not contribute at all.

When we added a new layer of complexity to our system, like with Wealth Inequality, we saw that when rich are much richer than poor agents, it facilitates the emergence of cooperation by shifting the responsibility for achieving the goal (threshold *M*) towards only a handful of agents. Our rational rich agents chose to cooperate among themselves and let the poor agents rip the benefits of their actions.

However, when we faced the agents to an uncertain threshold, results were poor. Cooperation was more difficult to achieve, this can be due to the fact that our reinforcement learning algorithm works best when that environment is constant. Agents had more difficulty to learn in these settings making the emergence of cooperation more difficult, but still possible if this variance were small.

But some results are common to all settings and experiments: cooperation in smaller groups (small *N*) and small threshold (*M*) is easier. This result is intuitive, is much easier to reach a consensus in small groups, especially when the goals are not very ambitious. A more exhaustive analysis of all results was already made in chapter 0.

Below, we discuss some of the main conclusions we can take from our experiments.

6.1 Global Results

In this subsection, we try to inter-connect all the previous results and provide a general view of our main findings.

From analysing all the experiments and their respective data we can extract the following results regarding the cooperation behaviour of the agents:

1. Cooperation is harder to achieve in larger groups
2. The higher the group threshold (*M*), the more challenging is to accomplish cooperation.
3. Uncertainty in the group threshold (*M*) prevents cooperation.
4. Wealth inequality promotes cooperation, by requiring fewer players to achieve the threshold. We also observed evidence that we should prefer stable negotiation groups over time, composed by individuals with consistent (over different groups) wealth levels.

By making a deeper analysis of how the learning system works, we know that these results all depend on what the agents learned using the Roth-Erev reinforcement learning algorithm implemented in this project.

In the subsection 4.2.1, we observed that the agents learned to converge to the same contribution (with very few exceptions). Regarding the learning behavior of the agents, we can thus learn that:

1. The propensity of the agents converges to a single value; for fixed conditions, agents consistently contribute the same after the initial learning period.
2. When cooperating, agents learn to play the fair Nash Equilibrium (contribution = *m* = *M*/*N*)
3. For low-risk values, it is fundamental that agents start playing close to their fair equilibrium (*m*); otherwise, cooperation does not emerge.

These three learning behaviours have several profound implications both on stability and contributions of the agents.

Most of times, there are two possible sets of contributions that form equilibria: contributing zero or any combination of contributions whose sum is *M* ($\sum_{i=1}^{N} c_i = M$, where c_i is the contribution of agent *i*). This set of possible contributions that satisfy the above formula can be very big even for the simplest game scenarios. For example, a game with two agents (*N* = 2), with a threshold of 0.6 (M = 0.6) and with a propensity vector with 11 possible playable strategies, we have the following possible combinations that satisfy the $\sum_{i=1}^{N} c_i = M$condition:

Table 5 Possible strategies where the sum of contributions is equal to the threshold. Written in red is the fair Nash Equilibrium. Parameters: *N* = 2, *M* = 0.6, *propensity vector size* = 11.

Agent A	0.6	0.5	0.4	**0.3**	0.2	0.1	0
Agent B	0	0.1	0.2	**0.3**	0.4	0.5	0.6
Sum	**0.6**	**0.6**	**0.6**	**0.6**	**0.6**	**0.6**	**0.6**

We can see that in our small scenario shown in **Erro! A origem da referência não foi encontrada.** we already have seven combinations. This number only gets bigger with more complex game environments.

One obvious question is why we see the emergence of both of these contributions equilibria? Moreover, most importantly, why in our practical experiments agents choose the fair Nash Equilibrium, where all agents decided to contribute the same endowment to the common pot?

In both equilibria (contributing zero or any combination of contributions whose sum is *M*), no agent has an incentive to change their strategy because it will not improve their payoff (Nash Equilibrium). This lack of incentives is the reason why agents have shown to have such stable contribution history after reaching the Nash Equilibrium, even when using exploratory methods, like the *error rate* or the *local exploration*. Likewise, any combination of strategies that respects $\sum_{i=1}^{N} c_i = M$ rule is Pareto Optimal, meaning that it is impossible for an agent to decrease their strategy without decreasing the payoff of the other agents. To better understand why agents, have a natural tendency to adopt the fair Nash Equilibrium contribution value, we propose a simple thought experiment. Let's say that we have a group of two agents, both contribution 0.5 for the pot, where the threshold (*M*) is 0.6, their payoff is given by $1 - c_i$ if the sum of contributions is equal or bigger than *M* or 0 otherwise (in this scenario we will ignore risk and other factors). Agents have 11 possible strategies to play (0, 0.1, 0.2, …, 1). Both agents start by getting a payoff of 0.5, but they can improve this value if both decrease their contributions, but still, their sum be equal or bigger than *M*. Because of the local exploration, they get an incentive to play strategies similar to the one previously played, so agents eventually try to play the strategy 0.4, getting a bigger payoff of 0.6 where they reinforce more strongly their learning. Note that this only happens because the 0.5 strategy is not a Nash Equilibrium, an agent can indeed change their play independently of the other agent and still get a bigger payoff.

This process repeats itself until both agents get to the point where either can decrease their strategy without decreasing their payoff. This stage in our experiment is where both agents' contributions are 0.3, which is a Nash Equilibrium. From this simple experiment, we can also extract an interesting conclusion. Both agents got to the Nash Equilibrium in a distributed way, without prior planning or communication between them by only using a basic learning algorithm. This learning process demonstrated in where we can see agents slowly but steadily converging to the Nash Equilibrium and in Fig. 8 where we see that the vast majority of the agents by the end of the simulation converged to the Nash Equilibrium.

The only exception, in when we subject the agents to incentives that are low enough not to see full cooperation but high enough not to have all agents not contributing (this scenario was analyzed in 4.2.2). In this case, agents tend to learn a bigger set of strategies. Agents tend to contribute to the common pot bigger values than the usual fair value because some agents in our experiments chose not to contribute, so contributing agents had to contribute more to compensate these losses of contributions. This is a remarkable result because it shows us that some agents prefer to pay more

(even if is not fair) to avoid being subject to the risk of losing all their wealth, even if it means to allow some agents to free-ride and take advantage of their efforts.

In the next chapter, we will conclude this work. We will discuss what contributions we made, a very general review of the results obtained and some improvements that can be made in future work.

6.2 Contributions

Below we discuss what we consider to be the main contributions of the presented work:

- **Novel Model**: Our approach to our problem was based on several papers [3, 21], but we used Reinforcement Learning.
- **Development of efficient code base**: The code used in the development of this work, was made from scratch, in order to make the necessary code as fast and reusable as possible inside the scope of our work.
- **Study of the impact several external factors on cooperation**: We studied how Wealth Inequality, uncertain thresholds and the error rate affected group cooperation.
- **Study of fairness in games**: We showed that agents, given when many possible combinations of actions available, many times opt to choose between contributing the egalitarian fair value, or if they don't have enough incentives to do so, chose not to contribute at all.
- **Learning process**: Significant effort was done to analyse not only what agents did, but also how agents learned with time.
- **Impact of risk in collective dilemmas**: We analysed extensively and confirmed that *risk* has a significant influence on cooperation.
- **Impact of group size in collective dilemmas**: Similar to *risk*, the group size (*N*) also has a considerable impact on group cooperation.
- **Uncertainty**: The use of an uncertain threshold showed to be contra-productive for the learning process of the agents, causing cooperation to be more difficult to emerge.
- **Wealth Inequality**: Our experiments showed that if rich agents contribute by themselves to the common pot, they alone are enough to reach much of our thresholds, making cooperation easier to achieve.

6.3 Future Work

Unfortunately, all works must come to an end, even when there are still much to learn. *Scott Adam* (author of the popular *Dilbert* comics) once said:

> *"Normal people … believe that if it ain't broke, don't fix it. Engineers believe that if it ain't broke, it doesn't have enough features yet"*

As such, we suggest some improvements to our work in the future:

- **Interaction Mechanisms**: In our simulation, on each game, we pick *N* random agents, but these groups can be improved. It would be interesting to know what happened if we forced the agents to always interact with the same agents. What if some elements in the group always defected? What if some agents were extremely generous, how this would affect the cooperation and the contributions of each group?
- **Provide more information to agents**: In this work, agents had no information about their environment. They learned by trial-and-error. Providing the agents with more and better information about their situation may improve cooperation.
- **More complex learning algorithm**: We used a very simple learning algorithm, there are more sophisticated algorithms available like *Q-learning* that are slower, but it could allow observing more complex behaviour dynamics in the agents. However is not guarantee that these complex algorithms will successfully model human behaviour.
- **More complex reward function**: Our game has very few components: it has the risk, the threshold, the forgetting rate, the local exploration rate and the error rate. However, we can add almost an infinite amount of rewards and incentives to the agents. For example, we considered that all agents faced the same value of risk of losing their wealth, but in the context of climate change this is not true. Some countries, like small islands, face a much risk of being affected but the rising sea levels than large continental countries, as such, we could introduce a new risk value that would be different to each agent.

6.4 Closing Remarks

We can draw many interesting conclusions from this work, even if the reader does not believe in climate change or that there are more concerting subjects.

Our simulations are abstract enough to its results can help us make decisions in other fields that require cooperation for a common goal, whatever it be at a negotiation table, or when discussing the fishing quotas for the next year.

We live in the world ever more connect and complex, cooperation with our peers is critical in order to achieve any goal that is sufficiently big.

We want to end this work with a quote from the former US president, *Bill Clinton*:

> *"We all do better when we work together. Our differences do matter, but our common humanity matters more".*

Bibliography

1. Milinski, M., Sommerfeld, R.D., Krambeck, H.-J., Reed, F. a, Marotzke, J.: The collective-risk social dilemma and the prevention of simulated dangerous climate change. Proc. Natl. Acad. Sci. U. S. A. 105, 2291–2294 (2008).
2. Hardin, G.: The Tragedy of the Commons∗. J. Nat. Resour. Policy Res. 1, 243–253 (2009).
3. Santos, F.C., Pacheco, J.M.: Risk of collective failure provides an escape from the tragedy of the commons. Proc. Natl. Acad. Sci. U. S. A. 108, 10421–10425 (2011).
4. Von Neumann, J., Morgenstern, O.: Theory of games and economic behavior. Princeton University Press, Princeton (1953).
5. Sigmund, K., Nowak, M.A.: Evolutionary game theory. Curr. Biol. 9, R503–R505 (1999).
6. Fudenberg, D., Levine, D.K.: The Theory of Learning in Games. MIT Press (1998).
7. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. (1998).
8. Rand, D.G., Nowak, M.A.: Human cooperation. Trends Cogn. Sci. 17, 413–425 (2013).
9. Dawes, R.M.: Social Dilemmas. Annu. Rev. Psychol. 31, 169–193 (1980).
10. Wooldridge, M.: Introduction to Multiagent Systems. Inf. Retr. Boston. 30, 152–183 (2002).
11. Shoham, Y., Powers, R., Grenager, T.: If multi-agent learning is the answer, what is the question? Artif. Intell. 171, 365–377 (2007).
12. Genesereth, M.R., Ginsburg, M., Rosenschein, J.S.: Cooperation without Communication. In: Proceedings of National Conference on Artificial Intelligence. pp. 51–57 (1986).
13. Jennings, N.R., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development. Auton. Agent. Multi. Agent. Syst. 38, 7–38 (1998).
14. Nowak, M. a: Five Rules for the Evolution of Cooperation. Science (80-.). 314, 1560–1563 (2006).
15. Rand, D.G., Greene, J.D., Nowak, M. a.: Spontaneous giving and calculated greed. Nature. 489, 427–430 (2012).
16. Nowak, M.A., Sigmund, K.: The alternating prisoner's dilemma. J. Theor. Biol. 168, 219–226 (1994).
17. Fudenberg, D., Rand, D.G., Dreber, A.: Slow to anger and fast to forgive: Cooperation in an uncertain world, (2012).
18. Pinheiro, F.L., Vasconcelos, V. V., Santos, F.C., Pacheco, J.M.: Evolution of All-or-None Strategies in Repeated Public Goods Dilemmas. PLoS Comput. Biol. 10, (2014).
19. Wedekind, C., Milinski, M.: Cooperation through image scoring in humans. Science. 288, 850–2 (2000).
20. Milinski, M., Semmann, D., Krambeck, H.-J.: Reputation helps solve the "tragedy of the commons." Nature. 415, 424–426 (2002).
21. Santos, F.P., Santos, F.C.S., Melo, F.S., Paiva, A., Pacheco, J.M.: Dynamics of Fairness in

Groups of Autonomous Learning Agents. Springer (2016).
22. Van Segbroeck, S., De Jong, S., Nowe, a., Santos, F.C., Lenaerts, T.: Learning to coordinate in complex networks. Adapt. Behav. 18, 416–427 (2010).
23. Tan, J.H.W., Bolle, F.: Team competition and the public goods game. Econ. Lett. 96, 133–139 (2007).
24. Böhm, R., Rockenbach, B.: The Inter-Group Comparison - Intra-Group Cooperation Hypothesis: Comparisons between Groups Increase Efficiency in Public Goods Provision. PLoS One. 8, (2013).
25. Voors, M.J., Nillesen, E.E.M., Verwimp, P., Bulte, E.H., Lensink, R., Van Soest, D.P.: Violent conflict and behavior: A field experiment in Burundi, (2012).
26. Gneezy, A., Fessler, D.M.T.: Conflict, sticks and carrots: war increases prosocial punishments and rewards. Proc. Biol. Sci. 279, 219–23 (2012).
27. Hagel, K., Chakra, M.A., Bauer, B., Traulsen, A.: Which risk scenarios can drive the emergence of costly cooperation ? Sci. Rep. 6, 19269 (2016).
28. Bernoulli, D.: Exposition of a New Theory on the Measurement of Risk. Econometrica. 22, 23–36 (1954).
29. Wu, G.: Generalized expected utility theory: The rank-dependent model - Quiggin,J. J. Behav. Decis. Mak. 9, 229–230 (1996).
30. Vasconcelos, V. V., Santos, F.C., Pacheco, J.M.: A bottom-up institutional approach to cooperative governance of risky commons. Nat. Clim. Chang. 3, 797–801 (2013).
31. Roth, A.E., Erev, I.: Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. Games Econ. Behav. 8, 164–212 (1995).
32. Thorndike, E.L.: Animal intelligence: An experimental study of the associative processes in animals. Psychol. Rev. 2, 1–107 (1898).
33. Newell, A., Rosenbloom, P.S.: Mechanisms of skill acquisition and the law of practice. Cogn. Ski. their Acquis. 1, 1–55 (1981).
34. Vasconcelos, V. V, Santos, F.C., Pacheco, J.M., Levin, S. a: Climate policies under wealth inequality. Proc. Natl. Acad. Sci. U. S. A. 111, 2212–6 (2014).
35. Traulsen, A., Semmann, D., Sommerfeld, R.D., Krambeck, H.-J., Milinski, M.: Human strategy updating in evolutionary games. Proc. Natl. Acad. Sci. 107, 2962–2966 (2010).
36. Sequeira, P., Melo, F.S., Paiva, A.: Emergence of emotional appraisal signals in reinforcement learning agents. Auton. Agent. Multi. Agent. Syst. 29, 537–568 (2014).
37. Dreber, A., Nowak, M. a: Gambling for global goods. Proc. Natl. Acad. Sci. U. S. A. 105, 2261–2262 (2008).
38. Hasson, R., Löfgren, Å., Visser, M.: Climate change in a public goods game: Investment decision in mitigation versus adaptation. Ecol. Econ. 70, 331–338 (2010).
39. Erev, I., Roth, A.E.: Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria. Am. Econ. Rev. 88, 848–881 (1998).