DALARNA
UNIVERSITY

Degree Project

Dalarna University, Master in Microdata Analysis with emphasis on Business Intelligence

All for one one for All

Author: Bamshad Shirmohammadi
Supervisor: Professor Arend Hintze
Examiner: Professor Siril Yella
Subject/main field of study: Microdata Analysis
Course code: MI4002
Credits: 15 ECTS
Date of examination: January 20th 2021

Abstract

Imagine a school where everyone tries to help the weakest student, and there is no competition among them except for being nice and supportive! Or companies who, instead of competing with their customers, try to make the world a better place! Yes, it is too big of a goal, but every big move starts from a small step. The small step in my thesis is working on the question "how to make groups work together in a better way" applied to the world of artificial intelligence. To train virtual agents to successfully work together, one must overcome the same problems humans face, such as selfishness. While agents might not be selfish, we typically optimize or preferentially select them on their individual performance. Thus, the selfish agents would also win again. To study and remedy this problem, I made a game for multiple artificial agents competing with each other. I also devised different objective functions so that selfishness might not be the optimal strategy. Specifically, Darwin's theory of Evolution and the group selection mechanism provides a different mechanism to incentivize groups to cooperate. The idea is that when evolving agents under these conditions, the last generation will include only cooperating agents. I used a framework that includes an evolvable neural network for implementing the agents and a genetic algorithm working based on specific rewarding schemes. I found that different rewarding schemes result in different agent behavior. Specifically, when agents evolve in groups whose performance is measured by their weakest member, they not only evolve to become an effective group, but they receive rewards more equally.

Keywords:

Artificial intelligence, Neural Networks, Cooperation, Group Selection, and Evolution

Acknowledgment

First and foremost, I have to thank my research supervisor, Professor Arend Hintze. His assistance and dedicated involvement in every step throughout the process is highly appreciated and was very helpful in accomplishing my goals. I would like to thank you very much for your support and understanding.

Contents

1. Introduction

The state of the art machine learning technique to optimize neural network AI controllers (agents) is back-propagation and specifically deep learning. This method assumes that solutions and examples of such solutions are known. A different approach to this is evolutionary computation and genetic algorithms. Instead of rewarding proper responses to specific situations and hoping for proper generalization, a genetic algorithm uses an objective function. This objective function evaluates the total performance of a system and can select controllers that perform better. Over many generations, such a system optimizes controllers, finds more creative solutions [1], and most importantly, does not require knowledge about specific actions but only the desired outcome. This method works well when optimizing individual controllers as their objectives do not conflict with the objectives of other controllers. When agents need to interact or work in teams [2], the situation becomes much more complicated, and often the goals of the individual are contrary to the success of the group. Imagine self-driving cars. Optimizing them to reach their goal as fast as possible might induce negative side effects. How to overcome such issues? Group-level selection plays an important role in Evolution. Often individuals do not act alone but in groups. As such, they can achieve together more than alone. Collaborative hunting is one of those examples where individuals are not selected individually but as a group [3]. This group-level selection scheme typically pools the resources the group collected and redistributes them back equally, which implies that groups are evaluated by their average performance. What if this scheme is altered? Imagine a group is rewarded according to the performance of its best individual (all for one) or its worst performer (one for all)? In the all for one case, you would pool all resources on one individual, and in the case of the one for all, you would distribute the resources as fairly as possible. The big question is, which of the two groups collected the most resources in total? This research will use neuro-evolution, where agents are simulated in a virtual environment and controlled by so-called Markov Brains. Group-level selection regimes such as the ones described above will be tested with respect to their effect on individual and group level performance.

1.1. Background

Societies are depending on the collaboration of their members. This collaboration is the basis of the economy, health care system, and education, among others. Obviously, such institutions benefit from everyone contributing to them via taxes. However, since they are carried by society, individuals who for example, evade taxes can still benefit from them while also benefiting from the resources they did not contribute due to cheating.

To study this problem, social science, experimental economics, and game theory use the "Public Goods Game." In this game, the participants are given equal amounts of money. They then can decide to withhold this money (defect) or contribute (cooperate) into a public pot. The total contribution of all participants will be multiplied by a synergy factor. This synergy represents gains that can only be achieved by pooling resources. The now larger amount of money in the pot will then be divided equally among all the players - regardless of them having contributed or not in the first place. Considering the game rules, a player who is careful about the social benefits; will contribute money, and those who are selfish and only consider their personal benefits will not. The scenario that individuals only think about their gain exploiting the contribution of others is called "Tragedy of the Commons" [4].

The question is now, how can people be incentivized to be collaborative? In other words, how we can avoid the tragedy of the commons?

If we see it from a government perspective, we would consider incentives to motivate people or punishment to avoid selfish approaches. But for a broader historical perspective, we have to ask where cooperation within groups comes from initially? I find that Darwinian Evolution Theory also struggles to explain why organisms cooperate [5], regardless of organisms, including humans, evolved to cooperate. To discuss how natural selection is the solution for the collaboration problem, let's first examine how we humans in agriculture select the best genes for reproduction. In this example, a farmer, based on a certain criterion (such as the biggest plants) selects the desired ones for reproduction.

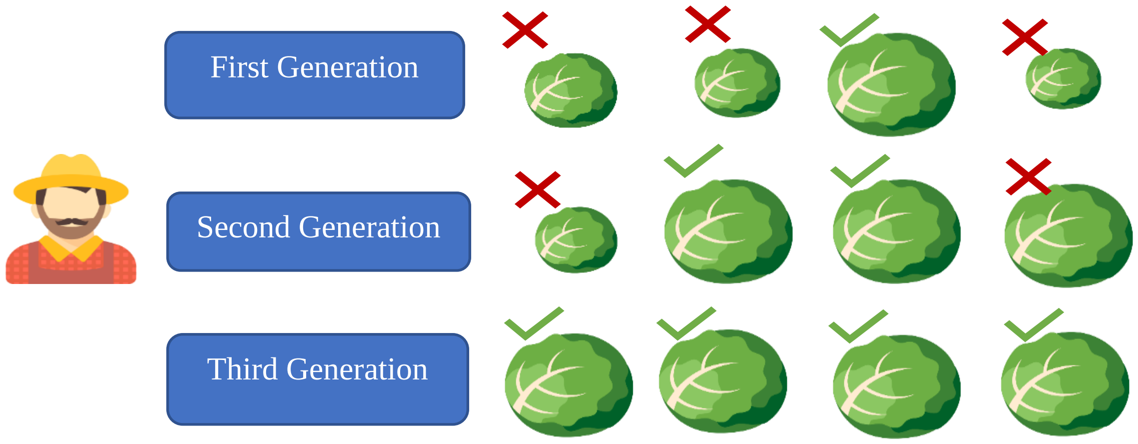


Figure 1. The human selection I have created this figure using free icons from [6] and [7].

The farmer harvests only big cabbages in the last generation by choosing only the biggest for reproducing (see figure 1). Nature does the same to all organisms and only lets some of them survive through history, and the rest will become extinct.

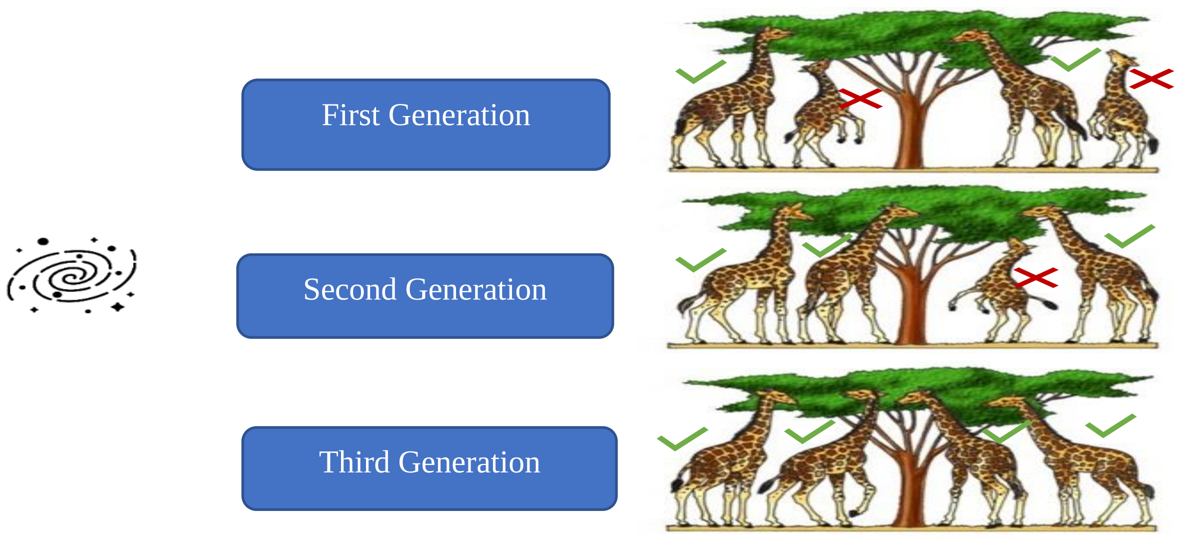


Figure 2. Natural selection. The images are downloaded from [8] and [6]

So, nature acts as the farmer and lets only the tall giraffes survive and produce offspring as they had an advantage in eating leaves on high trees. We call this *natural selection*. In the context of natural selection, the individuals' ability to survive and reproduce is called *fitness (w)*. You may ask how it comes that we have a variety of different species if only the fittest survives? Then all organisms should look like each other. The answer lies within the two other concepts of mutation and selection. Mutation means sudden and significant changes in the species genetics that results in offspring being different from their parents. Suppose we take a family of giraffes and divide them into two distinct islands which one of them has tall trees, and the other has short trees. If there were no mutation in nature, regardless of the islands' differences, the giraffes after generations would not be different while both groups are from the same family. The mutation causes significantly taller and shorter offspring in the two groups. In the short tree island, the smaller ones will survive and reproduce, and the opposite happens on the other island. Eventually, after a long time by natural selection and mutation, we will have two types of giraffes (short and tall) from the same routes. Consequently, the natural environment presents organisms with different ways to be the fittest, called niches. Those niches, together with inheritance, variation, and natural selection, leads to the biodiversity we observe today. Darwin's *Evolution* is, therefore, a mechanism composed of inheritance, variation (random mutation), and selection. It thus leads to an adaptation of organisms that fit their environment better. The proof of Darwin's Evolution is out of my thesis's scope, but, regardless of the dispute about this theorem, I have used the same concepts (inheritance, variation, and selection) in my experiments (which I will explain in the next sections).

However, evolution is inherently selfish, as its rewards happen on a short time scale. Reaping the rewards now, and making more offspring immediately, will always outcompete, saving rewards and reproducing later. Cooperation needs the investment of multiple partners, but from the tragedy of the commons, we know that the defectors fare better. Still, we find different biological mechanisms that allow cooperation to evolve. Among others (kin selection, green beard effect, reciprocity [5]) the one that most likely gave rise to multicellularity is group-level selection [9]. Here, not the individual reproduces, but the entire group benefits from the rewards the entire organism receives. Thus, the most prevalent choice for incentivizing cooperation is group-level selection.

Since Evolution already solved the problem of cooperation before, in this study, I would like to optimize groups of agents (robots) using a genetic algorithm to cooperate with each other. These agents are controlled by evolvable Neural Networks, specifically Markov Brains [10]. The MABE (Modular Agent Based Evolution) Framework [11] to run computational experiments has been developed before, and group-level selection reward functions have been tested before. I will build in these systems and test different forms of group-level selection, as well as new payoff schemes seeking to improve cooperation. This will be compared to none group-level selection.

Theoretically, all of the above can be implemented from scratch. There are also various libraries available that implement genetic algorithms. One of them is MABE, which not only has been well tested but also allows the evolution of agent controllers, such as neural networks of Markov Brains. Previous exploratory work has been carried out using MABE, testing the minimal and maximal rewarding schemes on more complex environments. Unfortunately, that work did not result in a publication. Too many technical and methodological difficulties were observed earlier, suggesting a simplification of the approach as it has been done here. Thus, it was easiest to build on that background by using MABE while also continuing the research without adding unnecessary overhead work. However, results should generalize to all other types of implementations and are not specific to MABE.

1.2. Purpose

This research seeks to improve the way we train groups of AI controllers (agents) to perform better individually and in teams at the same time. While this is a basic research question in the optimization of neural networks using genetic algorithms, it has direct applications to robotics and other autonomous AI decision making systems that need to work in groups.

As explained in the introduction, the thesis is in the sequence of other works using Markov's brain. Based on the literature review on other works in this chain of researches and also the other similar studies, I decided to work on the impact of incentives or rewarding schemes on team working of the AI controllers. According to the literature review, group-level selection has been shown to improve cooperation within groups of agents. Here I introduce a new set of fitness criteria where the effort of the group is not measured by the average performance of the group but instead by either the worst performer or the best performer. This work on the *minimum* and *maximum* reward schemes fills a particular *research gap* as it has not been tested before.

1.3. Literature review

I must mention that I have selected (and not written) the thesis proposal, and therefore, before starting the literature review, I was aware of the thesis objective that is based on the identified research gap by the supervisor. Finding similar works in this area and learning their methods is the main aim of my literature review. There is a huge number of studies in this field. I have selected those that satisfy these criteria: a) simulate organisms using artificial agents, b) include at least one concept related to task accomplishment c) have a new notion from other works.

The first research I want to introduce is [2]; they have simulated hierarchy among social animals using animats controlled by evolvable Markov Gate networks (MGN). The artificial animals are inside an area covered by walls and have a door to leave. These artificial animals' performance has been evaluated, and based on their observations; the best bots leave the room later than others. In other words, they have identified staying in the area as a dominant strategy. Working on the effect of different reward schemes on the formation of hierarchies is one of their future works, which is very near to my thesis's main objective, which I have explained in the previous section. Integrating disparate information sources by biological organisms is another aspect in this area that is investigated in [12]. In [12], information integration is used in terms of navigation, and it has been observed that the artificial agents' fitness increase by its (information integration) enhancement. In relation to the previous work, in [13] they have studied the integration of sensor inputs and memory to solve the tasks by artificial agents. For that study, they have created a 'Tetris-like' game for their adaptive logic-gate networks animats. Based on the 'Tetris-like' game results, increment in the agents' integrated conceptual information depends on the complexity of the environment. After the hierarchy among social animals and integrating disparate information sources by biological organisms, swarming behaviors in animals is the third concept from this world which is simulated in [14]. The authors of [14] have used Markov Network (MN) for their agents' controller. They "demonstrated that predator confusion provides a sufficient selective advantage for prey to evolve swarming behavior in a digital evolutionary model." [14] Moreover, they have proposed a new way to test hypotheses in terms of the evolution of swarming behavior in their work. The other important point is hazards in the environment, which is implemented in [15]. They have also highlighted the role of species' physiology in understanding behavior and how the environment encourages sensory systems development [15]. So far, in all the works, the agents face a problem and then try to solve it, whereas [16] propose a learning model in which agents can project them into future situations. "Projective simulation is based on a random walk through a network of clips, which are elementary patches of episodic memory." [16]

In all the mentioned works, multiple artificial agents do a task in a specific environment. At this point, the question is if the problems for these simple agents in their simple environments are similar to our real-world problems. To find an answer to this question, let's consider the humans' problems as the most complicated organisms in the world (as far as I know!). In [17], they have reviewed five models of intragroup conflict in management studies in which the groups are either a group of people or organizations. In their review of models, the conflict between individuals' payoff and the total payoff is discussed as one of the main concerns. [18] introduced the usage of incentives to address the conflict between individual benefits and welfare. So, even in humans as the most complicated organisms, the main problem is selfish behaviors, which we can solve using proper incentives. Based on the literature review similar to the technical trials such as [2] and [12], I am going to create a game for artificial agents, and I will test if the usage of incentives can train them to both be successful individually and in teams as this method works for humans [18]. So, here the problem and solution are similar to what humans experience, and we should enhance this sequence of research, so eventually, we can use the agents in real-life problems such as self-driving cars.

2. Material and Methods

This research used the MABE (C++ Modular Agent Based Evolution Framework [11]) to implement virtual test environments. Agents were controlled using Markov Brains [10], which are a particular evolvable type of neural network. After replicating evolutionary experiments were performed, data was analyzed and visualized.

To train groups of AI controllers (agents) to perform better individually and in teams at the same time, I have defined and implemented a game for a group of agents. They do the game under different conditions (rewarding schemes and type of groups), and I have evaluated their performance according to their own payoff and their team payoff. Based on the comparison of the agents' performance, I have confirmed which reward scheme allows them to evolve good teamwork behavior or to perform well individually. I will explain the game's definition, how I have trained the agents, and how I have evaluated them in the rest of the document.

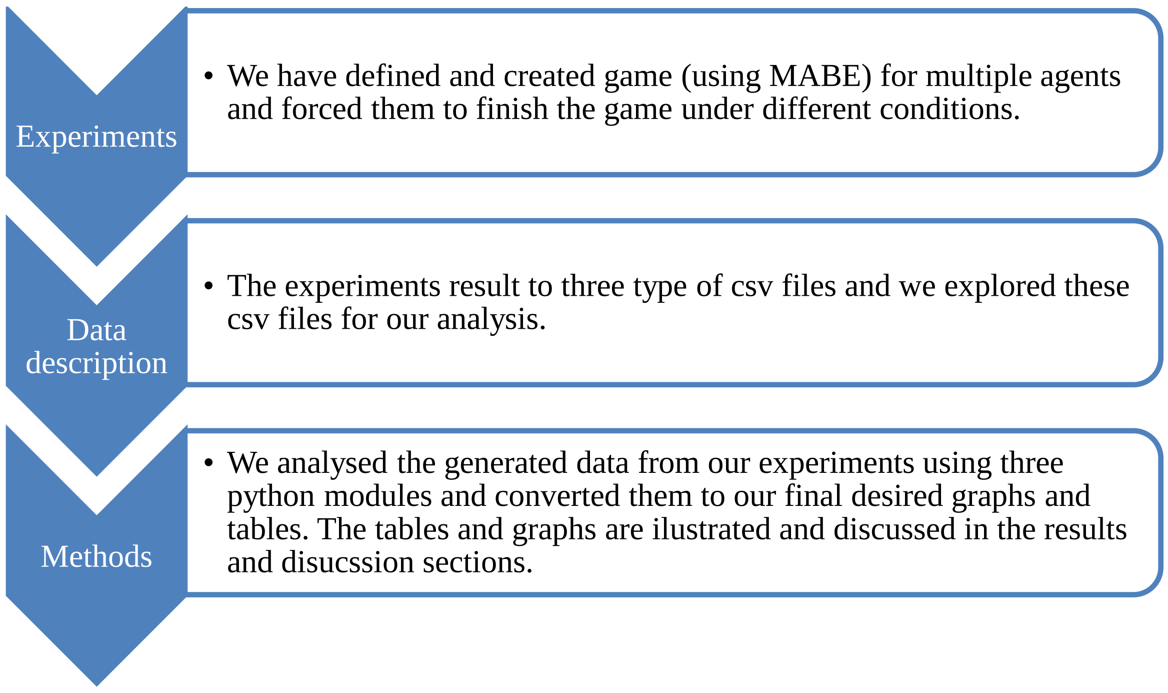

Experiments
• We have defined and created game (using MABE) for multiple agents and forced them to finish the game under different conditions.
Data description
• The experiments result to three type of csv files and we explored these csv files for our analysis.
Methods
• We analysed the generated data from our experiments using three python modules and converted them to our final desired graphs and tables. The tables and graphs are ilustrated and discussed in the results and disucssion sections.

Figure 3. Overview of the research methodology

2.1. Experiments

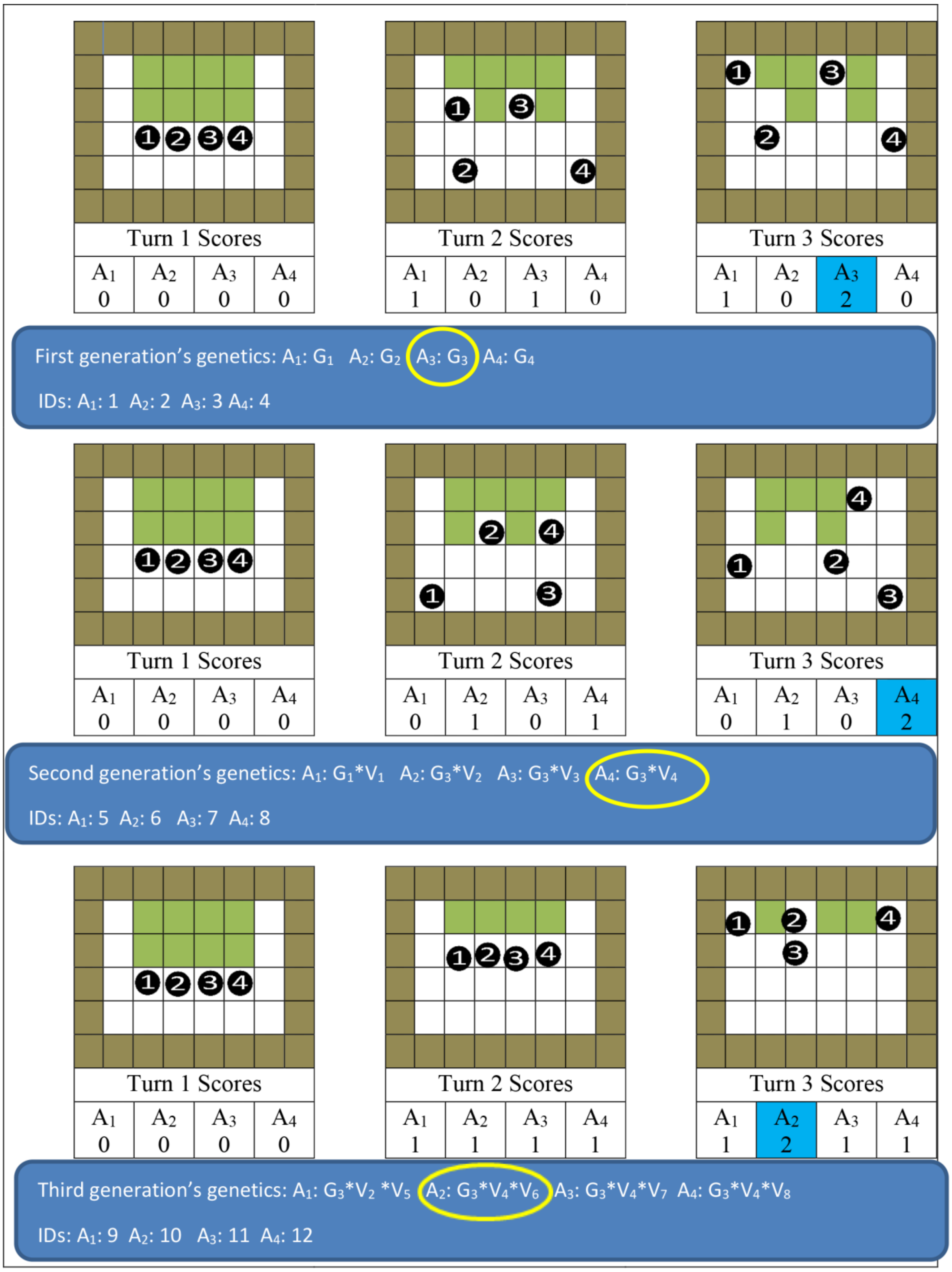
2.1.1. Game definition

In this simple game, four agents are tasked with collecting food in a confined space. Agents are placed in the four corners of an area surrounded by a wall that prevents agents from leaving this space. Agents can then sense their environments, specifically whatever is on the tiles in front of them. This sensor can distinguish empty tiles, walls, agents, and food items. They also have a sensor informing them about the total number of food collected, as well as three auditorial sensors that tell them if any of the other agents beeped in the last round. Each of these sensors is mapped to specific other agents. The auditory sensors allow for communication between the agents. Once sensor information is obtained, agents can perform computations defined by their Markov Brain. These computations lead to several possible actions that each agent can take: move forward, turn left or right, and if food has been previously collected, either drop that food on an empty tile or if an agent is in front of them, they hand the food to that agent. This computational process Markov Brains perform can be imagined like a recurrent neural network that received inputs in the first layer and then performs a forward pass so that outputs are computed. Agents can, in addition to their movements, also utter a beeping signal as a means to communicate something.

Food is distributed over the entire area of 16x16 tiles, which includes the walls. Then 50 random tiles are cleared of food to generate a less monotonous landscape. Agents are allowed to roam for 500 updates. After that, the amount of food agents collected is used in the different reward schemes to assign fitness such that the genetic algorithm can perform its selection routine on the population.

When a new generation is formed from the last, all new agents can experience mutations, deletions, and gene duplications, which can lead to changes in the Markov Brains. Over many generations, this process of evaluating agents, selecting better ones based on their fitness, and mutating them randomly evolves their strategy to perform more optimally.

A more complex environment is easy to imagine; however, as this work is a proof of concepts for the Minimal and Maximal rewarding scheme, as well as clonal group selection, I chose this simple environment. The MABE [11] environment, as well as the necessary changes to implement this world has been done before the project started by the members of the Hintze-lab (Clifford Bohm, Jory Schossau, and Arend Hintze).



Turn 1 Scores			
A_1	A_2	A_3	A_4
0	0	0	0

Turn 2 Scores			
A_1	A_2	A_3	A_4
1	0	1	0

Turn 3 Scores			
A_1	A_2	A_3	A_4
1	0	2	0

First generation's genetics: A_1: G_1 A_2: G_2 A_3: G_3 A_4: G_4

IDs: A_1: 1 A_2: 2 A_3: 3 A_4: 4

Turn 1 Scores			
A_1	A_2	A_3	A_4
0	0	0	0

Turn 2 Scores			
A_1	A_2	A_3	A_4
0	1	0	1

Turn 3 Scores			
A_1	A_2	A_3	A_4
0	1	0	2

Second generation's genetics: A_1: G_1*V_1 A_2: G_3*V_2 A_3: G_3*V_3 A_4: G_3*V_4

IDs: A_1: 5 A_2: 6 A_3: 7 A_4: 8

Turn 1 Scores			
A_1	A_2	A_3	A_4
0	0	0	0

Turn 2 Scores			
A_1	A_2	A_3	A_4
1	1	1	1

Turn 3 Scores			
A_1	A_2	A_3	A_4
1	2	1	1

Third generation's genetics: A_1: G_3*V_2 *V_5 A_2: G_3*V_4*V_6 A_3: G_3*V_4*V_7 A_4: G_3*V_4*V_8

IDs: A_1: 9 A_2: 10 A_3: 11 A_4: 12

Figure 4. Illustration of a simple example of the game. The black circles are the artificial organisms (agents), green squares are the grass, white squares are empty tiles, and brown ones are walls. This example game has only three turns and three generations. The agent who could gain the most number of grass is highlighted with blue in each generation. I will explain the three concepts of inheritance, variation, and selection in the next chapters.

2.1.2. Game implementation

The MABE computational modeling framework allows evolutionary experiments to be set up and conducted with ease. MABE consists of several modular components that have been extensively tested before, which brings high confidence in their functionality. The one component that needed to be adapted is the World-Class, which defines the environment agents have to perform in, and which measures their fitness. All other components, such as Markov Brains, their genome, the selection mechanism, the lineage, and data tracking, and the parameter system, remained untouched. For each condition, 100 independent replicate experiments were run, using the UPPMAX high performance computing center – "UPPMAX Grant for Neuro-Evolution Arend Hintze."

I. Area or world:

It is the place where the experiments will be done. This area is divided into tiles, and each tile can be set as empty, grass, and wall. The agents can only move in empty or grass tiles. The agents can collect and either consume it directly, increasing its count, or later hand it to another agent or put it in an empty tile for others. In the end, the amount of grass (energy) each agent collected will be used to determine the score of the group. If they go into a grass tile, it will be converted to an empty tile.

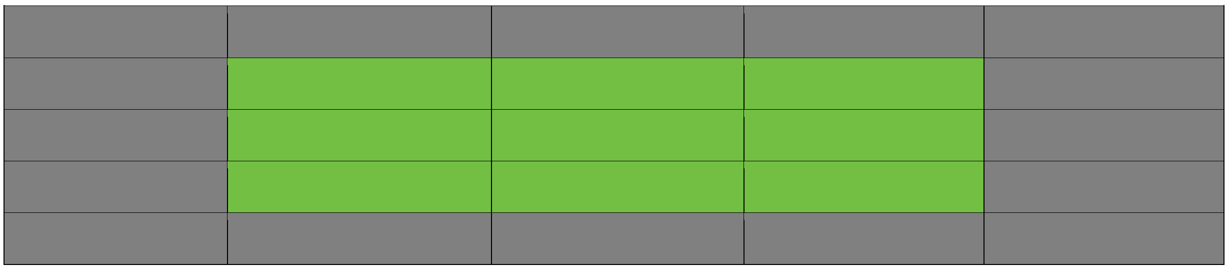


Figure 5. An area example. In this example, the gray tiles are walls, and the green tiles are grass.

From a technical point of view, the area is an object of the Area class created in ExampleWorld.hpp (world/ExampleWorld folder). The Area class is defined in area.cpp (core folder). I have set the area properties in the ExampleWorld.cpp (world/ExampleWorld folder).

The Area class has a constructor, which has three arguments that are x, y (as the dimensions), and startTile, which is for selecting the initial filling type (such as grass) of them. To specify the tiles type individually, I can use set() function in the Area class. X value and Y value are the first two arguments to select the tile position, and the third argument sets the tile's type.

The following table explains how to set the different attributes of the area:

Function name	Description
Area::Area(int _xDim,int _yDim,int startTile)	It is the constructor of the area class that set the area size and initial tiles type. _xDim and _yDim can be a positive number, and the startTile is defined as

	follow: 0 = empty 1 = grass 2 = agent 3 = wall
Area::set(int x,int y, int what)	Using this function, I can set each tile's type individually. x and y identify the tile position and type (grass, empty, wall, and agent)

Table 1. The main functions to set the area attributes

II. Agents:

Agents are the people of the world (area), and their task is to collect grasses. They can turn left, right, move forward/backward, talk to each other, or do nothing in each turn. As soon as they move to a new tile filled with grass, they will collect it. They can pass the collected grass to other agents, put it on an empty tile, or use it for their personal score. Our agents can reproduce and have offspring, and they are also evolvable (using Darwin's theory of Evolution).

Agents (organisms) are objects of the organism class, which is decelerated and defined in Organism.hpp and Organism.cpp, respectively. These four agents of the game can be either clone or different. These two types of agents are called group modes that are shown in the next table.

Group mode	Meaning
0	Clone
1	Four different organisms

Table 2. group modes

III. Genetic Algorithm & MABE

A genetic algorithm is an optimization method similar to the process of natural evolution. The algorithm implements the key features of Darwinian evolution: Inheritance, variation, and natural selection. While natural organisms perform these steps autonomously, the genetic algorithm has to implement all these transformations. It follows a simple scheme of initializing a population of random solutions. It then determines the score (fitness) of each individual in the population. Those scores are used to identify solutions that are allowed to propagate offspring into the population of the next generation. Offspring here is a copy of the ancestral solution but with small variations (mutations). Over many generations, this process optimized the fitness (scores) of the solution. The mutations are applied randomly, which causes the process to be slow, and it also does not guarantee that it finds the optimal global solution. However, it can often find solutions in cases where other methods get stuck.

MABE implements groups of individuals and groups of clones differently. When groups of individuals are tested, four random agents from the population are tested, and their fitness is measured according to the four different reward schemes. When the next generation is formed, mutations are applied normally. When groups of individuals are tested, a single organism is cloned three times so that a group of four clones can be used in the environment. After those four clones complete the test, three of them are removed, leaving one agent with an assigned fitness. When that agent is selected for reproduction, it can again experience the same mutations (variance) that exist when testing individuals.

IV. Objective Function

I used different reward schemes to evaluate agents, which can lead to different evolutionary adaptations. Individual and mean rewards are commonly used. Maximum and Minimum are novel in that they have not been tested in a GA, or at least I can not find references in which this has been done.

Reward schemes	Meaning
0	Individual reward
1	Mean score (What they got on average)
2	Maximum score (What their best performer received)
3	Minimum score (What their worst performer received)

Table 3. Reward modes

These reward schemes consider the amount of food collected, and given the conditions rewarded, translate the collected food into different fitness values for the genetic algorithm. For example, when I set the rewarding scheme to zero (individual), whatever a single agent received individually will determine its fitness. Thus, those who collect more food will get a better score and have a higher chance of producing more offspring. If the rewarding scheme is set to 2, for example, the agent who collected the most food is identified, and the amount of food of this agent now determines the fitness of each agent within the group. Thus all agents within the group receive the same rewards and thus have the same competitive advantage during Evolution. In order to have higher fitness than agents from other groups, they need to work together and ensure one agent collect more food than any other agent from any other group.

V. Markov Brains and Genomes of Agents

Here I use MABE to evolve Markov Brains as the controllers for the agents. Markov Brains are a form of the neural network, but they do not use the typical aggregation and threshold function. Instead, a Markov Brain defines a set of input and output nodes, similar to the input and output layer of a conventional artificial neural network. A Markov Brain further defines a set of hidden nodes (here, 9, previous tests suggested that this number is sufficient). These nodes function like the nodes of a recurrent layer in a recurrent neural network. They store information for one update and allow future computations to depend on their state. A Markov

Brain thus implements a hidden state machine, similar to a recurrent neural network. However, inputs need to be computed, and outputs have to be generated. This is done using deterministic and probabilistic (fuzzy) logic gates. Each of these gates can receive inputs from all nodes and send output to all other nodes. To which nodes they connect and which exact function each gate is performing is defined by a "genome".

The genome of a Markov Brain is a vector of bytes (0..255) that is read linearly. When a specific sequence of numbers (here 42 followed by a 213) is found, the code that defines a logic gate starts. Each of these sequences thus defines a "gene". The following numbers define the number of inputs and outputs of a logic gate, and the nodes the gate connects to. The following numbers define either a probability or a logic table that defines the function of a gate. Thus, one gene defines one gate. It can happen that gates write into the same node. When that happens, an OR function is used to define the state of the node.

Mutations to Markov Brains do not happen directly to them, but only to the genomes defining them. Point mutations (with a per site probability of 0.001) would change a single site in the vector to a new random byte. Gene deletions of random stretches between 128 and 512 sites long happen with a 20% chance every generation, effectively shrinking the genome. At the same time, duplications of genetic material happen. Again, stretches between 128 to 512 bytes randomly selected are copied and inserted into the genome at a random location, also with a 20% chance every generation. This grows the length of a genome. Genomes are randomly created at the start of each experiment with a length of 5000. Over the course of evolution, genomes are not allowed to expand beyond 20.000 bytes (sites). All these parameters have been used before and allow Markov Brains to evolve consistently. They are the default parameters of MABE. It can be assumed that more optimal parameters exist, but searching for them might not be warranted, like the ones I use already function.

2.1.3. Game execution

So far, I have explained how I have created a world and set our agents' incentives. Each time I run the experiment, I only change two parameters (rewarding scheme and group mode), and the rest will remain constant. The rewarding scheme has four possible options, and group mode has two, and therefore, totally, there are eight types of combinations. Each experiment is repeated one hundred times to ensure our results do not happen accidentally.

For MABE to properly evolve agents, not only the task and how it is rewarded needs to be specified but also several other parameters dealing with Evolution itself are required. The parameters I used here are default parameters that have been shown to function properly before. As they are not changed, I will not discuss them further. Only the number of generations to be 5000 has been tested such that Evolution has sufficient time to optimize agents. Also, the 500 turns each agent in the game can take has been tested and shown to be sufficient to evaluate the performance of the agent or the group. The next two tables explain the constant parameters and variables for the executions:

Parameter	Possible values
Group mode	Clone and Not clone
Rewarding scheme	Individual, Maximum, Minimum and

	average

Table 4. Variable parameters of the experiments (the game)

Parameter	Value
Number of generations	5000
Number of turns	500
Population size	100
Type of brain	Markov Brain, using deterministic logic gates and genetic programing mathematical operators
The initial score of agents	0
Starting genome size	5000
Maximum genome size	20000
Starting number of gates seeded	5
Point mutation rate	0.0001
Deletion and Gene-duplication rate	0.2
Number of hidden nodes	5
Entire area	16*16 tiles including walls

Table 5. Constant parameters of the experiments (the game)

2.2. Data Description and Format of Output Files

When I run our compiled file, it generates three groups of CSV files: LOD, movement, and beep. As I have explained before, I have eight possible combinations (two options for group type and four for reward schemes). Each combination is repeated one hundred times. Therefore, each of the groups (groups of CSV files) includes 800 CSV files. So, our data folder has three subfolders for LOD, movement, and beep, and inside of each of these subfolders, there are 800 CSV files. I will explain the structure of these files in this section:

2.2.1. LOD.csv

LOD is the abbreviation of "Line of Descent", and a concept from evolutionary theory. The LOD describes the sequence of ancestors from a presently living individual back to the origins of life. Because reproduction is asexual, each organism has exactly one ancestor. Here in the computational model, I can trace from any organism in the last generation back to the first generation as MABE keeps track of the ancestral relations automatically. Every mutation that lead to the organism for which the LOD is constructed must necessarily be on this line, and is thus the record of all evolutionary adaptation. LOD is the first and most important part of our data. It includes the score of agents belonging to the line of descent I have used to select the best combination (set of group mode and reward scheme) for our agents (in the results section, I will explain the best combination). I have 5000 generations, and I set the resolution of the LOD files to 100 so, each of them has 50 rows. So, each row contains information of one generation that are the score of all the four members and the score and the ID of the one which belongs to LOD. I have explained the columns in the next table. The term score is the actual payoff an agent received given the different selection regimes. OwnScore, on the other hand, is the exact amount of food each agent collected, on which the reward scheme bases the payoff. I used the LOD files to generating the evolution graphs (see results section: How agents adapt over the course of Evolution), finding the best combination (see results section: Comparison of the different scenarios), and the analysis on clonal groups (see the last two subsections of the results).

Column name	Explanation	
Generation	Generation number	
ID	ID of each agent	
Score	This field, based on rewarding scheme value, has the following meanings:	
	Reward mode	Score Meaning
	0	Individual score
	1	Mean score of the four agents
	2	Max score of the four agents
	3	Minimum score of the four agents
rawScores	Scores of all the group members	
ownScore	The own score of the agent which is selected in the line of decent	

Table 6. Explanation of the LOD files structure

2.2.2. Movement.csv

I used the movement files to analyze our users' actions, such as using a beep or giving their grass (energy) to other agents (see Agents' behaviors in the results section). These files contain only information about the last agent on the LOD from the last generation. They explain the agents' actions in each turn, and therefore they have 500 rows (because our game has 500 turns). These columns (actions) are explained in the following table. I must mention I worked on all of them but, only A and B representing usage of beep and passing the grass (either to an agent or an empty tile) and do nothing are used in our results and analysis.

Column name	Explanation	
T	Turn number	
X	Position in X axis	
Y	Position in Y axis	
D	Direction	Meaning
	0	Up
	1	Right
	2	Bottom
	3	Left
E	The number of grasses that the agent gathered. It can be named energy as well.	
A	Action type	Meaning
	0	Do nothing
	1	Turn left
	2	Turn right
	3	Move forward
	4,5,6,7	giving grasses to an agent or

		putting it on a tile.
B	Beep (yes or no)	Meaning
	0	No
	Any other number	Beep

Table 7. Explanation of the movement files structure

2.2.3. Beep.csv

The last group of files is beep. These files are used to identify if our agents use beep purposefully. So, I have tested their usage of beep in two conditions: mute (when they can not beep) and talking (when they can beep). Each test is replicated 100 times, which is equal to the number of rows for these files.

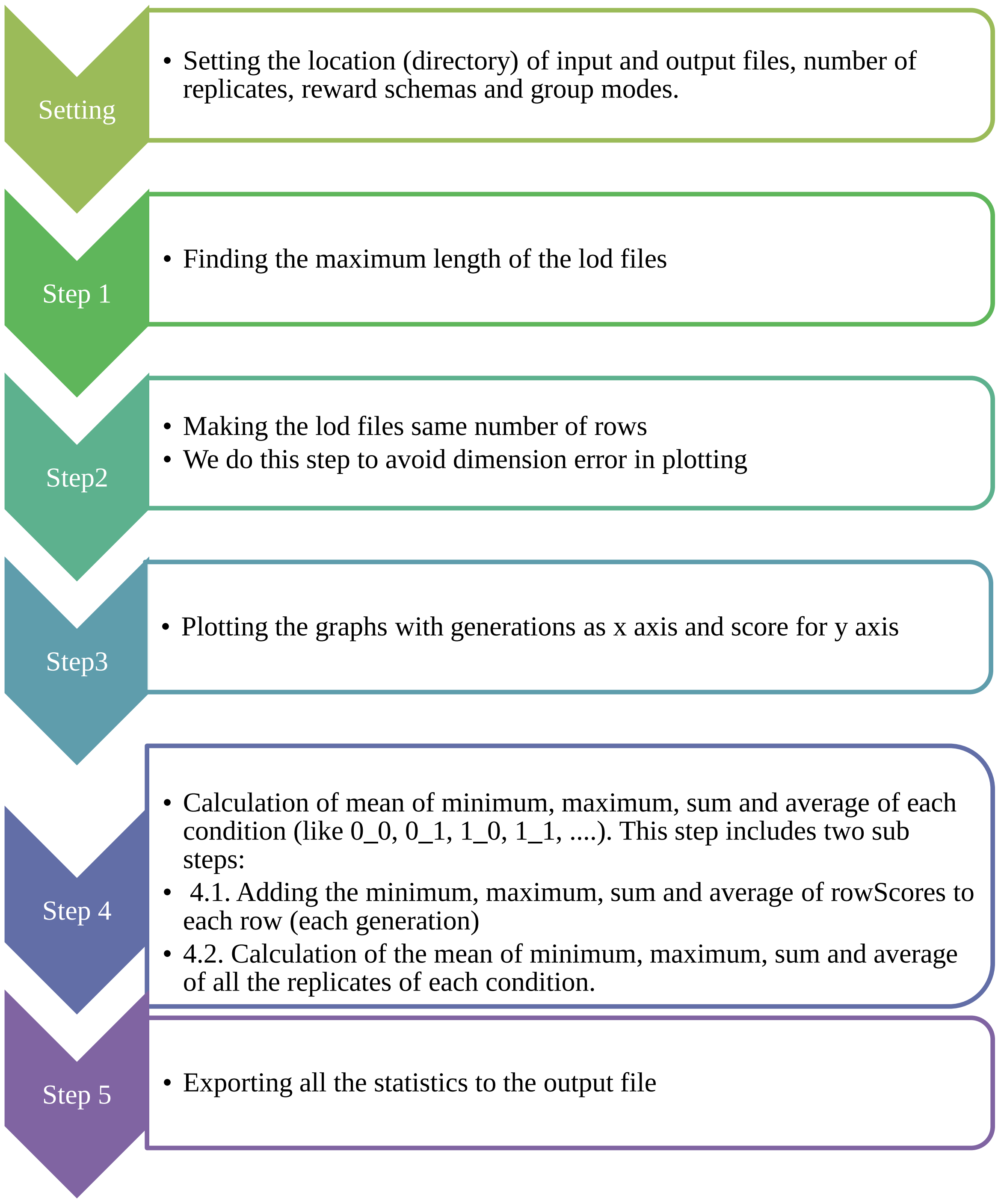
Column name	Explanation
Replicate	The replicate number
m0 to m3	These four columns show the score of the four agents when I have muted them.
b0 to b3	These four columns show the score of the four agents when I have allowed them to communicate (using the beep)

Table 8. Explanation of the beep files structure

2.3. Methods

2.3.1. LOD analyzer

I have created lodAnalyzer.py to convert LOD files to our desired graphs and a table. Its input is all the 800 LOD files. It generates eight graphs (one for each combination) which explain how agents adapt throughout Evolution (see the first subsection in the results) and a table called lod statistics, which I have used for comparison of the different scenarios (see results' second subsection). Also, I have used the information on this table (lod statistics) to compare clone and non-clone groups (see the third and fourth subsections in the results).



- Setting the location (directory) of input and output files, number of replicates, reward schemas and group modes.

- Finding the maximum length of the lod files

- Making the lod files same number of rows
- We do this step to avoid dimension error in plotting

- Plotting the graphs with generations as x axis and score for y axis

- Calculation of mean of minimum, maximum, sum and average of each condition (like 0_0, 0_1, 1_0, 1_1, ...). This step includes two sub steps:
- 4.1. Adding the minimum, maximum, sum and average of rowScores to each row (each generation)
- 4.2. Calculation of the mean of minimum, maximum, sum and average of all the replicates of each condition.

- Exporting all the statistics to the output file

Figure 6. The LOD analyzer code flow

To ensure this module's results (graphs) are accurate and correct, I have compared a small percentage (0.25% of LOD_0_0_0 and LOD_0_0_1) of the python code results with the manual outputs from excel. The next figure shows that the python code graphs match the graphs created by excel tools.

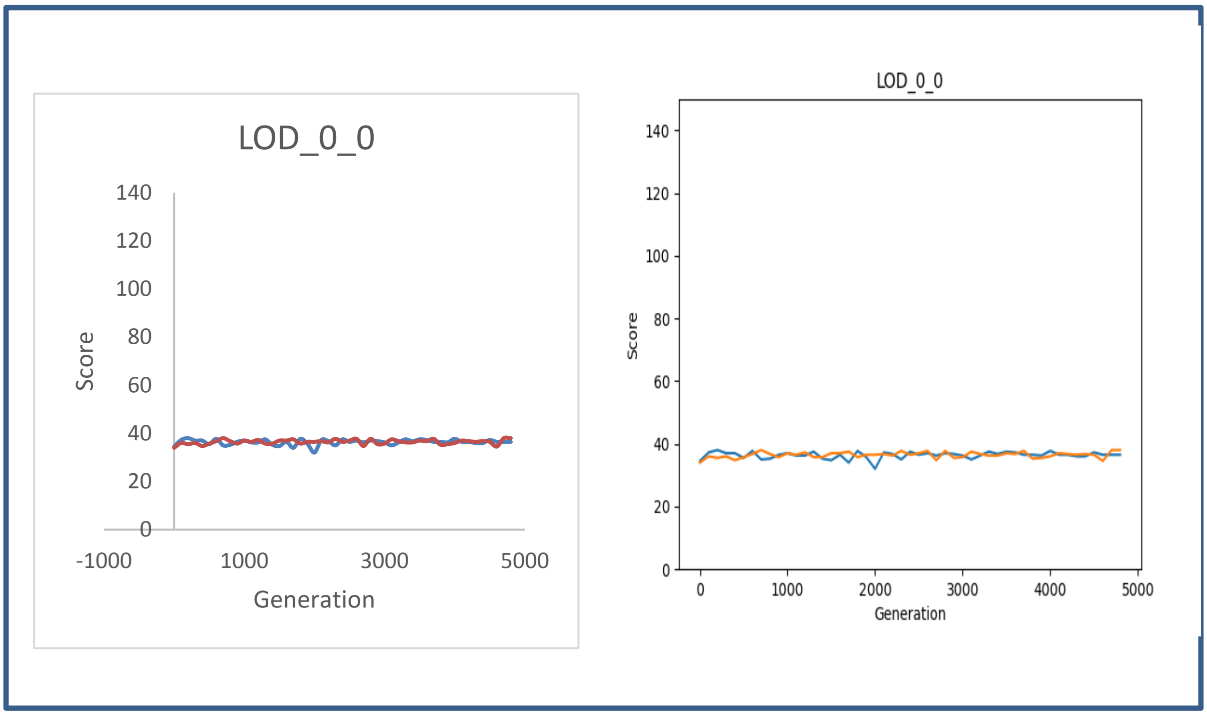


Figure 7. comparison of the python cod's exported graph and graph created using excel. The left graph is generated using manul calculations with excel and the right one is from python codes. The two collors are used for the two replications.

As I mentioned before, this python code (lodAnalyzer.py) also generates a table containing all the combinations' statistics. For this test, I have checked if I calculated the minimum, maximum, sum, and average of each row correctly. For this case, I calculated two of the rows from LOD_0_0_0 and LOD_0_0_1 manually and compared them with the python results.

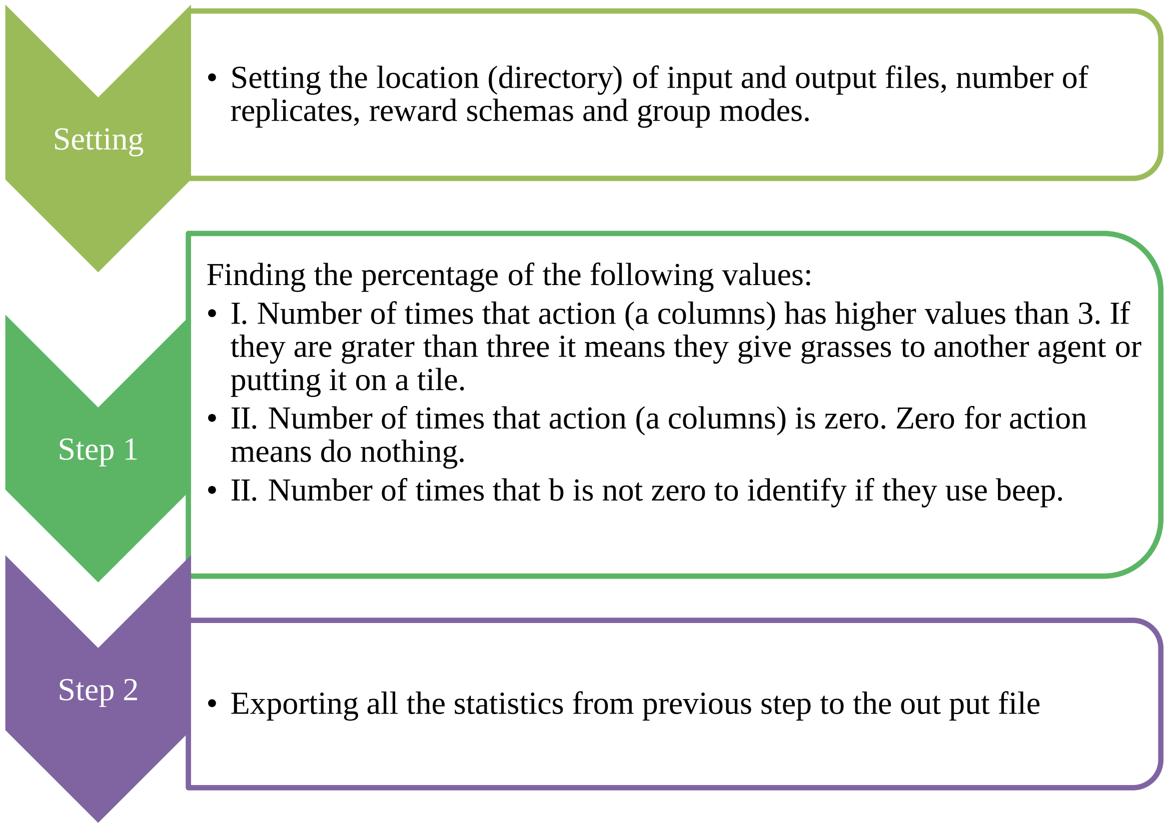
Reward GroupModes	Mean OwnScores	Mean Minimums	Mean Maximums	Mean Averages	Mean Sums
LOD_0_0 Replicate 0 & 1 (Excel)	36.41326531	25.83673469	48.3775510	36.413265	145.653061
LOD_0_0 Replicate 0 & 1 (Python)	36.41327	25.83673	48.37755	36.41327	145.6531

Table 9. comparison of the python cod's results and manual calculations using excel

All the results from python and manual calculations using excel for this small sample are matched (see table 9).

2.3.2. Movement analyzer

This module (movementAnalyzer.py) gets the movement files (800 files) as the input and exports two excel files. The first file (movStatisticsLog) contains all the files' statics, and therefore, it has 800 rows. The statistics are about the three main agents' actions (do nothing, passing the grass, and beeping). The next excel file (movStatistics), which is the main one, summarizes the previous file for the eight combinations. I have used this information to analyze the agents' behaviors (fifth subsection in the results).



- Setting the location (directory) of input and output files, number of replicates, reward schemas and group modes.

Finding the percentage of the following values:

- I. Number of times that action (a columns) has higher values than 3. If they are grater than three it means they give grasses to another agent or putting it on a tile.
- II. Number of times that action (a columns) is zero. Zero for action means do nothing.
- II. Number of times that b is not zero to identify if they use beep.

- Exporting all the statistics from previous step to the out put file

Figure 8. The movement (actions) analyzer code flow

First, to test this module, I should make sure that each movement file's percentages are calculated correctly. For this proposal, I selected two of the files and compared their results using python and excel.

Secondly, the next testing part is about the averages, which again I compared python and the Excel results (for two of the files).

Reward scheme and group mode	Mean of *giving or putting grasses* percentage	Mean of *do nothing* percentage	Mean of *beep* percentage
Movement_1_1 Excel	0.002	2.0775	21.978
Movement_1_1 Python	0.002	2.0775	21.978
Movement_2_1 Excel	2.006	0.8965	21.8615
Movement_2_1 Python	2.006	0.8965	21.8615

Table 10. comparison of the python cod's results and manual calculations using excel

As the previous table shows, the manual calculation using excel and the python outcomes are identical.

2.3.3. Beep analyzer

I have made this code to ensure that our agents use beeping purposefully (not randomly). Its input is the 800 beep files, and it makes an excel file called beepStatistics. I used this output file information is in the results section for the usage of the beep (table number 15 and 16). Like the other modules, I tested it by comparing its results for a small percentage of data with excel results.

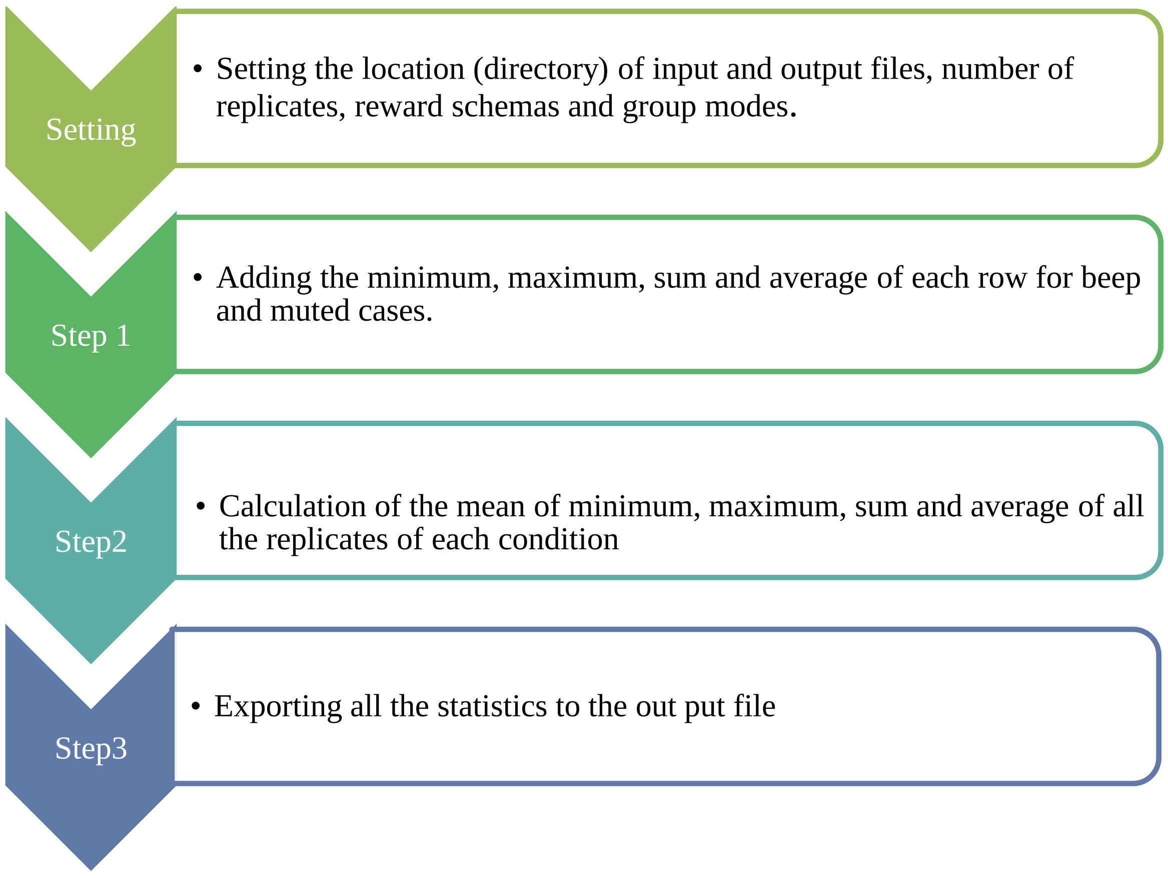

Setting
Setting the location (directory) of input and output files, number of replicates, reward schemas and group modes.
Step 1
Adding the minimum, maximum, sum and average of each row for beep and muted cases.
Step2
Calculation of the mean of minimum, maximum, sum and average of all the replicates of each condition
Step3
Exporting all the statistics to the out put file

Figure 9. The beep analyzer code flow

2.3.4. Statistical Methods

Two different statistical methods were used. The standard error was computed to estimate if measurements are different from each other, used for Table 14.

The two-sided Kolmogorov-Smirnov Test is used to determine if the unknown distributions of values are different from each other, and to estimate a p-value of the statement that both distributions are different by chance. This test was used to assess if values in Table 15 and 16 are significantly different from each other. I do not know if the samples compared in those cases are normally distributed or not; thus, this more complex test and not a simple t-test or other method was necessary.

2.4. The time complexity of the algorithms

The process duration and system description of the experiments (MABE) and analytical modules (python codes) are useful for those who will continue this chain of researchers in optimization.

I used the Michigan State University, Institute for CybeEnabled Research (ICER), High-Performance Compute Cluster with the capacity of ~29000 compute hours (or about 3.2 years a single computer would run) for the experiments with the MABE framework. As I have explained, the experiments are done using clonal and non-clonal groups. The clonal group mode runs take about 12 hours for the 5000 generations, and the non-clonal group mode runs take about three hours for an equal number of generations. The experiments' long durations show that the Genetic Algorithm used in the MABE is not optimized in speed.

The analytical process using the python codes took less time. The durations in seconds are 9, 150, 79, respectively, for the LOD analyzer, movement analyzer, and beep analyzer. I have used a dual-core processor laptop with a frequency of 2 GHz and 8G memory. The python codes are not optimized, and they can be enhanced by reducing the number of loops that go through all the data.

3. Results

The question to be addressed here is about payoff distribution within groups. Specifically, groups of cloned agents were compared to groups composed of randomly selected agents from the population. Secondly, the payoff scheme was varied: Distributing the mean payoff the group obtained to everyone (mean), identifying the payoff the best performer in the group had and rewarding all agents of the group with that performance (max), and lastly, awarding each group member according to the least performer in the group (min). While group-level selection, as present in clonal groups, is believed to improve cooperation, here additionally, payoff according to the worst (min) or best (max) performer is tested, with the hope that it might reveal an even better incentive for cooperation.

For this purpose, all conditions were tested using an evolutionary computational model, testing each condition in 100 replicate evolutionary runs. The results of these experiments are organized into five sections: a) How agents adapt over the course of Evolution, b) Comparison of the different scenarios, c) Minimum and Maximum Selection Schemes and Clonal Groups, d) Agents' behaviors e) do agents communicate for a reason?

3.1. How agents adapt over the course of Evolution

Agents were created randomly at the beginning and had 5000 generations to evolve, given the different experimental conditions. Over the course of Evolution, their performance is expected to improve according to the selection criteria.

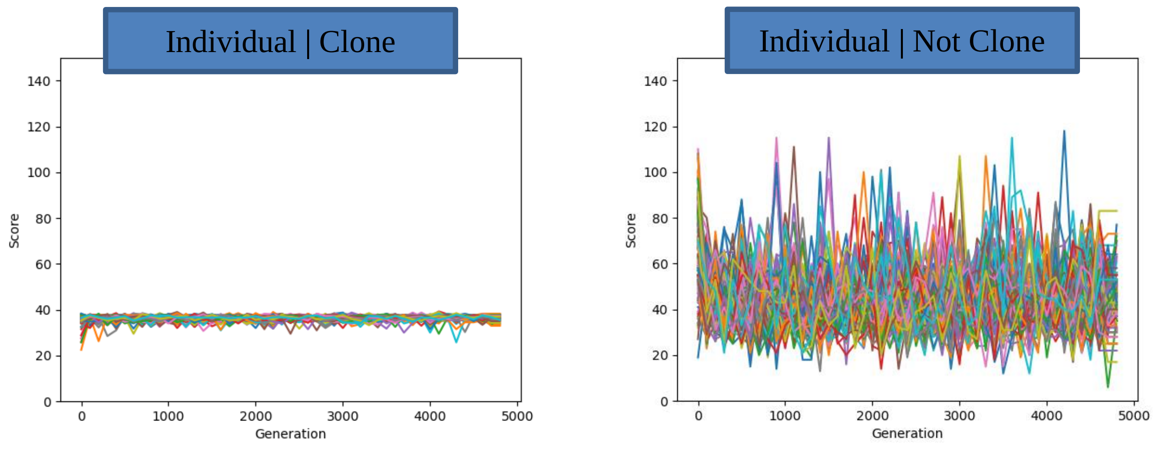


Figure 10. Score variations of the individual rewarding scheme

When rewarding individuals according to the amount of grass collected by themselves, and agents are clones, I find little improvement of performance, and the performance remains roughly constant over the course of Evolution (see Figure 10 left). When the group is not composed of clones, little performance improvements are observed, but performance varies highly during Evolution (see Figure 10 right). Comparing these two different scenarios suggests that the not-clonal group evolves better performance than rewarding clones. Additionally, higher variance during Evolution seems to have a beneficial effect on performance, but this needs to be explored further, comparing more conditions.

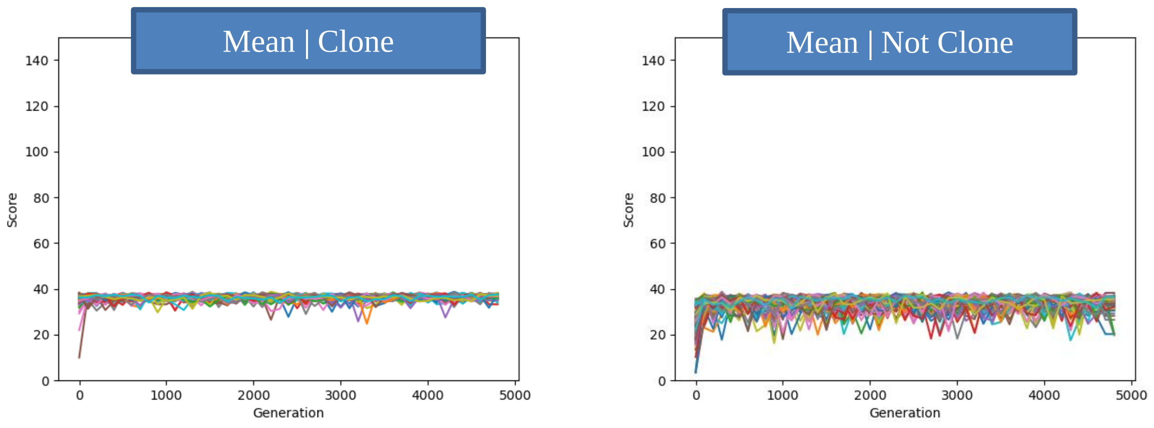


Figure 11. Score variations of the mean rewarding scheme

When rewarding the group members not according to their own performance but according to the average performance of the group, the results differ. While clonal groups again show little improvement over 5000 generations, the not clonal groups show some improvement, but both conditions have little variation during adaptation (see Figure 11 left and right).

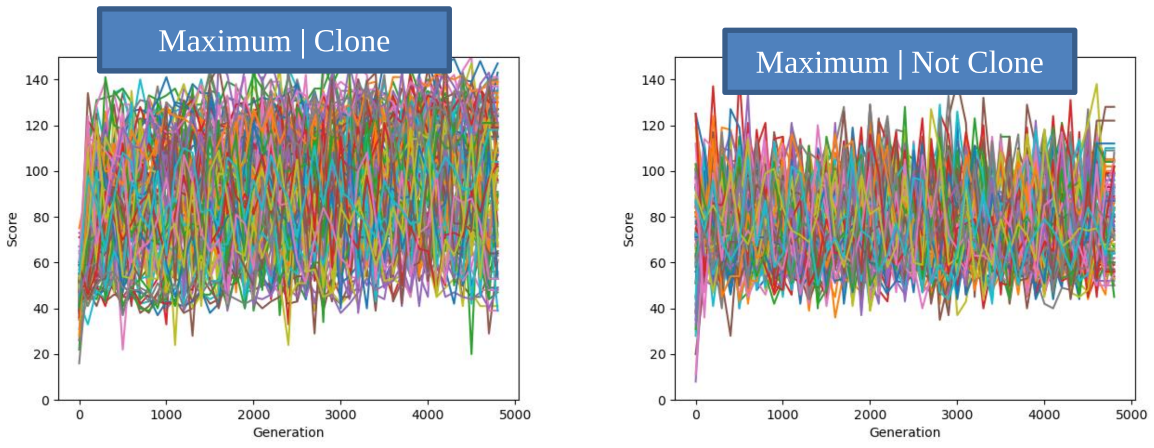


Figure 12. Score variations of the maximum rewarding scheme

The max reward scheme identifies the individual of the group who collected the most grass and now rewards all agents in the group according to that maximum performance. Thus, the group should evolve a strategy that pools resources on one individual or at least avoid that individuals compete about resources. Both clonal and not clonal groups show that their performance improves over generations while also showing a high degree of variance throughout (see Figure 12 left and right). The clonal group seems to have a higher variance as well as a higher performance in the end.

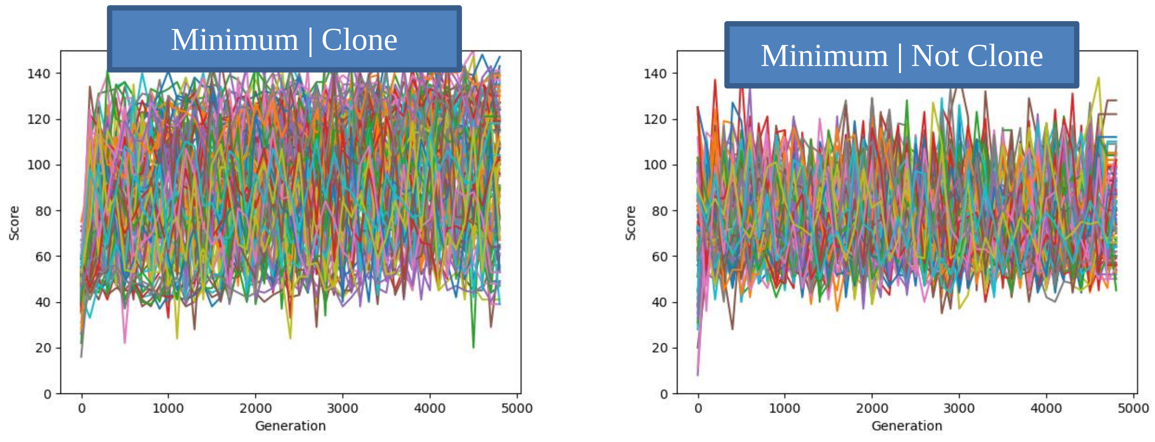


Figure 13. Score variations of the minimum rewarding scheme

The last condition identified the individual who collected the least grass and then rewarded all group members according to that performance. This reward scheme should prevent agents to not participate in collecting grass at all, as that would result in a payoff of zero. At the same time, not one individual should collect overly much grass; instead, agents should devise a strategy to keep the amount of grass collected by everyone equal. Like in the maximum rewards scheme, I find an improvement in the performance of 5000 generations, as well as a high variance within the adaptive periods (see Figure 13 left and right).

At this point, the results are only subjective observations. I am not so much interested in the shape or differences of fitness over generations, as I am interested in the final outcome of the optimization. Thus, in the next section, the performance of agents after evolution will be compared and statistically evaluated. Further, the observation that some conditions lead to high variances in performance, while others do not, seem interesting as well and should be investigated further. Is it possible that those evolutionary runs that show a high variance explore more options and thus have a higher chance to find better solutions?

3.2. Comparison of the different scenarios

As I have discussed in the experiment section, I kept all the parameters constant except the rewarding scheme and how groups are composed (clone or not clonal), resulting in eight different experimental conditions. I expect different experimental conditions to affect individual behavior and how the groups as a whole perform. Thus the criteria compared are the average payoff of the individual across all 100 replicate experiments with the same condition. Further, the average performance of the group member who collected the least (mean minimum) and the most (mean Maximum), the average amount the groups collected (mean Averages), and finally, the average of what groups collected in total (mean sums).

rewardGroupModes	Mean OwnScores	Mean Minimums	Mean Maximums	Mean Averages	Mean Sums
LOD_0_0 Group mode: Clone Reward mode: Individual	36.4275510	26.0175510 🥈	47.4516326	36.4275510	145.710204
LOD_0_1 3 Group mode: Not Clone Reward mode: Individual	44.7497959	20.1351020	52.0906122	36.5244898 🥇	146.097959 🥇
LOD_1_0 2 Group mode: Clone Reward mode: Average	36.4372959	25.8336734 🥉	47.5702040	36.4372959 🥈	145.749183 🥈
LOD_1_1 Group mode: Not clone Reward mode: Average	47.0285714 🥉	11.6457142	56.7748979 🥉	33.9260204	135.704081
LOD_2_0 Group mode: Clone Reward mode: Maximum	91.9987755 🥇	2.70897959	91.9987755 🥇	30.8365306	123.346122
LOD_2_1 Group mode: Not clone Reward mode: Maximum	67.5675510 🥈	1.56244898	75.5930612 🥈	31.0185204	124.074081
LOD_3_0 1 Group mode: Clone Reward mode: Minimum	30.7763265	30.7763265 🥇	42.3973469	36.4311734 🥉	145.724693 🥉
LOD_3_1 Group mode: Not clone Reward mode: Minimum	34.1008163	23.0051020	44.5406122	33.0433673	132.173469

Table 11. Comparison of the different scenarios. Observe that "own score" does not imply the amount of grass collected but what that agent was awarded due to the reward scheme. For example, an agent might have collected more grass than it received as a reward in the minimum scheme, since another agent collecting less defined that reward.

There are five measurements for each scenario (see table 11), and they are ranked using gold, silver, and bronze medals. The performance of individuals is different from who well each member in a group fair. Thus, the average minimal performance could be understood as an indicator for the entire group. Obviously, the reward that every group member receives will always be equal or better than the amount of grass the worst performer collected. Similarly, the total performance (Mean Averages, and Mean Sums) of the group reflects how well groups work together.

With respect to the total number of grass collected (Mean Averages, and Mean sum), I find that the not clonal group with individual payoff fairs best, followed by clonal groups with the average payoff. Interestingly, the third-best performance is achieved by using clonal groups awarded by the minimum. This confirms that this reward scheme is indeed capable of driving groups to perform optimally.

However, when considering how well every member of the group performs, I have to consider the mean Minimum payoff (which is an individual payoff). The clonal minimal rewarding scheme optimizes this parameter the best (place 1 Mean minimum) followed by Clonal groups with the individual payoff, and Clonal groups with the average payoff. While maybe not surprising that the reward scheme that selects for the highest minimal payoff optimizes this parameter the best, it still confirms that this reward scheme is very capable of not only optimizing the least amount each agent collects, but more importantly, it also compares very well on the total number of grass collected within the entire group (third place in Mean Averages and Mean sum).

When ignoring how well the group performs but focusing on a single individual to collect the most grass, I find the maximum clonal reward scheme to perform best, followed by not clonal groups maximum, and not clonal groups with the average payoff. Again, this might not be surprising as the reward function selects for a single group member to perform optimally, even incentivizing others to give their collected grass to said the best performer.

To sum it up, when I select the clone agent and set the reward scheme as minimum, I get a high average score while ensuring that all group members also fair well. In other words, this combination *trains* our agents to *perform better both individually and in teams at the same time*!

3.3. Minimum and Maximum Selection Schemes and Clonal Groups

As it has been argued before, group-level selection drives cooperation. Here I further find that the two new selection regimes (minimum and maximum) result in the best performances, if the groups are clonal (see Figures 14 and 15).

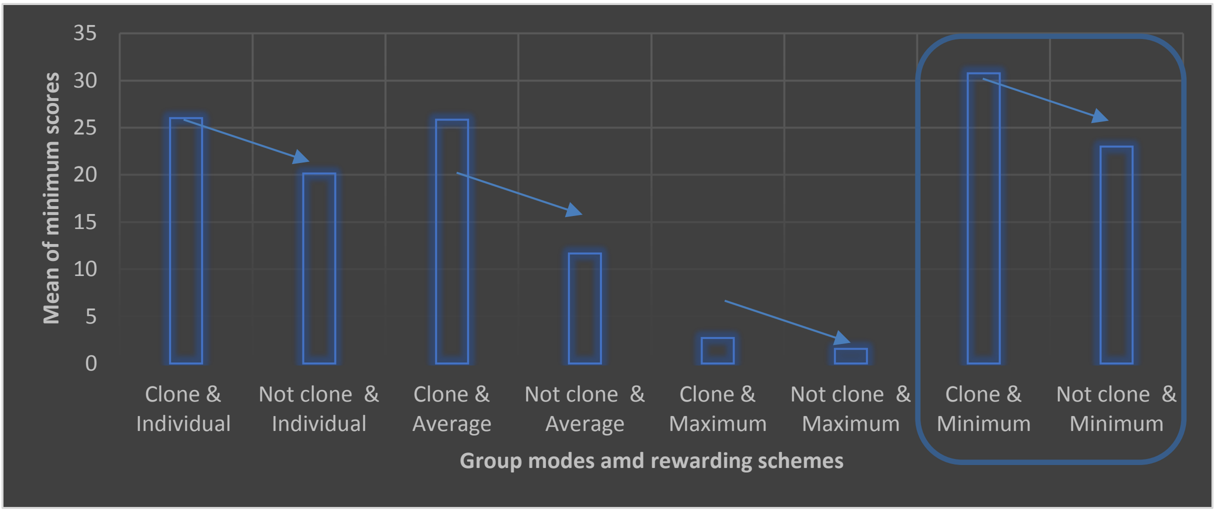


Figure 14. comparison of clone VS Not clone modes in different rewarding schemes based on the mean of minimum scores. The blue rectangle shows the selection regime matching the y-axis, here the selection scheme rewards the group according to minimal payoff, and shows the highest minimal payoff across all other conditions.

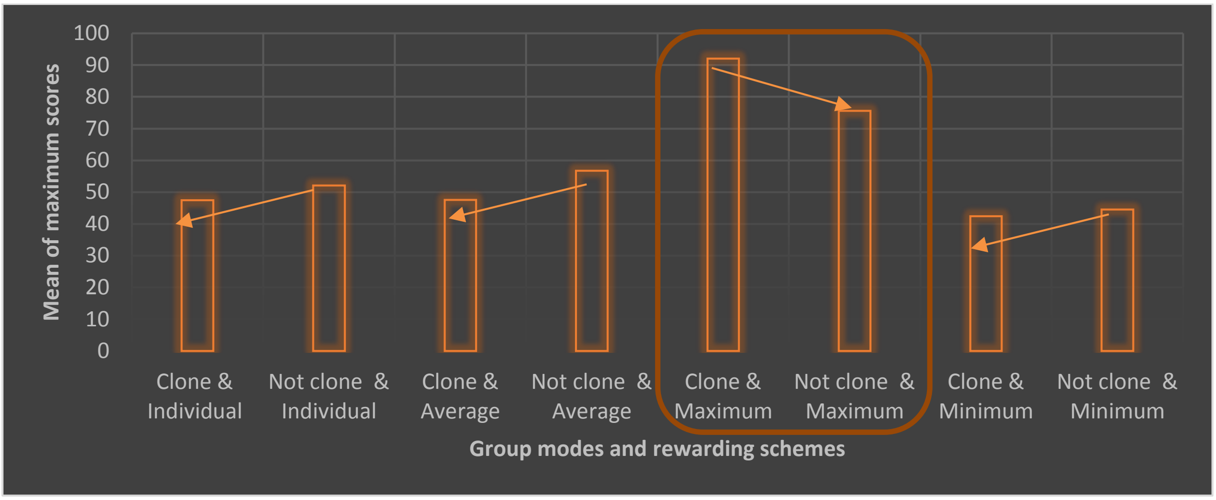


Figure 15. comparison of clone VS Not clone modes in different rewarding schemes based on the mean of maximum scores. The orange rectangle again shows the matching selection regime for the y-axis. Here, the selection regime rewards group members for the maximal performance within the group, which drives the highest individual payoff to be maximal.

While obviously, the selection regime that optimizes for minimal and maximal payoff should drive those outcomes, performance is better in either case, when groups were clonal. In none clonal groups, individuals have to compete against each other, while in clonal groups, all members benefit equally from the success of the group, as they are the same genotype. Thus, both selection regimes should be applied to clonal groups as that promises the best results.

3.4. Other observations about clonal groups

So far, I have discussed the maximum and minimum rewarding scheme which was the main aims of this research; now, I will explain the rest of the rewarding schemes, and check if clonal groups evolve better than none clonal groups.

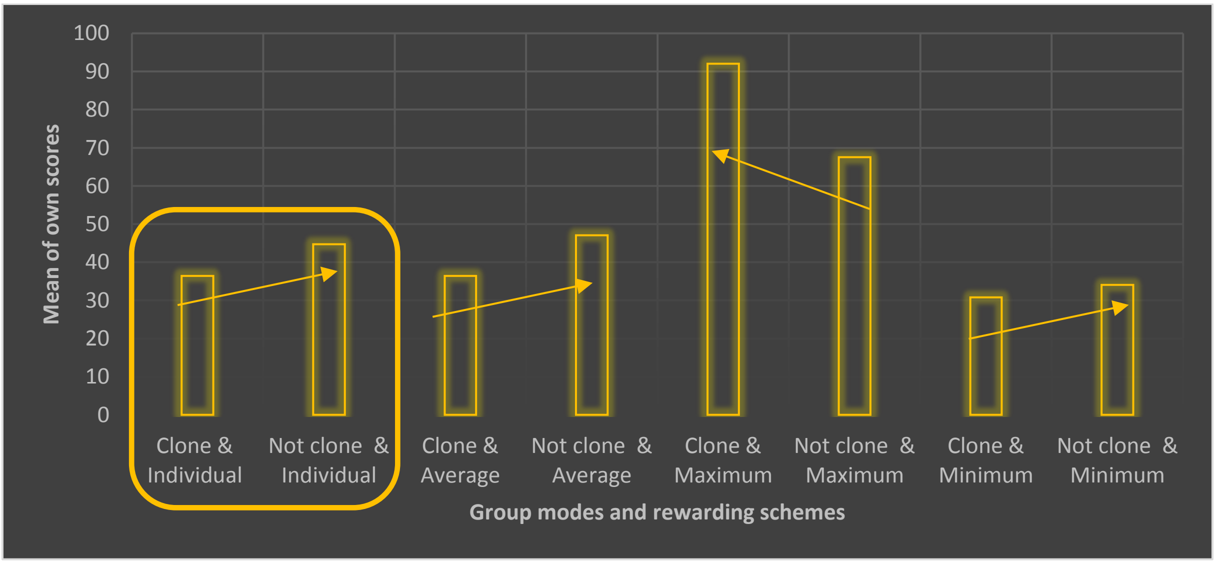


Figure 16. comparison of clone VS Not clone modes in different rewarding schemes based on the mean of own scores. The yellow rectangle again shows the matching selection regime for the y-axis. Here, the selection regime rewards group members for the individual performance within the group.

As you can see in the case of the own-score, when I using the individual reward scheme none clonal groups fair better. Probably because individual rewards benefit selfish agents more. In clonal groups individual benefits would be shared, hampering selfish behavior.

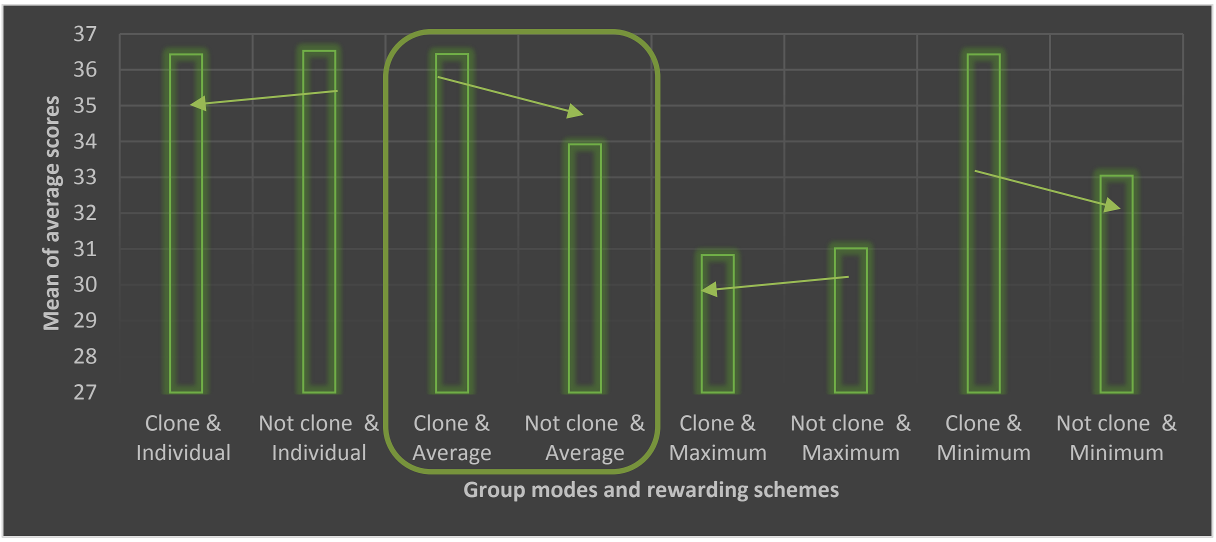


Figure 17. comparison of clone VS Not clone modes in different rewarding schemes based on the mean of average scores. The green rectangle shows the matching selection regime for the y-axis. Here, the selection regime rewards group members for the average performance of the group.

When considering the average score, like in minimum and maximum, groups are dealing with an objective requiring teamwork. Here clonal groups perform better, as those goals are not selfish goals but succeeding in those benefits the entire group. In general, I can conclude that whenever teamwork in this context is required, clones are better.

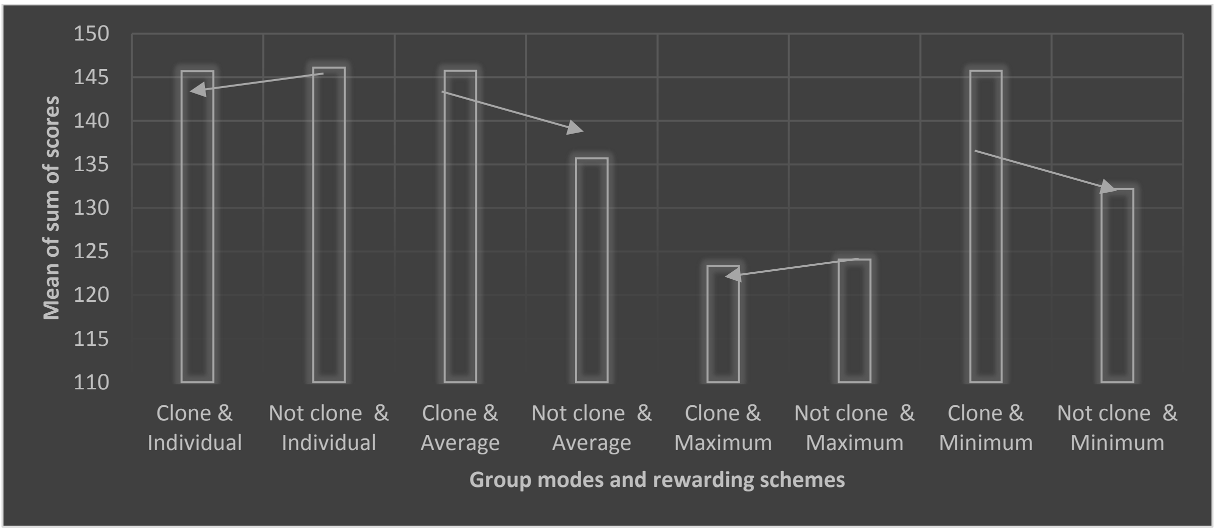


Figure 18. comparison of clone VS Not clone modes in different rewarding schemes based on the mean of the sum of scores. Unlike the other figures in this section here, there is no matching rewarding scheme.

I have not tested a rewarding scheme that is selected for the total payoff of the group directly, and consequently, I do not find a single rewarding scheme to optimize this parameter better in clonal or none clonal groups. In the case of average and minimum rewarding schemes, clones have better performance probably because, in general, they are more evolve more optimally when working in teams. The exact opposite ratio happens in the case of maximum and individual rewarding schemes that causes not clones to perform better. As before, I find that using clonal groups allows to evolve behavior that is benefitted by teamwork easier. At the same time, when the task requires individuals to optimize their own payoff independently of others, not clonal selection is better suited to evolve agents to perform.

Scores	Clonal vs. None clonal groups
Individual food collected	None clonal groups evolve better
Total food collected as a group	Sometimes clonal groups perform better, sometimes none clonal groups perform better
Average food collected with in a group	Mostly clonal groups perform better
Maximum and minimum food collected within a group	Clonal groups manage to work together better, and thus fulfil these objectives better

Table 12. Clonal vs, none clonal groups and their effect on the different scores, such as individual, average, group total, and minimum/maximum score within in group.

3.5. Agents' behaviors

In the comparison of the different scenarios, I have found the best three combinations that were 1. Clone and minimum, 2. Clone and average, and 3. Not clone and individual. I have to ask how those different results are obtained? In order to achieve different payoffs and distributions of payoffs, agents should pursue different strategies and thus should behave differently. For example, in order to maximize the payoff of a single individual (selecting for maximal payoff) one agent should accumulate all rewards, which needs to be organized somehow. Similarly, if selection for minimal payoff is applied, agents should share the workload more equally, again resulting in different behavior. In this part of the results, I will check which actions are the reason behind success or failure in our agents' performance. I have worked on three actions that are explained in the next table.

Action name	Explanation
Beep	Beeping is a way for the agents' to communicate. The beeping channel can have different values. When it is zero, they do not use a beep; any other value is considered a beep, but what those values "mean" depends on the individual brain's Evolution. Here, I distinguish only beeping from none beeping; what those other values mean is neglected for now.
giving grass to other agents or on empty tiles	I have allowed the agents to give their energy (grass) to other agents or simply put it on an empty tile. In this study, I have considered both cases as one action.
do nothing	As its name suggests, it means that the agent does nothing in its turn. While not increasing the amount of food of the individual in most cases, it might be used to avoid taking food from someone else – for example, when selecting for maximum payoff.

Table 13. Agents' actions definitions

Reward scheme and group mode	Mean of *giving or putting grasses* percentage	Mean of *do nothing* percentage	Mean of *beep* percentage
Individual Clone	0 ste +/- 0	0.7005 ste +/- 0.199	13.4225 ste +/- 2.801 L
Individual Not clone	0 ste +/- 0	0.136 ste +/- 0.047	24.147 ste +/- 3.453
Average Clone	0 ste +/- 0	0.296 ste +/- 0.112	17.156 ste +/- 3.207
Average Not clone	0.002 ste +/- 0.001	2.077 ste +/- 0.357 *	21.978 ste +/- 3.485
Maximum Clone	24.948 ste +/- 3.009 *	3.304 ste +/- 0.834 *	53.814 ste +/- 3.637 H
Maximum Not clone	2.006 ste +/- 1.210 *	0.896 ste +/- 0.248	21.8615 ste +/- 3.541
Minimum Clone	0 ste +/- 0	0.963 ste +/- 0.379	51.2315 ste +/- 3.164 H
Minimum Not clone	0 ste +/- 0	8.108 ste +/- 0.671 *	9.5385 ste +/- 2.424 L

*Table 14. The Agents' actions are shown as the percentage of actions taken per evaluation. The standard error was calculated and shown. Different conditions that have an effect are highlighted by *, L, and H. * indicate conditions different from the rest. The letter L indicates conditions to be low; the letter H indicates conditions to be high with respect to remaining values.*

When rewarding groups by their individual or average amount of food collected, I do not expect any cooperative behavior. Payoff can be maximized when every agent individually performs best, and collects the most food. When rewarding the group by the maximum or minimum food collected by an individual on the other hand, coordinating behavior seems to be better, than just trying to collect as much food as possible individually. In the case of maximum, in fact three agents should either do nothing, and leave all food for one agent, or should collect food and hand that collected food over to the one who collected the most food. Either way, also one agent to collect the most food needs to be identified. I find, that agents evolved under this condition, indeed hand over food often. In the maximum selection regime with clonal groups I find agents to give food on average 25% of the time, with a standard error of ~3, suggesting that this behavior evolved across all experimental replicates (see Table 14, left most colum *). Further, under these selection conditions, I find that these agents also on average do nothing about 3.3% of their time (+/- 0.8 standard error). Which might indicate situations in which one of the agents waits for the leading agent to collect food. I observe the same waiting behavior when selecting for minimum (~8) and average (~2) rewards in none clonal groups (see Table 14, middle column *), while all other conditions have agents wait less than 1 action per evaluation in total, which I think has an neglectable effect. For the minimal selection criterion it can be again imagined that waiting for another agent, probably that one who collected the least food so far, again provides a benefit. However, how such behavior benefits agents selected for the average food collected is unclear.

Beeping allows agents to communicate with each other, and it can be expected that this helps them to coordinate their actions. For the minimal and maximal selection regime, this kind of coordination could provide a fitness benefit. Thus, it is not surprising to find this notion confirmed, as agents selected in clonal groups indeed beep on average more than 50% of their time (with a standard error of 3) (see Table 14, right most column marked with H). This is about twice as much as other agents beep, for cases where beeping would not convey a fitness benefit. However, I also find that when selecting for individual payoff in clonal groups, as well as in groups selected for minimal payoff none clonal, beeping is reduced to about 10% of their time (with a standard error of ~2.5) (see Table 14, right most column L). It seems that under these conditions beeping is suppressed. However, it is unclear to us how this avoidance of beeping could provide a fitness benefit.

This leads us to conclude that the two new selection regimes (minimum and maximum) not only impact the total or individual payoff, but also change agent behavior to be in general more cooperative or to be better organized.

3.6. Do agents communicate for a reason?

In the previous section, I have identified that the usage of beep, in general, has a positive impact on our agents' performance for clonal groups. But this result was only a correlation, now we wanted confirm that beeping improved performance directly in an experiment. Therefore, I allowed evolved agents to act without any restriction, and compared those results to agents that were prevented from beeping. The beeping channel was muted (forced to 0, regardless of the agent behavior). The agents tested here, were the product of the previous evolutionary experiment, just tested again under different conditions.

Reward And Group	Beep Mean Minimums	Mute Mean Minimums	Beep Mean Maximums	Mute Mean Maximums
Individual Clone	26.09 pvalue=0.000151	25.24	47.19 pvalue=0.000246	46.11
Individual Not clone	26.958 pvalue=0.699400	26.954	47.53 pvalue=0.545742	47.43
Mean Clone	26.14 pvalue=4.69e-06	25.27	47.45 pvalue=2.85e-05	46.20
Mean Not clone	25.41 pvalue=0.699400	25.45	46.97 pvalue=0.746165	46.99
Maximum Clone	1.56 pvalue=3.4e-176	3.74*	92.88 * pvalue=reported 0.0	34.00
Maximum Not clone	25.13 pvalue=0.011563	25.57	46.89 pvalue=0.192315	47.41
Minimum Clone	31.51* pvalue = reported 0.0	8.72	41.49* pvalue= reported 0.0	15.19
Minimum Not clone	23.72 pvalue=0.048228	23.46	43.85 pvalue=0.192315	43.47

Table 15. Beep vs. mute for the mean of the minimum and maximum scores. The p values of the pairs are presented in the beep columns. Some pvalues have been reported by the python function to be 0.0 (indicated by a pvalue=reported 0.0). This only means that the pvalue is smaller than the precision of the floating point numbers, and thus smaller than 10e-323 and not actualy 0.0. But python interprets these values as 0.0.

Reward And Group	Beep Mean Averages	Mute Mean Averages	Beep Mean Sums	Mute Mean Sums

Individual Clone	36.40 pvalue=0.000193	35.40	145.61 pvalue=0.000193	141.63
Individual Not clone	36.97 pvalue=0.967081	36.96	147.88 pvalue=0.967081	147.84
Mean Clone	36.52 pvalue=2.8484e-08	35.46	146.10 pvalue=2.8484e-08	141.86
Mean Not clone	35.88 pvalue=0.305041	35.92	143.55 pvalue=0.305041	143.69
Maximum Clone	30.00* pvalue=reported 0.0	16.24	120.03* pvalue= reported 0.0	64.97
Maximum Not clone	35.75 pvalue=0.021410	36.21	143.03, pvalue=0.021410	144.87
Minimum Clone	36.46* pvalue= reported 0.0	11.89	145.84* pvalue=reported 0.0	47.56
Minimum Not clone	33.44 pvalue=0.026427	33.15	133.77 pvalue=0.026427	132.61

Table 16. Beep vs. mute for the mean of the average and sum scores. The p values of the pairs are presented in the beep columns. Some pvalues have been reported by the python function to be 0.0 (indicated by a pvalue=reported 0.0). This only means that the pvalue is smaller than the precision of the floating point numbers, and thus smaller than 10e-323 and not actualy 0.0. But python interprets these values as 0.0.

In the case of rewarding according for minimal and maximal in clonal groups I find their performance with respect to the food collected to be highly dependent on beeping. When beeping is muted, clonal groups perform significantly worse. Interestingly, muting communication does not effect the none clonal groups, suggesting that they do not communicate using this channel. Thus, communication evolved only for clonal groups. In itself, this is an interesting find. For communication between genetically unrelated individuals to evolve, it must convey some fitness benefit. Here, such coordination and communication behavior does convey such a benefit, otherwise the clonal groups shouldn't fair better under minimal and maximal reward schemes. However, this potential benefit seems to not be enough for communication to evolve. Or in other words, the benefit from communication and coordination can only be obtained when using clonal groups in the genetic algorithm.

4. Discussion

Optimizing how groups of agents or robots perform will become an important challenge the more autonomous machines will populate our environment. Two prominent optimization techniques have been developed: deep learning and neuro evolution. Here, I focus on neuro evolution, with the aim to develop an objective function that trains groups of AI controllers (agents) to perform better *individually* and in *teams*. Even though it is not tested here, objective functions such the ones developed here to be used in a genetic algorithm can also be used in deep-Q learning as those methods also require to properly assess the performance of agents or groups of agents.

Groups of agents are tested in a simple virtual game, where they need to collect food within a confined space. The details of the game and its parameters are explained in the material and methods; however, the environment allows two parameters to be varied. Groups can be made from clones or random individuals from the population. Further for rewarding schemes have been proposed and tested: a) *minimum* rewarding the group according to the performance of the weakest member, *b) maximum* rewarding the group according to the performance of the strongest member, *c) average* groups are rewarded based on the average performance across group members, and *d) Individual* where each member received a reward identical to its own performance. This results in eight experimental conditions, of which the minimum and maximum conditions are new, while other conditions have been used before. It is also known that selection within clonal groups (similar to group-level selection in Evolution) drives cooperative behavior. As such, the primary question asked here is about the efficiency of the minimum and maximum rewarding objective function as compared to the other ones.

I found that the minimum reward scheme applied to clonal groups not only works equivalent to average and individually competing groups of none clones. Further, this reward scheme ensures that each agent has on average collected more food than in all other groups compared to. Secondly, when selecting for maximum performance, a single agent of the group collects way more food than in any other condition, but at the expense of the total food collected by that group. As such, I get at least one agent to perform excellently, but others sacrifice their performance to accomplish that goal. Therefore, it has been shown that under these experimental conditions, the minimal reward scheme for clonal groups is a viable alternative to the individual or average reward schemes given to clonal or none clonal groups, with the extra benefit of potentially having an equal payoff distribution among group members.

I also found that those reward schemes that resulted in a high variance of performances during the optimization of the genetic algorithm tend to result in higher performance in the end. It is possible that this variation indicates that more and different strategies get explored, resulting in the possibility to choose among more solutions. However, at this point, this is just an observation and needs to be explored further.

Another interesting observation is that when using clonal groups, the minimum and maximum reward scheme result in overall higher performance. While it has been shown that group-level selection improves cooperation, it is new that it also improves the kind of cooperation required to solve this task when using the minimal and maximal reward scheme. One can speculate that when using none clonal groups, competition between group members is maintained, as selection during the evolutionary process remains selfish. When using clonal groups, each member of the group has the same genome, and thus differences between the group members are irrelevant during the selection process. Again, further analysis of this phenomenon is in order, as at this stage, the result is observational.

Since the minimum and maximum reward schemes require agents to coordinate behavior, I allowed agents to communicate using a far-ranging beeping signal. Agents evolved under those conditions beep more often and also lose their performance if those signals are muted. This proves that beeping is used to coordinate behavior among clonal group members. When not using clonal groups, the use of beeping to coordinate behavior does not evolve under these conditions. When thinking about possible future tasks of robots who have to work on complex tasks that also require them to communicate, this result suggests that using clonal groups will allow them to evolve communication easier. However, here I only correlated beeping to function and did not further study the actual nature of the signal. In theory, a binary communication channel like the one provided here might allow for a more sophisticated kind of communication. For example, under the maximum reward scheme, an individual can be identified to collect all the food. How this works has not been studied here, again suggesting a further investigation into this matter.

However, I must mention that these conclusions are based on the specific experiments conducted here. While they might generalize, I should do more experiments. The environment, for example, is a simple food collection game, and a more complex environment are easy to imagine. Similarly, only Markov Brains were used, which present a subset of all possible computational neural controllers. Lastly, the genetic algorithm used simple roulette wheel selection, but much more sophisticated methods such as map-elites [19] and novelty search [20] as well as methods for genetic recombination or sexual selection [21] are possible. Further, it might be tempting to generalize these results to human behavior, suggesting that rewarding groups according to the performance of the weakest member leads to better performance or an equal distribution of resources. An interesting perspective that again warrants further investigation.

This model, specifically the minimum and maximum reward schemes, might allow the study of altruistic and selfish behavior. If collected food is handed over to other agents, without a direct fitness benefit to the one who gives the food, I would truly observe altruism. At the same time, such true altruism is believed to not be able to evolve, as only things that are neutral or beneficial will prevail during Evolution. As such, different conditions that reward seemingly altruistic behavior have been suggested, such as reciprocity [5] or the green beard effect [22] [23]. Here, the benefits come from the selection regime that rewards effective groups over selfish individuals. Studying these effects in more detail is another application of the approach presented here.

5. Conclusions and Potential Practical Applications

The goal of identifying a new reward scheme that improves the individual, as well as the group's performance, has succeeded. Rewarding clonal groups by the performance of the individual that collected the least amount of food (minimum reward scheme) drive the total number of food collected up and also ensures that each member on average performs equally or better than under control conditions

I could do go beyond the initial research aim and study three more aspects. Firstly, the high variance in the agents' scores through the generations causes better performance at the end. The high variance in the scores is probably because of testing a wider variety of methods that enhances the agents' Evolution. Secondly, when I set rewarding schemes to minimum, maximum, and average, the clones perform better, whereas non-clones are better adapting under the individual reward schemes. Finally, I have identified that there is a relationship between agents' scores and their actions (beep, do nothing, and giving food to others or empty tiles). In general, communication or usage of beep causes better performance for the agents when coordination is needed. This work is a proof of concept and sets the stage for a series of more detailed investigations.

While it might be difficult to measure groups of humans by the performance of their weakest group member, it is still a compelling idea to think about. However, when training self-driving cars we have the opportunity to not measure them by how fast they get an individual from A to B, but by the time everyone has to wait in traffic. Which leads to the potential practical applications of this work.

When robots or other AI agents are currently optimized to fulfill a particular task, it is done by optimizing their individual performance, while the effect on the group or environment they act in is ignored. This is at least seems to be true for most genetic algorithms or reinforcement learning approaches. This work, however, suggests that one can not only take the effects that individual performance has on the group into account but that the minimal reward scheme allows us to optimize individual performance while also improving the performance of the group. I do not know of a company that takes advantage of this concept, and it might be difficult to apply the results here to the specific application such as swarms of robots, self-driving cars, recommendation systems, etc. but it suggests that we might overlook an important dimension when only optimizing for individual performance.

Similarly, this method might be a way to incentivize groups of workers or teamwork in general. However, this becomes a psychological and social question, and here only simple machines optimized for functionality have been investigated. Never the less, rewarding teamwork also remains a possible application domain. Concepts for societies or economic growth might also be affected by this research. As mentioned before, the public goods game is a model for the tragedy of the commons and also a metaphor for a society that would benefit from collaboration. However, selfish behavior prevents such fruitful collaboration. It can be easily imagined how a reward scheme like the minimal reward I used here could be beneficial. At the same time, it might be extremely hard to change policies or apply such a reward scheme in practice. Thus, at the moment, the idea to stimulate cooperation within the public goods game or similar real world contexts using a minimal reward scheme remains an idea - but maybe a stimulating one.

Finally, this work was done using a genetic algorithm; however, reinforcement learning seems to be an alternative technology with gains more traction, specifically with deep-Q learning (see, for example, deep mind technology [24]). How the objective function can be used within the domain of reinforcement learning remains unanswered here but presents an extremely important question to be answered for the results here to generalize to reinforcement learning.

References

[1] L. Joel, C. Jeff and M. Dusan, "The surprising creativity of digital evolution," in *Artificial Life Conference Proceedings 14, pages 76-83. MIT Press,* 2017.

[2] A. Hintze and M. Mirmomeni, "Evolution of autonomous hierarchy formation and maintenance," in *Artficial Life Conference Proceedings 14, pages 366-367. MIT Press,* 2014.

[3] B. Gergely and S. Szabolcs, "Beneficial laggards: multilevel selection, cooperative polymorphism and division of labour in threshold public good games," *BMC evolutionary biology,* 2010.

[4] G. Hardin, ”The Tragedy of the Commons,” *Science,* vol. 162, nr 3859, pp. 1243-1248, 1968.

[5] M. Nowak, ”Five Rules for the Evolution of Cooperation,” *American Association for the Advancement of Science,* vol. 314, nr 5805, 2015.

[6] "flaticon," [Online]. Available: https://www.flaticon.com/. [Accessed 25 January 2021].

[7] "myiconfinder," [Online]. Available: http://www.myiconfinder.com/. [Accessed 25 January 2021].

[8] "La evolución," [Online]. Available: https://laevolucionsp.weebly.com/the-natural-selection-of-darwin-and-wallace.html. [Accessed 23 January 2021].

[9] W. DS och S. E, ”Reintroducing group selection to the human behavioral sciences,” *BEHAVIORAL AND BRAIN SCIENCES,* vol. 17, nr 4, pp. 585-608, 1994.

[10] A. Hintze, J. A. Edlund, S. O. Randal, B. K. David, S. Jory, A. Larissa, T. S. Ali, K. Peter, S. e. Leigh och G. Heather, ”Markov brains: A technical introduction,” 2017.

[11] C. Bohm and A. Hintze, "Mabe (modular agent based evolver): A framework for digital evolution research," in *Artificial Life Conference Proceedings,* 2017.

[12] J. A Edlund, N. Chaumont, A. Hintze, C. Koch, G. Tononi och C. Adami, ”Integrated information increases with fitness in the evolution of animats,” *PLoS Computational Biology,* vol. 7, nr 10, 2011.

[13] L. Albantakis, A. Hintze, C. Koch, C. Adami och G. Tononi, ”Evolution of integrated causal structures in animats exposed to environments of increasing complexity,” *PLoS*

Computational Biology, vol. 7, nr 10, 2011.

[14] R. S. Olson, A. Hintze, F. C. Dyer, D. B. Knoester och C. Adami, ”Predator confusion is sufficient to evolve swarming behavior,” *J. Royal Society Interface,* 2013.

[15] C. Ward, F. Gobet och G. Kendall, ”Evolving collective behavior in an artificial ecology,” vol. 7, nr 2, pp. 191-209, 2001.

[16] H. J. Briegel och G. De las Cuevas, ”Projective simulation for artificial intelligence,” *Scientific Reports,* vol. 2, nr 1, 2012.

[17] M. W. McCarter a and others, "Models of intragroup conflict in management: A literature review," *Journal of Economic Behavior and Organization,* 2018.

[18] C. Donald E, Incentives: motivation and the economics of information, New York: Cambridge University Press, 2018.

[19] C. Antoine, C. Jeff, T. Danesh och M. Jean-Baptiste, ”Robots that can adapt like animals,” *Nature,* vol. 521, nr 7553, 2015.

[20] L. Joel och S. Kenneth O, ”Exploiting open-endedness to solve problems through the search for novelty,” i *Artificial Life XI: Proceedings of the 11th International Conference on the Simulation and Synthesis of Living Systems, ALIFE 2008,* 2008.

[21] J. KOZA, ”GENETIC PROGRAMMING AS A MEANS FOR PROGRAMMING COMPUTERS BY NATURAL-SELECTION,” *MIT press,* vol. 4, nr 2, pp. 87-112, 1994.

[22] K. Laurent och R. Kenneth G, ”Selfish genes: a green beard in the red fire ant,” *Oxford university press,* vol. 394, nr 6693, pp. 573-575, 1998.

[23] R. Dawkins, The Selfish Gene, Oxford university press, 2016.

[24] D. Silver, A. Huang, C. J. Maddison, A. Guez and others, "Mastering the game of Go with deep neural networks and tree search," *nature,* vol. 529, no. 7587, pp. 484-489, 2016.

Appendix: the source codes

All the source codes for the thesis are uploaded in the following Github repository:

https://github.com/bamshad-shrm/allOne