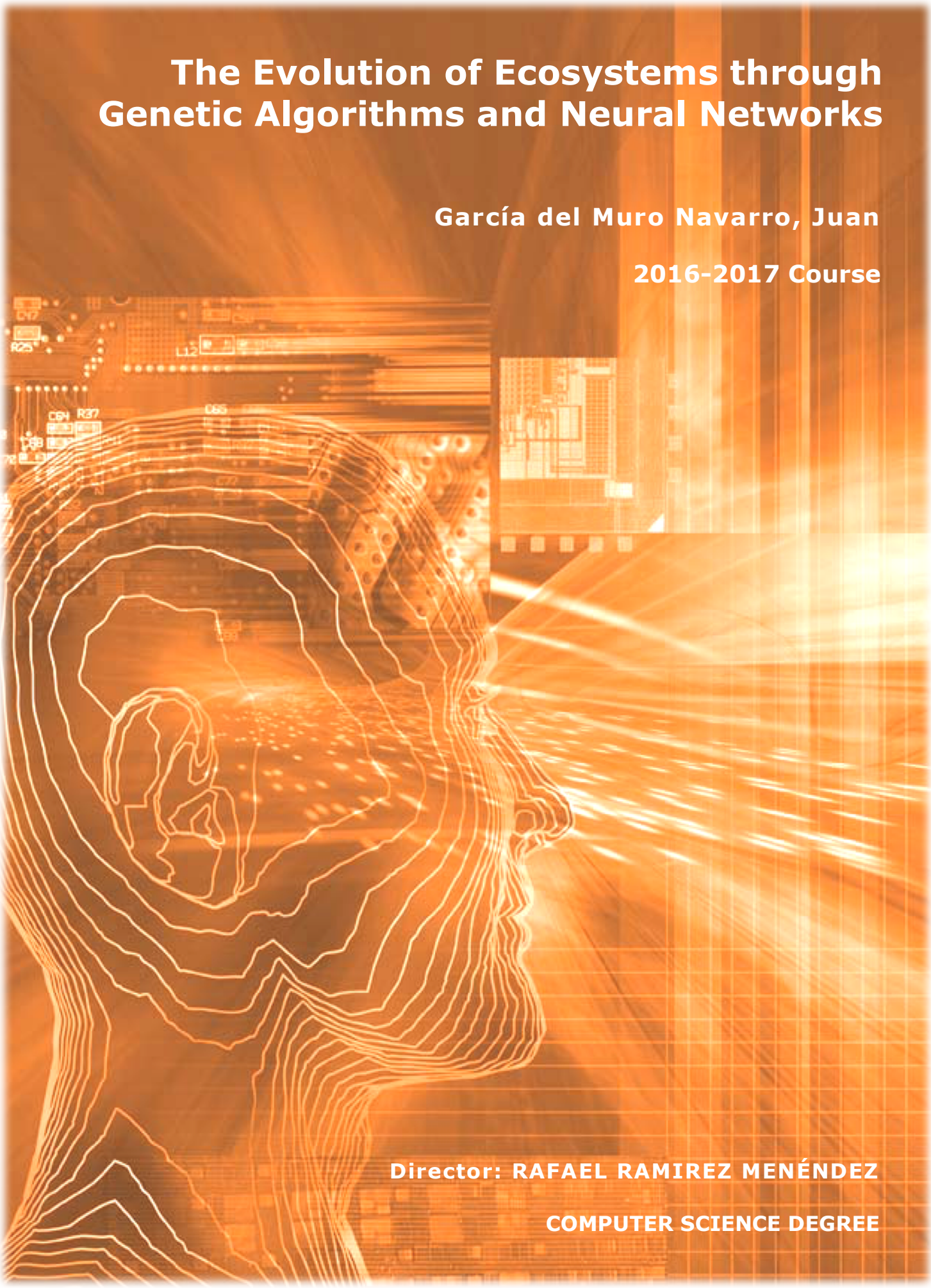
The Evolution of Ecosystems through Genetic Algorithms and Neural Networks

García del Muro Navarro, Juan

2016-2017 Course



Director: RAFAEL RAMIREZ MENÉNDEZ

COMPUTER SCIENCE DEGREE

Universitat Escola

Artificial Intelligence machines are programmed to achieve a certain goal, and so are organic living forms. The only true difference between the two is that they have met their creators, and it won't take them long to outsmart them.

Acknowledgments

I would like to thank my family for always supporting me in everything I do. It means the world to me.

To Laura, for the incredible effort that she has put into this document.

To Rafael, for his guidance and help, and to all of the people that have selflessly collaborated with me in this project.

Abstract

The goal of this TFG was to build an ecosystem simulation platform through genetic algorithms and neural networks. The behaviour of the entities involved depends on a virtualised model of genetic material. This genome model and the genome mixer in charge of generating the new generations of individuals mimics biological evolution.

As this platform is biologically accurate, and the core principles for evolution according to Darwin have been modelled and included into the simulator, it could be a helpful research tool that would help corroborating or discarding evolutionary hypothesis and theories. Some examples would be the evolutionary advantages of a limited lifespan versus a non-limited lifespan, or the drivers for a species to become carnivore or herbivore. In the second part of the paper, we delve into some of these theories and through ecosystems simulations, we will contrast them.

Resumen

El objetivo de este TFG era construir una plataforma de simulación de ecosistemas a través de algoritmos genéticos y redes neuronales. El comportamiento de las entidades involucradas depende de un modelo virtual de material genético. El modele genético y el mezclador de genes encargado de generar las nuevas generaciones de individuos imita a la evolución biológica.

Ya que esta plataforma es biológicamente precisa, y los principios centrales de la evolución según Darwin han sido modelados e incluidos en el simulador, puede ser una herramienta de investigación útil que podría ayudar a corroborar o descartar hipótesis y teorías sobre la evolución. Algunos ejemplos serían las ventajas evolutivas de un tiempo de vida limitado, en comparación con un tiempo de vida no limitado, o los factores que empujan a una especie a convertirse en carnívora o herbívora. En la segunda parte del trabajo, profundizaremos en algunas de estas teorías y, mediante simulaciones de ecosistemas, las contrastaremos.

Prologue

Where do we come from? Where are we headed? The origin of our species and the meaning of life has been one of the oldest and greatest mysteries for humanity. We have spent centuries theorising about it, seeking answers in religion, philosophy and science, coming up with thousands of different hypotheses, most of them contradicting each other. In many cases, those contradictions led to some of the greatest wars in history.

It was not until the 18th century when modern science emerged and tackled this question. Joining the scientific revolution in 1859, Charles Robert Darwin, an English naturalist, published what would arguably become the most notorious publication in the field of biology, *The Origin of Species by means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life.* In this book, Darwin presented his theory of evolution, proposing a new explanation to account for the phenomenon of biological diversity. Leaning on the work of the scholars of his era, he developed his theory solely by observing nature, armed with a notebook and a pencil.

Centuries have passed since then, and the evolutionary hypothesis was refined with contributions coming from very diverse sources: Gregor Mendel from Genetics, Pierre Teilhard de Chardin from Philosophy, Motoo Kimura and Linus Pauling from Biochemistry and Ernst Mayr from Biology. In the last hundred years, science has advanced more than in the previous three thousand and has given mankind greater powers over the forces of nature than ancients ascribed to their gods. We have technological tools at our fingertips that they couldn't even dream of, carrying devices with enough computational power to solve a billion floating point operations in a second, and I am convinced that it is our duty to take over and use these tools to broaden humanities knowledge and understanding of the universe.

This project is my own small contribution to it. Artificial Intelligence allows to channel all of that computational power and through genetic algorithms, neural networks and virtual genome models, we can not only theorise about these matters but simulate ecosystems, varying the conditions at will. I have modelled the core concepts of the transformation of species according to the evolution theory accurately enough for it to become a tool capable of helping contrast biological theories.

Index

Figure List

Chart List

1. Introduction

1.1 Aims and Goals

The main goal of this project was to build a platform capable of simulating ecosystems, based on the biological evolutionary theory. We want to create a system where the individuals taking part in the simulation would learn from their experience in the environment and evolve, an interdisciplinary tool for researchers in the field of biology and computer science. To achieve it, we decided to take an artificial intelligence and machine learning approach, using genetic algorithms and neural networks as the core of the simulator and building the platform around it.

In a real ecosystem, there are an incredibly large number of relevant factors to take into consideration, so we had to decide which aspects we were going to invest more resources modelling and which ones were going to be simplified. We have focused on the animal organisms and their evolution, implementing an asexual reproduction based on mutation. We have simplified plant organisms, as well as factors that were not related directly to the evolution, but were still essential to the simulation.

A crucial point when designing the algorithms was that, in order to reach a stable ecosystem, the simulation had to include a significant number of individuals, and the higher that number, the richer the results would be. For instance, if the simulation takes place in a small area, many potential strategies are automatically ruled out. These strategies could have been adopted by an individual in order to survive. If the area where the simulation takes place is too small, there is a risk for the plant organisms to go extinct and the remaining individuals would simply die of starvation. Because of this phenomenon, it was key to design a platform capable of running simulations with as many individuals as possible, leading us to the last goal: optimisation. We had to design the algorithms with a linear execution cost function, to improve the system's scalability.

1.2 Thesis Structure

This thesis is divided in three main parts. In the first part titled Background, we gathered all of the information required to fully comprehend the concepts that appear in the paper. It serves as an introduction and summary of the biological evolution theory, artificial intelligence and machine learning, and the connection to genetic algorithms, genetic models and neural networks.

In the second part, Tools and Methods, we will dive into the platform itself and its technical details, from a theoretical, mathematical, and practical prospective, as well as the software implementation and the hardware used to develop and train the system.

In the last part of the thesis, we will present the results of different simulations, designed according to different evolutionary theories, illustrating the way the platform works.

2. Background

2.1 Biology

a) Evolution Theory

As we mentioned in the prologue, the modern theory of evolution is mainly attributed to Charles Robert Darwin, but he was not the first one to abandon the idea of fixed, invariable species. In fact, long before he published the work where he presented his theory, *The Origin of Species by means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*, most biologists agreed that the variation between species was achieved through some sort of organic evolution. But Darwin's work was the first solid theory that explained in detail how this might occur. It is worth highlighting that, even though Darwin is the father of modern biology, and that with his theory, he stablished the foundation of organic evolution, from now on, when we speak about the evolution theory, we will not only be limited to Darwin's work, but also take into account many other contributions from a variety of other fields, like biochemistry, genetics, physics and biology that have helped create the modern theory of evolution.

According to the theory of evolution, the population of a given species includes individuals of varying characteristics at a given time. The population of the next generation will contain a higher frequency of individuals descending from those that have been more successful at surviving and reproducing. Evolution of a species as a whole results from this frequency. Organic evolution is viewed as a sorting process, the survival of the fittest in a certain environment. This process is what Darwin called natural selection.

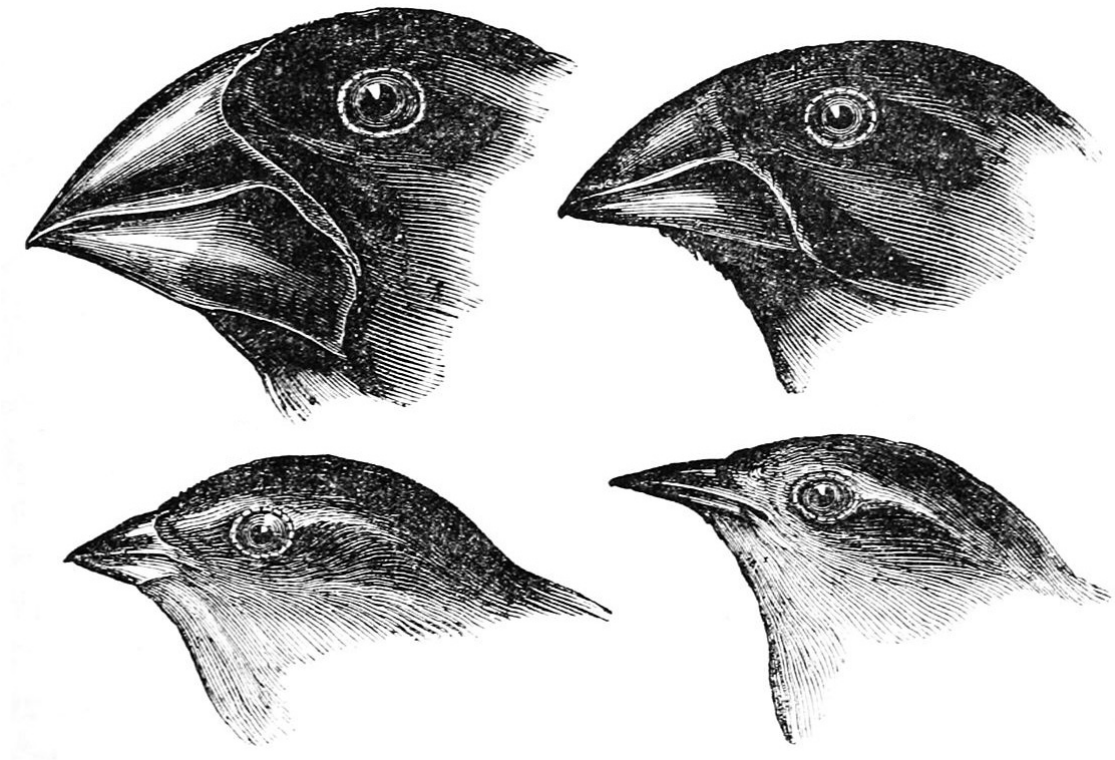


Figure 1 Representations of four different species of birds that share the same ancestor. The size of their beak varied in order for them to adapt to the environment, resulting in new species

We can summarise the theory of evolution in three core principles that all organic ecosystems that evolve over time share:

- Principle of variation: Among individuals within any population, even if they pertain to the same species, there is a variation in morphology, physiology, and behaviour.

- Principle of heredity: The individuals which are part of the next generation resemble their parents more than other individuals.

- Principle of selection: In a given environment, some individuals are more successful at surviving and in extension, at reproducing. This selective process clearly leads to a change in the population's composition. That variation must be in some part heritable, in order for it to impact the composition of the population. Even though the variation is heavily dependent on the predecessor or predecessors, there is also a random component linked to it.

Including these three principles in the simulation is essential in order to mimic organic

evolution, and we will refer to them later on in this paper simply as the requirements for evolution. And while this is enough information on the evolution theory to understand the main concepts of the mechanisms of an ecosystem and its evolution, there is a very important matter that we still have to tackle.

In the next section, we will speak about genetic material as a way of storing all of the information directly related to morphology, physiology, and to some extent, behaviour.

b) Genetic Material and Mutation

Genetic material is the hereditary component where all of the information directly related to the individual, and its genetic heritage is stored. In particular, DNA, or deoxyribonucleic acid, is the genetic material present in almost all organisms. The way DNA stores information is through a chain of four chemical bases: adenine (A), guanine (G), cytosine (C), and thymine (T). The order of these bases determines the information available for building and maintaining an organism, as in a computer string variable. If that were the case, considering that the length of the sequence for humans is approximately three billion (American billions) bases and using four different bases, the number of bits storable would be astronomical. The storage capacity would correspond to $4^{3000000000}$ bits, equivalent to $9.6 * 10^{1806179961}$ Terabytes.

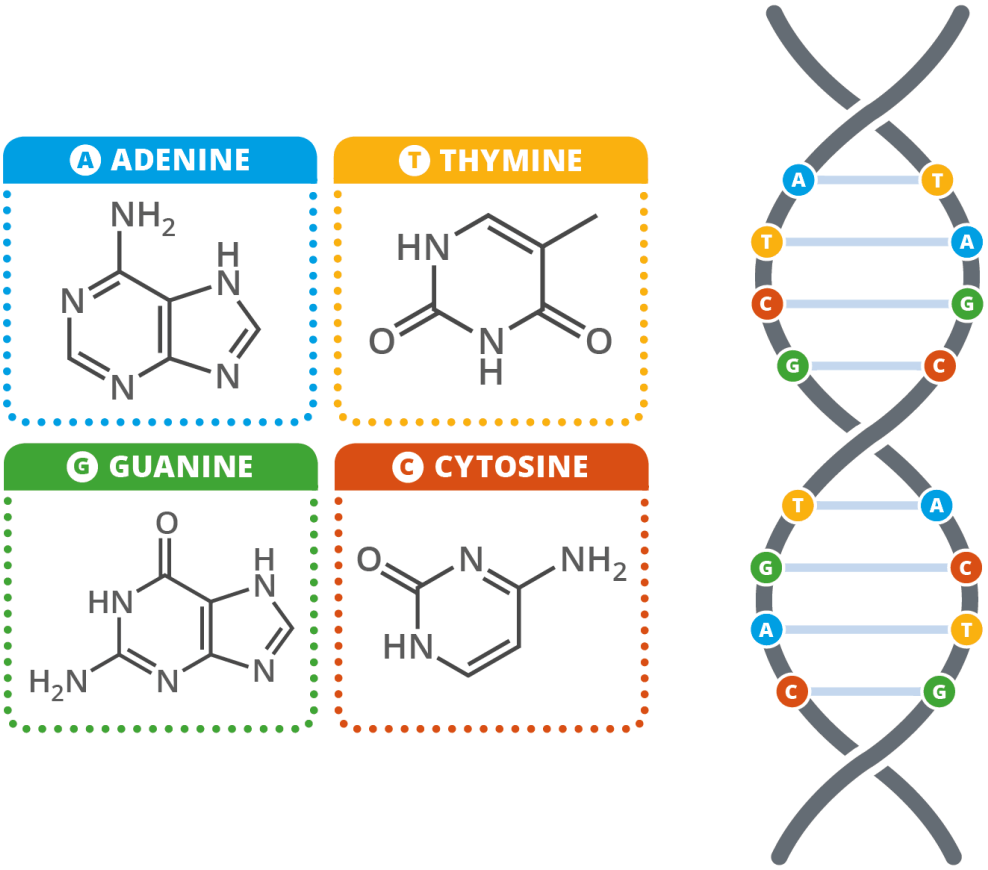


Figure 2: The chemical structure of DNA

A genetic mutation is a permanent alteration of the DNA sequence. It is important to note that they range in size. There are many reasons that cause DNA to mutate, but as in our simulation we have chosen to implement an asexual reproduction method, our mutation will be coming solely by chance. In this case, the offspring are an exact copy of their predecessors, with a small chance of mutating.

In the next section, we will speak about reproduction, and will review how different types of reproduction work at a genetic level.

c) Reproduction

There are two ways for an organic living form to reproduce: sexually and asexually, with completely different repercussions at a genetic level.

On one hand, asexual reproduction is a mechanism in which an organism creates a very similar or identical copy of itself, without the need for genetic material from any other organisms. It is the simplest process of reproduction out of the two, and the most primitive. It is the method adopted by most microorganisms, although there are also bigger animals that reproduce this way, like jellyfish. These organisms often do not possess genders, they can simply divide into two or more organisms.

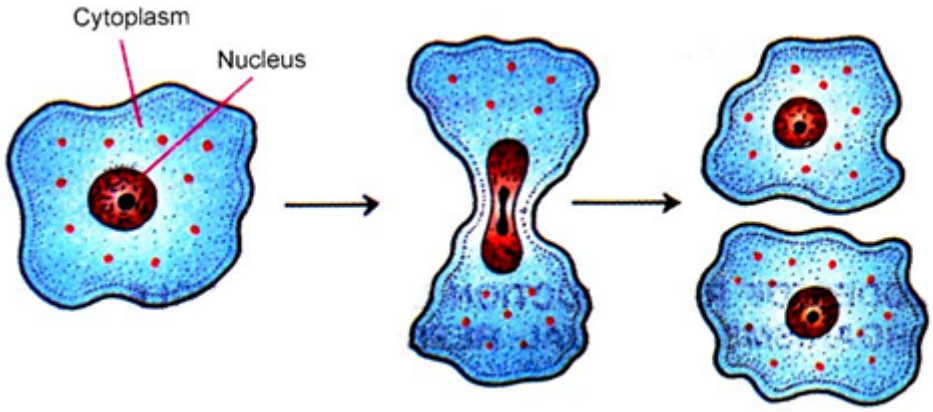


Figure 3: Asexual reproduction of a unicellular organism

On the other hand, in sexual reproduction, for the creation of new individual, the combination of the genetic material of two different organisms is required. Most animals in the planet reproduce this way.

It is worth mentioning that organisms that reproduce through asexual reproduction tend to grow in number exponentially. It is much more optimal if growth of the population is a priority. It also represents lesser risks for the individuals, that do not have to look for a mate.

In general, sexual reproduction requires more energy, and usually a longer and more complex process, with a higher risk for the individuals taking part in it. It also tends to cause competition within the same species for the females, which in some cases results in mutilation and death. The process is much simpler in asexual reproduction.

The main drawback with asexual reproduction is that, as it only requires one individual's

genetic material, species tend to have similar vulnerabilities. Also, genetic mutation by itself is significantly slower than the genetic crossover that takes place in sexual reproduction. That is why, while asexual is more energy efficient, sexual reproduction is much better at adapting to the environment.

In the next section, we will review some concepts related to energy, mass and calories that are crucial for the understanding of ecosystems.

2.2 Physics: Energy, Mass and Biomaterial

On the one hand, as Albert Einstein described in his theory of special relativity in 1905, there is an equivalence between mass and energy, represented by his most famous equation: $E = mc^2$. According to this equation, the energy E of a physical system numerically equals to the product of its mass m and the speed of light c squared. We also have to take into account another really important statement that Einstein made: Energy cannot be created or destroyed, it can only be changed from one form to another.

On the other hand, energy is the most fundamental basis of life. It is stored by cells in biomolecules like carbohydrates and lipids. This energy is released during cellular respiration, which is the very reason why almost all organic living forms need oxygen to survive.

Although these approaches are quite theoretical, we can extract some ideas from the relationship between energy and mass described by Einstein and the need for energy of organic living forms. The way most animals get energy is through their digestive systems, which are designed to break down certain materials and chemical compounds to extract energy, which can be used or stored in the body as lipids. There are thousands of different species of animals, typically categorised in three main groups, depending on their diet: herbivores, carnivores and omnivores. They have adapted to different environments in a way that their digestive systems are able to digest and extract energy from a wide variety of sources. Most species though, have specialised in digesting carbon-based organic materials, most likely because its relatively easy through the right chemical reactions. We

use the energy measurement unit of the calorie to quantify the extractable energy in carbon-based organic materials. It is important to note that two different species may not extract the same number of calories from the same source. It heavily depends on the biological adaptations of the species. As an example, even if both a bear and a cow can feed upon certain plants, the ruminant anatomy of the second will mean that the cow will extract significantly more energy from this source than the bear could.

In our project, we will need a way of representing energy and the exchange of energy in the form of calories that occurs when an individual feeds upon another individual, or during reproduction. We will represent this relationship between energy and biological mass with biomaterial, which will be a unit used to describe the amount of energy contained in a system which is ideally extractable. Depending on the adaptations of the individual consuming that biomaterial and whether it is vegetal or animal, more or less energy will be extracted and absorbed.

In the next section, we will review the requirements of the project, the components that need to be modelled and how they could be implemented through different technologies.

2.3 Requirements of the project, traditional algorithms and introduction to artificial intelligence

As we mentioned in the evolution theory section, the three main concepts that we need to take into account are the principle of variation, the principle of heredity and the principle of selection. But, to fully understand the complexity behind implementing these principles in the system, we first have to speak about traditional algorithms and their basic mechanics.

Without going into too much detail, traditional algorithms, or simply algorithms, are a series of well-defined steps, typically performed by a human or a machine. That is the very reason why these types of algorithms are not the best approach in order to simulate organic evolution. A better approach would be to use artificial intelligence to achieve our goal. Even though strictly speaking, artificial intelligence is built as a compound of

traditional algorithms, most artificial intelligence implementations share a common characteristic: automatic learning and evolution over time, making this approach a very competent candidate to build the core of the system.

When it comes to modelling the three principles, the implementation of the principle of variation is trivial. We just have to build a module capable of generating a set of individuals with randomly generated attributes.

The modelling of the principle of heredity is where the complexity level raises. Predecessors need to transmit their traits to their offspring. To do that, we need to define a structure where all of that information about an individual can be stored and easily accessed by the simulator's methods and mechanisms. We will call this data structure the genome model. We also need to define the necessary components that are capable of handling the genome mutation during reproduction.

Lastly, for the modelling of the principle of selection, we need to build a system where the individuals taking part in the simulation can interact with the environment and with each other. It's also crucial that their behaviour has a direct impact on their survival. This way, we would simulate natural selection. We also need a solid genetic model that allows individuals to pass down traits to its offspring as genetic heritage, which will make them resemble their predecessors more than other individuals of the same species.

Another necessary mechanism is the module that will act as the decision maker of the individual, a very primitive brain. We considered that the best way to model this was to use artificial neural networks, as they are a simulation of an interconnected collection of neurones that operate emulating a biological brain.

In the next sections, we will delve into artificial intelligence and neural networks, focusing on a branch called genetic algorithms, the core component of this project.

2.4 Artificial Intelligence: The learning process of a machine

a) Background

Since modern computers were invented, their huge potential performing various tasks has been clear and their computing power has grown uninterrupted. Artificial Intelligence is the science of making software think intelligently, in a way that mimics the way humans think. It is certainly a multidisciplinary field, based on disciplines such as Computer Science, Mathematics, Engineering, Biology, Linguistics and Psychology. In fact, artificial intelligence has evolved so much over the last decade partly and based on the study of the human brain, how it learns, decides and functions.

There is no limit for the practical applications of AI. Mankind is at a point where every two days, we create as much information as we created from the dawn of civilisation up until 2003. The volume of information available is incredibly huge, and it is not consistently organised, due to its many different sources. And not only that, but every second, new information is being created. This a great example of the potential of artificial intelligence and a real-world application, analysing and processing all of that raw data automatically and extracting some useful knowledge, at a fraction of the cost and time compared to traditional methods. It over performs traditional algorithms, (if there was even a solution that did not required AI before) and any other methods in execution time and use of resources. It is also incomparably faster than a force brute approach.

The key of artificial intelligence is not only that it tends to be the optimal solution in complex problems, but that, depending on the implementation, a machine learns from experience, it modifies itself in order to adapt to the situation and complete a task successfully. This particular approach where machines learn for themselves when given a collection of data is called **Machine Learning**, and it is present in most AI implementations. Programs can absorb modifications by processing different pieces of information highly independent, and automatically improve their methodologies, effectively learning by themselves. That fact makes the **Machine Learning** a perfect fit for the project, as an algorithm based on it would make the evolution of species possible.

b) Machine Learning Classification by Type of Learning

There are many ways to implement **Machine Learning** in a system. The first way in which these different approaches can be classified is depending on their **type or learning**.

- Supervised Learning: A set of data called training data, which has previously classified or tagged with the the correct result is necessary. The program analyses and processes the data and builds a model accordingly. The next step is the actual training of the system, where more information is given (it is also required that this information was previously tagged correctly) and check if the returned results are correct. It is a requirement that when those results are wrong, some sort of feedback is sent back to the system in order for it to automatically correct its previous model. The process continues until the program reaches the desired level of accuracy. An example of a real-world use of this type of learning would be in Logistic Regression. One of the most popular examples is to calculate the probability of passing an exam, based on the hours of study. Because of the nature of our project, we can discard this type of learning. Individuals in an ecosystem are not trained with an independent training set and they do not receive external feedback, they must learn on their own. This crucial requirement takes us to the next type of learning, the semi-supervised.

- Semi-Supervised Learning: It works in a similar way, but in this case the input data is only partly tagged. The system must build a model around the tagged data, and make predictions when dealing with new data. In most implementations of this type, the accuracy of the system is greatly improved if direct feedback is received in order to modify the original model. An example would be image classification, as typically the datasets tend to be large, with few previously classified pictures. The same main problem remains. The system could be modelled following this approach, but it would not resemble biological evolution in its core.

- Unsupervised Learning: In this case, the input data has not been previously labelled, and does not have a known result. The system builds a model based on some general rules, previously established or through a predefined mathematical process. Even though no direct feedback is required, the system has to have some way of evaluating

the success of the prediction, otherwise it would not be possible for it to learn and improve its model. An application of this type of learning is to solve clustering problems, where it is required to design a procedure that classifies a set of vectors, according to predefined criteria, typically distance or resemblance between them.

This learning method resembles biological evolution a lot more. There is still the need for a scoring system, to decide which have been the best predictions in the system. Luckily nature has a way of rewarding the best performers: the lifespan, which is directly connected to the amount of times that the individual can reproduce, which will impact the population, successfully simulating natural selection.

c) Machine Learning Classification by Category

There are many ways of classifying machine learning algorithms, and many algorithms could be fitted into more than one category. For that reason, the next classification has been made based on their basic structure and mechanism.

- Decision Tree Algorithms: In this method, the decisions are taken based on very specific values of attributes in the data. Decisions fork in tree structures until a determined point is reached and the best decision is returned. Decision trees are fast and tend to be accurate, making them a big favourite in machine learning. They are especially useful in situations where the possibilities are low to moderate for each decision fork. As an example of Decision Tree Algorithms, they are typically used to build chess AI players. Following a brute force approach, for each move, the system recognises every possibility, forks them and repeats the process until it reaches a predetermined depth. Finally, it would return the move with the best score, meaning the best potential move. The search depth is based on the computing power available. In this case, the algorithms would have to explore the whole decision tree, but there are a lot of methods better than the brute force algorithms that we just described. In the Alpha Beta Pruning approach, the system just forks the decisions that are potentially better, discarding significant parts of the decision-making tree in the process. While this is a fast and accurate solution to solve problems like chess, its inviable to solve other types of problems, even the variations like Alpha Beta Pruning. For example, to build an AI

capable of playing the ancient Chinese board game *Go*, a different method needed to be used. Unlike in chess, for every decision that has to be taken in *Go*, there are hundreds of different possibilities. That means that exploring the decision-making tree until a significant depth in order to make a reasonable decision, would simply be too much. It would take too much computing power to reach a point where the algorithms had enough information to make a well-informed decision. In fact, in a nineteen by nineteen standard go board, there are more ways for a game to play out than there are atoms in the universe. In some cases, like *Go*, it is not about getting a more powerful computer, it is about looking for an alternative approach.

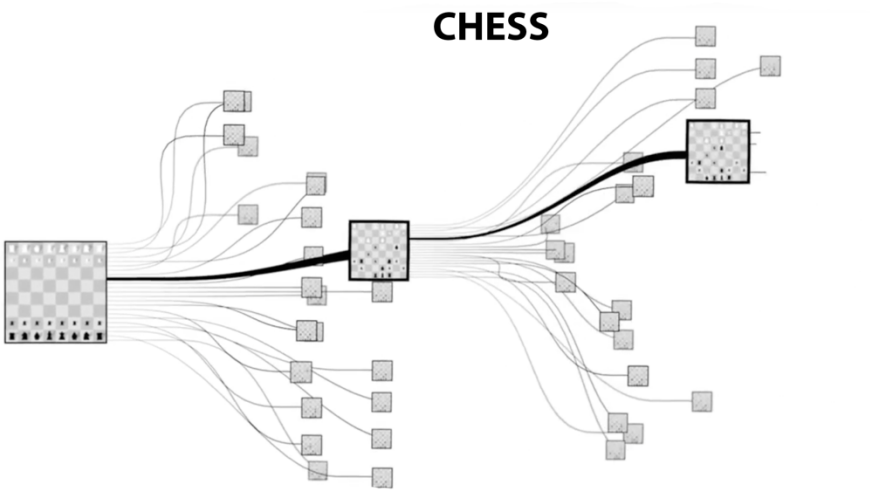


Figure 4: Chess decision tree

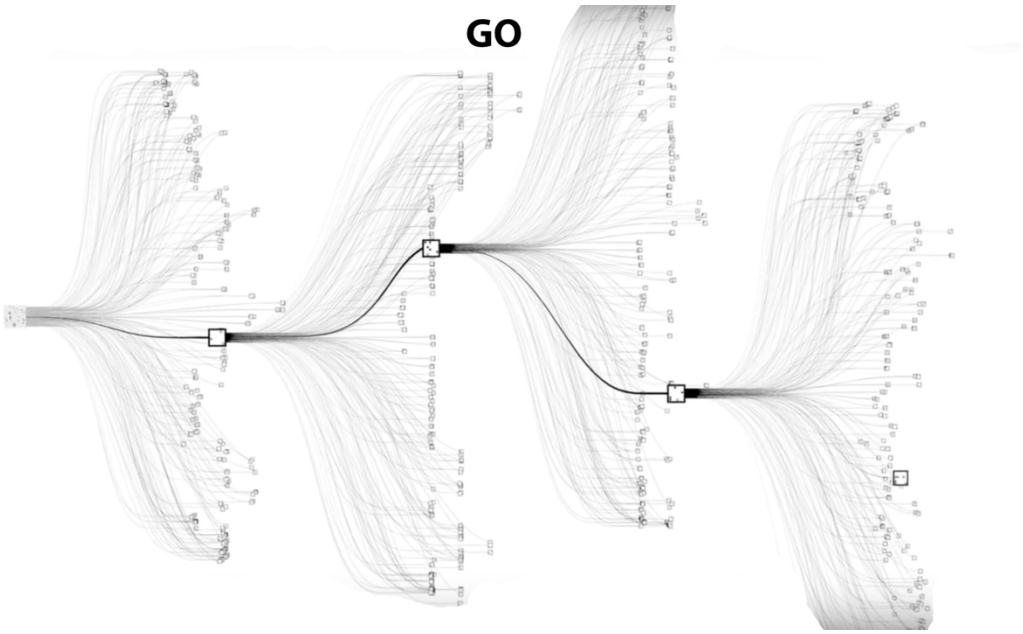


Figure 5: Go decision tree

- Regression Algorithms: Regression analysis is the modelling of the relationship between variables. This is achieved iterating, using a measure of error in the predictions made by the model. It would be interesting to use these types of algorithms for further exploration of the relationship between different attributes of the individuals participating in a simulated ecosystem, and how that relationship might affect their overall lifespan.

- Bayesian Algorithms: They are those that apply Bayes' theorem. They are mostly used in problems such as regression analysis, and classification.

- Clustering Algorithms: While there are a lot of different implementation of clustering algorithms, all methods are focused on organising data in groups by similarity. They achieve that using their inherent data structures.

- Instance Based Algorithms: A solution applied to decision problems where a set of previous examples are required. Typically, the most common implementation of this algorithm is through the creation of a database which contains a set of examples. That way, the algorithm makes a prediction comparing the new information to the one in the database.

- Association Learning Algorithms: The goal is the extraction of general rules that explain the relationship between variables in a given data. This algorithm is broadly used by companies with large datasets and want to extract useful knowledge that can help make well informed business decisions.

- Artificial Neural Network Algorithms: They are models inspired by the structure of biological neural networks. Their main goal is the pattern matching and are used to solve problems of all sorts, like perception or image recognition. As they are based on biological neural networks, it is a really strong contestant to become the structure in which the core decision maker for the individuals in the simulated ecosystem will be built around.

- Deep Learning Algorithms: A modern implementation of artificial neural networks that focus on the use of now cheaper computation power. The goal of the algorithm is to build much larger and more complex neural networks.

- Dimensionality Reduction Algorithms: The objective of the algorithm is to find inherent structures in a given dataset, in an unsupervised way. It is used to get a description of the data, simplifying the trivial components and emphasising the relevant ones.

- Ensemble Algorithms: Unusual method where the overall system is composed by a series of components which are independently trained. In order to return a prediction, the system combines the results from all of the different components.

- Genetic Algorithms**:** This type of algorithm was created mimicking biological evolution. The goal is to use the power of evolution to solve optimisation problems. Genetic Algorithms are a perfect fit for our project, as they use a virtual genetic model to store the individual's information, and they implement the concepts of natural selection, mutation and heredity. Genetic Algorithms have been an important part of this project, as they are the base on which we have built the platform. That is why in the next section, we will delve into the topic, explaining in detail their structure and the way they function.

2.5 Genetic Algorithms

a) Introduction

As we briefly mentioned in the past section, Genetic Algorithms were designed in the early 1970s by John Holland, to resemble biological evolution. Many scientists and biologists at the time were shocked by the level of complexity presented in nature and the relatively short time that took for it to evolve. John's goal was to design a method that resembled evolution and could solve optimisation problems.

b) The Algorithm

Genetic Algorithms are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetics. As in evolution, the main tool used for evolution is trial and error through mutation, but this process is by no means random. They use historical information to direct the search into the regions where the overall performance within the search space is better. As we mentioned in the first section, genetic algorithms are based on the biological principle of the survival of the fittest. In nature, competition among individuals results in the survival of those who have better adapted to the environment. Because those who survive longer have a better chance to reproduce, the next generation is primarily based on those who survived longer, making them a better fit for the environment.

According to John's work, in comparison to other Artificial Intelligence Algorithms, Genetic Algorithms are robust and they can support slight input variation without collapsing. Even though they might be outperformed at some tasks by modern algorithms, they also support noise reasonably well.

After seeing the biological principles of evolution in the first sections, the basic way in which genetic algorithms operate will come as no surprise. They simulate the survival of the fittest individuals over generations in order to achieve the best possible policy or

solution for an optimisation problem. Every generation has to consist of a population of individuals with a set of data, analogous to the genetic information found in DNA.

Each individual represents a point in a search space, and is potentially the best solution. Those points form the population, which is subjected to a process of evolution. These types of algorithms are based on an analogy with the genetic model and behaviour of DNA within a population of individuals using the following main rules:

- Individuals compete for survival, resources and reproduction.
- The most successful individuals will produce more offspring than those individuals that perform poorly.
- As the better performers will reproduce more, their genes will propagate throughout the population over the generations, eventually producing individuals that perform even better than their predecessors.
- This is an iterative process, where each generation will be potentially better suited to survive in a given environment.

As we introduced in the first part of this section, a set of individuals with specific attributes is maintained within a search space, that defines frontiers for the algorithm. Each of this individual's attributes correspond to a potential solution for the problem. Each individual's attributes are coded as a data structure, typically as a finite length vector of components, in terms of a predefined alphabet, binary alphabet being the most popular. The idea is that this stored data resembles the genetic material contained in organic DNA. In the most popular implementation of these types of algorithms, the variables stored by each individual are called genes and the individuals are called chromosomes. This way, we can say that a chromosome (a solution for the problem) is composed of several genes.

The system must have a way of assigning a score to each of the chromosomes, a score that represents the performance of an individual in a given environment. In Genetic Algorithms, this is typically achieved through a fitness score function, or simply fitness function.

The system is built to recognise which are the individuals with a highest fitness score,

and breed them to generate a set of new solutions. A population of individuals being controlled by a chromosome will be scored through the fitness solution. Highly fit solutions are given more opportunities to reproduce, so the offspring inherits characteristics from each parent.

It is interesting to mention that, as in organic living forms, John claimed that as parents reproduce and produce offspring, there needs to be more room for the new arrivals, since in his implementation, the population is kept at a static size. The parents die after reproducing and are replaced by the new solutions, eventually creating a new generation once all mating opportunities in the old population have been exhausted. This way, new-borns do not have to compete with older individuals for resources as much, and the competition is limited to the new generation.

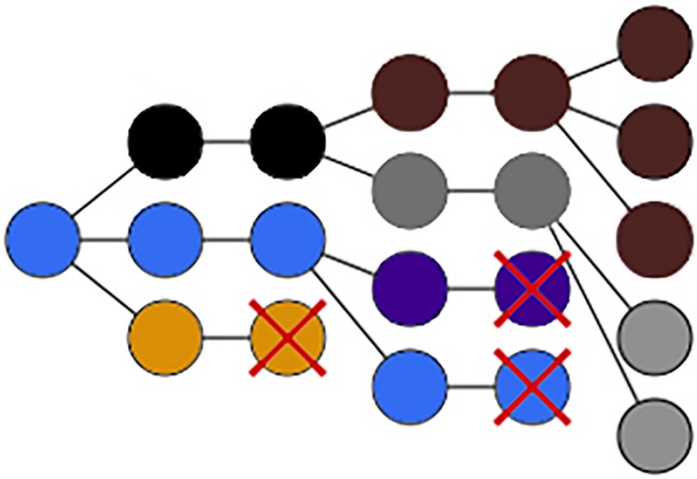


Figure 6: Representation of the evolution and mutation of species

With this approach, on average, newly created generations of solutions potentially contain better fitted genes than older solutions. Each generation will contain more good partial solutions than previous generations, until the population has converged and is not producing offspring significantly different than previous generations. When that happens, it is said that the algorithm has converted to a set of solutions for the problem. Those solutions are the optimal policy, corresponding to the best performers in the given environment.

There are various ways in which the concept of genetic algorithms has been implemented.

Most of them share the same basic concepts, but there are significant differences that are worth mentioning.

The first big difference between implementations is the genetic model that composes the genes and chromosomes of the individuals. The most common choice is to use a set of variables that represent certain attributes which depend on the behaviour of the individual. There are also other implementations that try to resemble nature as much as possible, and for that reason, use a set of characters or values that act like the four chemical bases that we mentioned in the biology section: adenine (A), guanine (G), cytosine (C), and thymine (T). This way, while not being as optimal as using data structures with a set of variables that act as genes, it is definitely the closest option to genetics.

Another decision that needs to be made is the reproduction method. As we saw in the biology sections, there are mainly two reproduction methods. There are major differences between the two on a biological genetic level, which will impact directly in the algorithm's implementation.

To emulate sexual reproduction, the genetic material of two individuals is needed in order to reproduce. For that reason, the system needs a crossover operator, a method that receives two chromosomes and combines them, with a random seed and returns the chromosome corresponding to the parent's offspring. The reason why this is implemented with a random seed is to resemble biological evolution, where just one of the spermatozoids will reach the ovule, each of them with very different attributes and genetic material. As it would take too much effort to simulate it, and the results nor the efficiency of the algorithm would vary, we simply randomly mix the chromosomes from the parents to build the successor's chromosome.

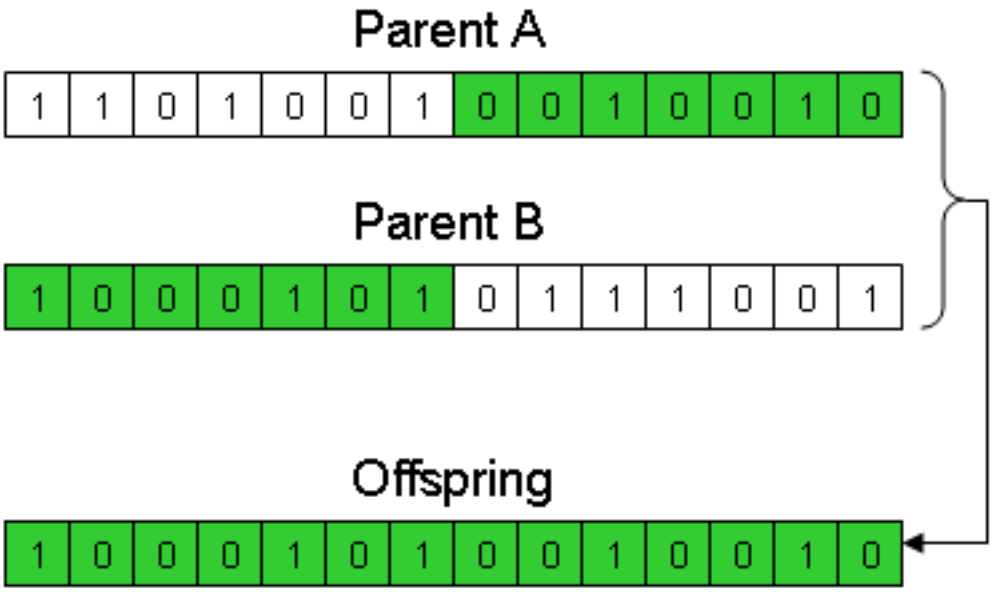


Figure 7: Artificial Representation of Genetic Crossover

When it comes to asexual reproduction, the approach taken is slightly different. As in biology, when an individual is ready to reproduce, it will simply create a copy of himself. This new individual will present some minor differences in its genetic material, caused by mutation. To emulate this, in implementations with asexual reproduction, the offspring is generated taking the progenitor's chromosome as a base and mutating it. As natural selection and mutation are the only mechanisms of adaptation, the mutation rate is usually defined as a much higher value than we can observe in nature.

Just applying natural selection to a population, without mutation or genome crossover, the population would tend to be filled with copies of the best individual. Adding crossover operations to natural selection will make the algorithms converge on an acceptable but non-optimal solution. Using just mutation, we would get a walk through the search space as a result. Using natural selection and mutation, we achieve a noise-tolerant algorithm that will converge on the optimal solution, but it will be greatly impacted by the size of the search space, scalability being one of its drawbacks.

c) Implementation

Anyhow, using any of the methods described below, the structure of the algorithm

remains the same, defined in pseudocode below:

1 Randomly Generate population
2 Determine fitness score of each individual of the population
    [**While** New Generation's Chromosome $\simeq$ Last generation's Chromosome]
    1 Select best performers from population
    2 Breed selected individuals, mutating their chromosomes
    3 Determine fitness score of each individual of the population

This is the basic structure that needs to be followed in order to implement these types of algorithms, regardless of the type of reproduction and the data structure chosen to store the population's chromosomes.

To sum up, genetic algorithms' methodology is based on the creation of a population of individuals that will evolve towards the optimum policy or solution. The reason why this approach is a fast, useful and robust method is due to the fact that it combines the direction and chance in the search for the optimal solution. The population also contains much more information than simply the individual's fitness score, as genetic algorithms combine hidden information within a solution with another solution's information to produce even better information inherited from the parents, or simply naturally evolves towards it through natural selection and mutation. For all of those reasons, and the high noise tolerance, genetic algorithms are a powerful optimisation technique.

After studying genetic algorithms, we can claim that they are clearly a perfect fit for our project. Not only do they resemble natural selection, but they were designed to mimic organic genome mutation and reproduction. For all of these reasons, the platform has been built around these types of algorithm.

Once we decided the basic structure of the platform, we needed to look for a controller that acts as a brain for the organisms. In the next section, we will dive into neural networks, a potential candidate.

2.6 Neural Networks

a) Introduction

The field that studies neural networks was established way before the invention of computers, and has benefited significantly from the inexpensive computer simulations that are at our fingertips today. The first artificial neural network was built in 1943 by Warren McCulloch and Walter Pits but back then, their research was greatly impacted by technological limitations, they didn't have sufficient computation power. Now, technology has advanced to the point where computation power is much cheaper than back then. Neural network based systems have gained in popularity, and are used by the biggest companies in the world to execute heavy computations and large data classification. Deep learning is a modern implementation of artificial neural networks that focus on the use of that newly acquired computation power on a large scale. building much larger and more complex neural networks.

Artificial Neural Networks are an information processing method that was designed to resemble the way biological nervous systems process information. Neural Networks are composed of a large collection of information processing elements called neurones. The idea is that they work together to solve problems.

They learn by example. They are configured and deployed for a very specific application, such as data classification or pattern recognition or computer vision. Through training, the connections between neurones are built and modified, improving the efficiency of the network.

b) Advantages

The reason why neural networks have become so popular is for their ability to extract meaning from extremely complex or imprecise data. Their main use is to recognise patterns and trends, too complex to be detected by humans or too costly computationally for other types of algorithms. Once a neural network has been trained, it is extremely

efficient at processing the type of information that it has been trained for.

Some of the advantages of using neural networks, as opposed to traditional algorithms are:

- It is able to do computations in parallel, especially if the hardware platform is designed with the intention of running these types of algorithms.

- It is an extremely decentralised system, meaning that the destruction of a part of the network leads to the corresponding performance reduction and not stopping completely.

- It learns how to do a task solely based on data given during training. It creates its own structures to classify that information, and it does all of these autonomously.

And these three points are the key on why neural networks are far superior than traditional algorithms at solving specific problems and performing certain tasks. Using a traditional algorithm, the programmer would have to define a set of instructions in order to solve a problem. Those instructions tend to be very specific and have to be modified if some aspects of the problem vary slightly and the algorithm has not contemplated that possibility. Neural Networks though, are much more dynamic and can adapt to many of these changes if trained for it. In fact, they process information in a very similar way the human brain does.

c) Structure of the Network

Artificial Neural Networks are composed of a set of interconnected components, each one able to process information independently and in parallel. These components in the model are called neurones, and reference the neurones in the human brain. As we mentioned at the beginning of the section, these networks learn by example, so they cannot be programmed to perform a specific task. They are given a set of information carefully selected so its relevant enough to cause the network to modify itself and learn.

The system will not stop working if the data is not carefully selected, it just won't learn as fast as it could, but it is worth mentioning that if the data is not consistent or not varied enough, the network might end up functioning incorrectly.

Conventional algorithms use a cognitive approach to problem solving. The solution of the problem must be known by the programmer, for it to be implemented and included in the algorithm's set of instructions. They are completely predictable. In case of malfunction, its caused by a software bug or a hardware failure. The main problem with this approach is that, as with many problems like image recognition, it cannot be modelled. The programmer is able to tell, for example, if there is a cat in an image, but he is not able to define pixel by pixel, all of the colours, angles and shapes that a cat can be represented by and define them in a system.

Neural networks are not the best approach to solve everything. Conventional algorithms are much better at solving arithmetic operations, or solving problems, well defined from the beginning. They are even used to supervise the training of neural networks, for maximum optimisation.

Although the design of artificial neural networks is primarily based on the human brain, there is still a lot that we don't know about how the brain trains itself, so we have had to fill the gaps with theories and approximations. What we do know is that a typical neurone receives signals from other neurones through a structure called dendrites. Those signals are based on spikes of electrical activity through a thin connection called axon, which splits into hundreds of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical signals that inhibit or excite activity in the neurones. When a neurone receives excitatory input that is large enough based on its inhibitory input, it sends a spike of electrical activity down its axon. The biological neural network learns when the effectiveness of the synapses change. This causes the influence of one neurone over another to be modified, and the neuronal circuitry changes, potentially being able to adapt to new situations.

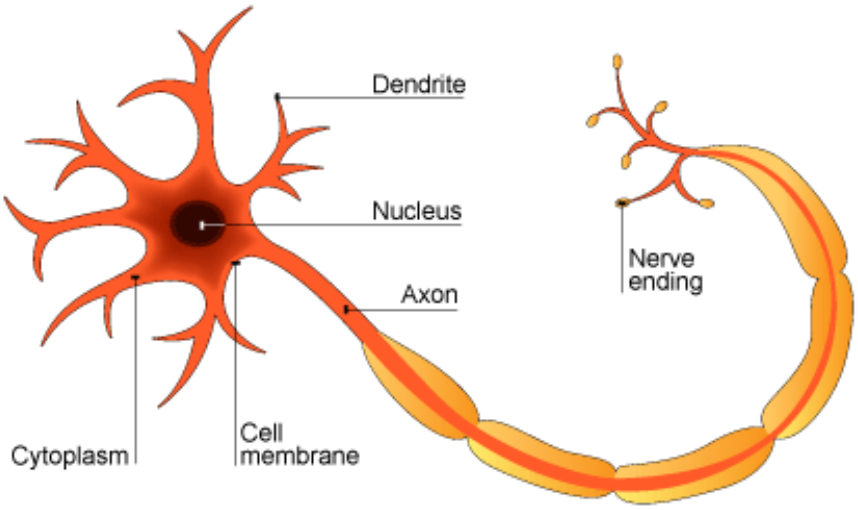


Figure 8: Components of a biological neurone

The first step when building an artificial neural network is to try to deduce the essential features of neurones and their interconnections but, as our knowledge of biological neural networks is limited and far from complete, and the computing power available is significantly lower than the equivalent computing power of a brain, our implementations are approximations. If we fully understood the way the brain works, artificial neural networks could be significantly improved.

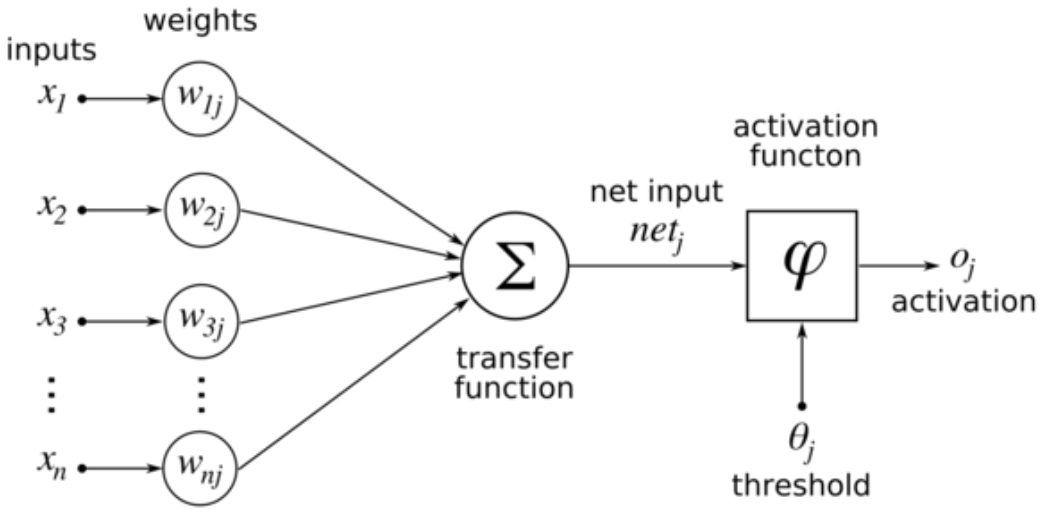


Figure 9: Components of an artificial neurone

d) How it works

The way an artificial neurone works is quite simple. An artificial neurone is an entity with many inputs and one output. It has two modes of use, the training mode and the execution mode. In the training mode, the neurone is trained to activate with specific inputs. During training, the neurone is trained to recognise certain patterns in the input. In the execution mode, the neurone uses its current knowledge and gives an output for each input. The firing rule is used to determine whether or not the neurone needs to fire. These rules are core when building neural networks, and the key of their high flexibility. A neurone has to fire for any input pattern that matches its training, and not just for the ones that it has specifically been trained for.

As the implementation of these neural networks is not a trivial process and in the project, we have designed a very simple network, we won't get into detail in the technical part. We will just say that every connection between inputs and neurones has a weight that represents how probable it is to fire at a given input. It is also important to know though, that there are different types of learning:

- Associative mapping, when a network learns to produce a pattern on the set of input units when another particular pattern is applied on the set of input units.

- Regularity detection, where neurones learn to respond to particular properties of the input patterns. The response of each neurone has a particular meaning.

As we have seen in this section, neural networks have a huge potential at solving pattern recognition or data classification. Also, because it mimics a biological brain, it was very appropriate to include a simple implementation of a neural network in our project. It matches the requirements of the individuals' decision maker module, the piece that connects the virtual genome from the genetic algorithms and the direct behaviour of the individual in the simulation.

3. Implementation: Tools and Methods

3.1 Core of the Platform

a) Introduction

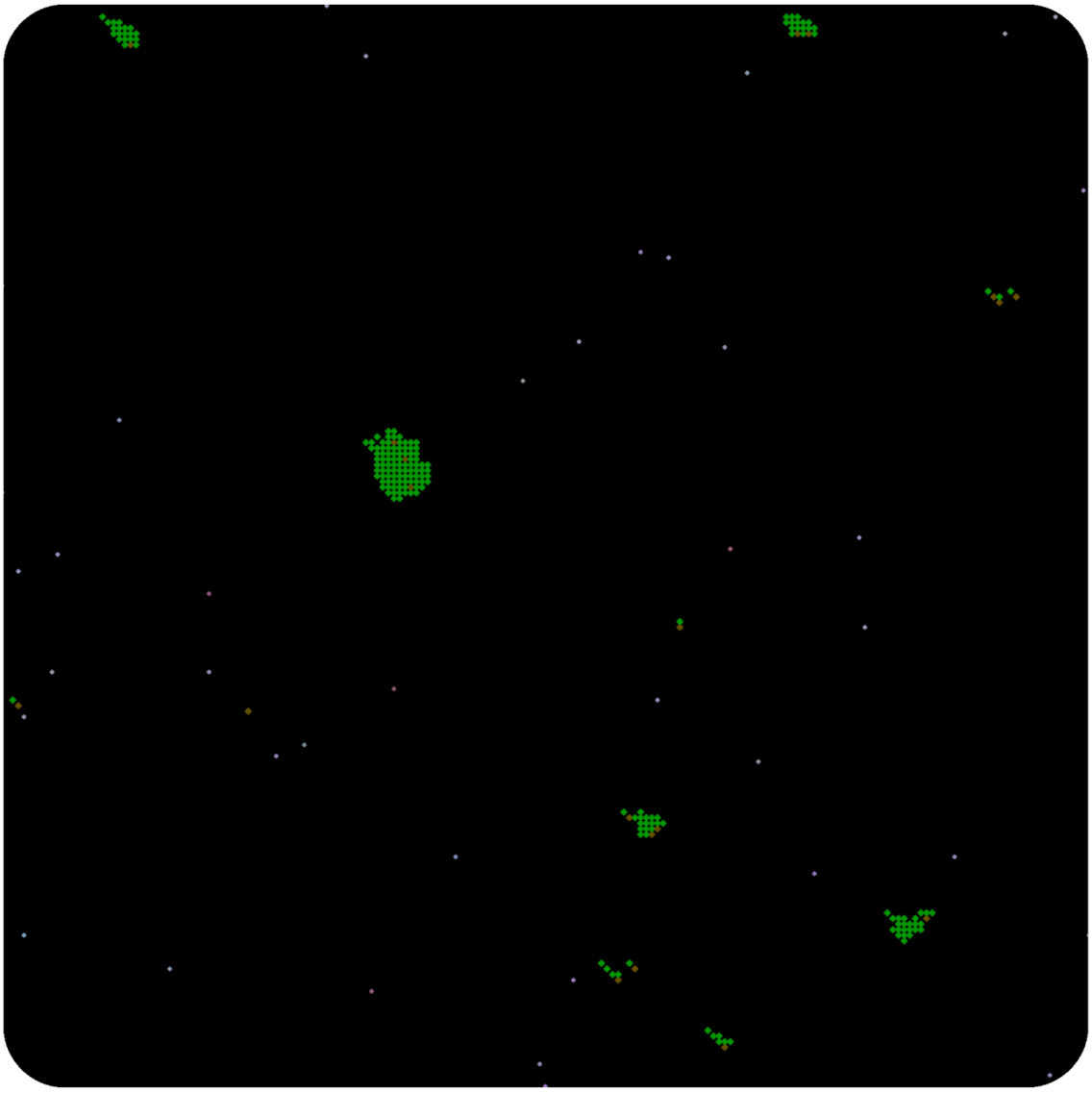


Figure 10: Simulator's graphical interface

The main goal of this project was to build a platform capable of simulating ecosystems, and we have done that through genetic algorithms and neural networks. Our intention was to design a system that resembles biology as much as possible. As we have seen in the second chapter, genetic algorithms are based on the concept of natural selection (survival

of the fittest), represented by a genetic model. That is the main reason why we decided to build the platform, taking this concept as a foundation. We also needed a module that acted as a connector between the genetic model presented in the genetic algorithms and the behaviour of individuals in the simulation. A simple implementation of a neural network was our approach to extract the information from the genetic model and directly impact the actions of the individual accordingly. We designed this so the values of the variables stored in the genome model corresponded to the weights of our implementation of the neural network.

b) Participants

There are two types of entities that participate in the simulation. The first type are the animal organisms, which have been the focus of the artificial intelligence implementation in this project. The second type are the plant organisms, governed by a simple algorithm that results in predictable, configurable behaviours. These two types of participants will be represented by a small dot on the screen. The colour of animal organisms will vary, as is linked to their genome, and will change as the simulation advances and the organisms evolve. The plant organisms will always remain the same colour, dark green when they are young and brown during their adulthood, representing soft leaves and tougher, tree wood.

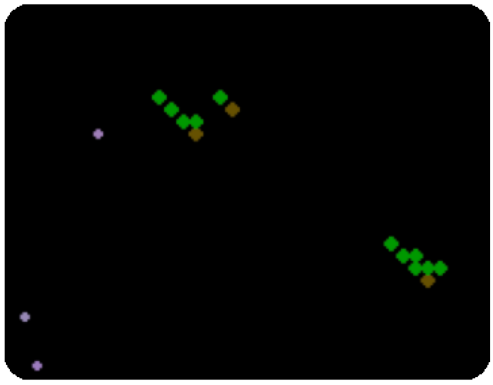


Figure 11: Closer look at the simulation participants

The simulation takes part in a delimited area that we have called the grid, an invisible data structure conformed by individual cells. The size of the grid is defined by the user before the simulation starts, and can be set as visible with debugging purposes. The user will see a graphical representation of the grid from above, and will be able to move its point of view through the arrow keys. In every cell, there can only be one entity at a given point in time. Entities can only exist in cells, and each entity can only habit one cell at a time. If an entity cannot be found in any of the grid cells, is because that entity does not exist anymore in the simulation, either because it has been consumed by other entities or

ran out of biomass, a concept that we will introduce in the next section.

c) Interaction with the Environment

There are three main actions that entities can perform: they can move (or stay still), attack and reproduce.

- Movement: In order for an entity to move to a contiguous cell, the destination cell must be free, with no plant organisms or animal organisms in that position before the movement action is performed.

- Attack: Entities can attack as well. When an animal organism and an entity are in contiguous cells, the animal organism can perform the attack action. The attacker steals a certain amount of the attacked entity's biomass. Only animal organisms are capable of attacking.

- Reproduction: Entities will reproduce when their biomass reaches a certain amount. When that happens, the entity will divide itself into two independent entities, copying its mutated genome into the new-born.

There is an implicit energy cost in movement. For that reason, when an entity moves, a certain amount of biomass will be extracted from it. This is what causes that, eventually, entities that have no access to a source of energy die. Plant organisms are not able to move in the simulation, and they cannot attack either. Instead of moving, it will simply receive a certain amount of biomass. This represents the natural growth of the organism. The amount of biomass that the plant organisms receives can be modified, in order to simulate different environments where the resources are abundant or harder to find.

d) Simulation

At the start of the simulation, a set of plant organisms spawn in the grid, occupying

random cells. This set of plants are identical, and will start growing and reproducing from the very beginning. The first generation of animal organisms also spawns in random cells. This generation is composed by individuals whose genome has been randomly generated. Most of them will die shortly, as they are not naturally inclined towards consuming plant or animal organisms. Other genomes will dictate that they stay still, or that they simply escape. Its normally a very small group of individuals that approach plants and consume them. The best individuals at it will reproduce, and the more advanced the generation, the better fitted the individual will be for the specific environment. After some generations, we can see a clear tendency in some individuals, orbiting towards an herbivore behaviour, while others clearly adopt a carnivore behaviour.

The simulation runs until it is manually stopped. While some species reach a solid genetic model, and change very little from that moment onwards, we can still see unusual strategies coming from spontaneous mutations, that might or might not end up becoming the beginning of a new species. Genetic algorithms are designed as a method to solve optimisation problems, to find an optimal solution for a specific problem. The singularity in the case of ecosystems is that, while in a system as simplified as ours, there are certainly optimal behaviours depending on the individual's general tendencies, there is not a unique solution. For instance, for the simulation to emulate a real ecosystem, there needs to be a handful of different species, and a significant variety of individuals at any point of the simulation. The solution in our case, would be represented by a set of individuals, generated as a result of many generations competing for resources, that would have adapted to the corresponding simulated environment.

We also had to limit the genome's evolution by defining a search space. The entities will evolve until they reach the predefined search space limit. The evolving component is the genome, a data structure that holds a collection of variables. Each one of these variables is linked to a behaviour, that is adopted if certain conditions are met.

The reason why we had to limit evolution by defining a search space where the simulation was going to take place is that, even though in real ecosystems there is no such limit, it wouldn't be possible to build a limitless, fully open system. We only have a handful of variables representing a genome, that produces a significant amount of combinations. In nature, there are an infinite amount of possibilities and potential situations, not only

genetically speaking, but physically as well. We would have to take into account and model the different types of atoms that form an organism and how they interact with each other, chemical processes, biological mechanisms, too many to be represented with our current technology, and many things that we don't fully understand yet. Defining a search space is a way of simplifying the problem and implementing a framework where we can simulate an ecosystem. We will delve into this concept in the next section, reviewing the modelled concepts, the simplified principles and we will also mention some of the omitted ones.

e) Class Diagram

To illustrate the implementation of the different components of the platform, as well as the structure of the code, we have added the class diagram. It is worth mentioning that this diagram shows the latest version of the code at the time that we are writing this paper, but our aim is to keep working on the project.

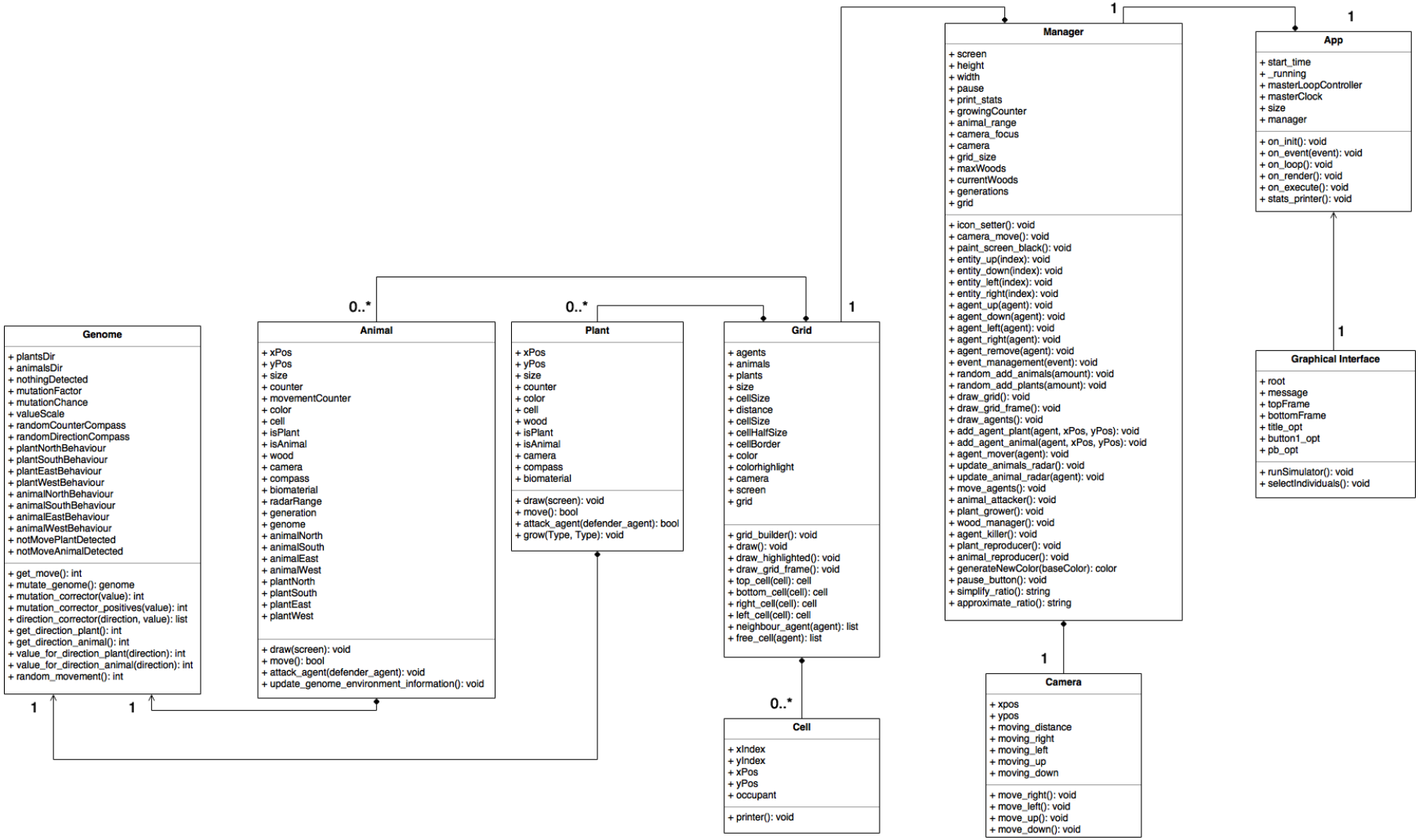


Figure 12: Class Diagram

3.2 Biological Concepts modelled in the simulator

a) Evolution and Natural Selection

In our implementation, we have focused on the modelling of the animal organisms' evolution, while plant organisms were designed to serve as their main energy source. In nature, plant organisms are able to extract energy from the environment, in the form of nutrients in soil, water and sun. Animal organisms, while still being able to get some energy from those sources, they tend to extract most of their energy from either plant organisms or in other animal organisms. The competition between individuals occurs as a result of the limited amount of resources, space and mating partners. The reason for this is that plants are extremely energy efficient, barely spending any energy in other processes other than growing and doing the photosynthesis. Animal organisms though, need a lot more energy, as their activity levels are generally higher. It would not be enough to solely feed upon environment resources.

b) Calories, Nutrients and Biomass

As we mentioned in the energy and biomass section, we need a way of quantifying the energy exchange that occurs between organisms when they feed, when they are attacked, when they reproduce and when they move and grow. In nature, calories are a measure for this exchange, although many other factors play a part in it. For instance, the nutritional requirements of a living organisms can be divided into two main groups: macronutrients and micronutrients. Micronutrients are those that the organism's body needs a minimum amount to function correctly. In this group, we can find vitamins and various chemical components essential for most living beings. On the other hand, macronutrients are the main source of energy for organisms, divided into carbohydrates, fats and proteins. To simplify this concept, we defined a measure unit in our platform that we called biomass. This is a representation of both macronutrients and micronutrients that other organisms will need to survive in the simulation. Every individual will have a variable that will

indicate the amount of biomass that it currently possesses and when the individual moves, it will lose some of it.

In nature, organisms are able to extract energy from two main sources. The majority of individuals and species are herbivores, following a diet based in plant material. There are also carnivores, organisms that derive their energy and nutrient requirements from a diet consisting of animal tissue, either as predators or scavengers. The last type are omnivores, those organisms that have adapted to be able to extract energy and nutrients from both animal and plant sources. It would seem that the best possible option is to be an omnivore, as you would be able to fulfil your needs consuming both sources of food. That is far from the truth, as digestive adaptations play an important role in this biological process, and in our platform.

c) Digestive Adaptations

Digestive adaptations are, as any other evolutionary adaptation, a method that allows species to adapt not only to an environment, but specifically to the available nutrients and energy sources in them. A well-adapted herbivore species that has evolved to consume grass, will be able to extract significantly more energy than an omnivore species. This last species would still be able to extract more energy and nutrients from grass than a fully carnivore species, whom would not get enough to subsist. These type of adaptations is crucial in an ecosystem, and we decided to model them in our simulator. We can see a graph that represents these digestive adaptations, and the amount of energy and nutrients that an individual can extract from different sources. The x axis represents how adapted a species is to extract energy from a specific type of source. A species on the left of the chart, would correspond to an herbivore species. One on the right of the chart would be a carnivore species and in the middle, an omnivore species.

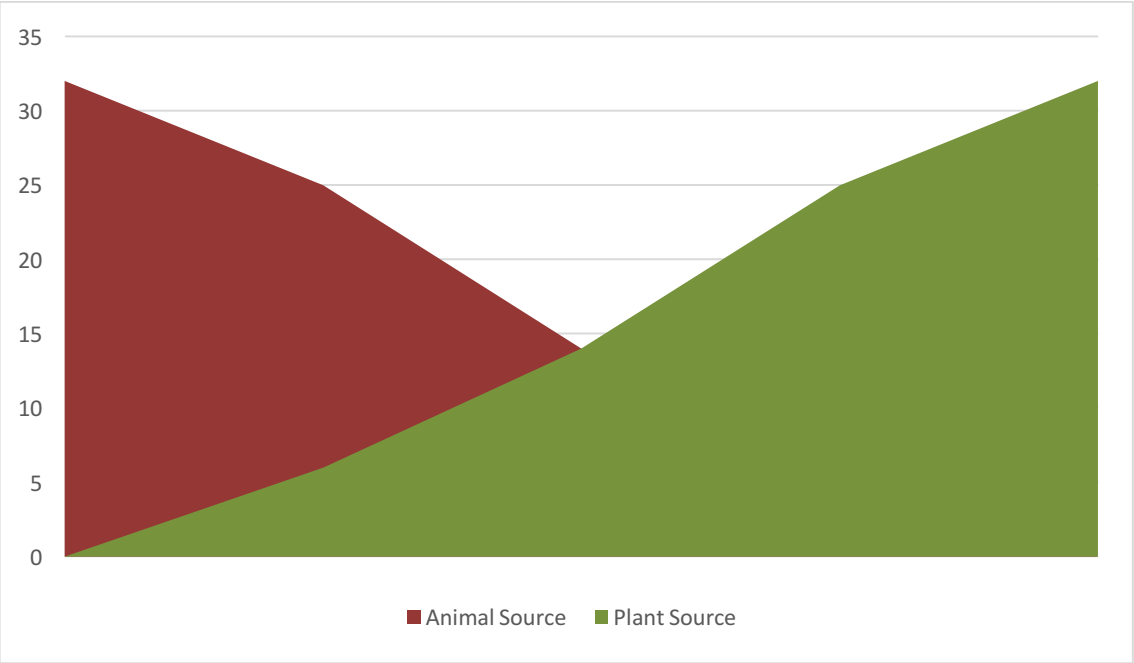


Chart 1: Digestive Adaptations

d) Asexual Reproduction and Mutation Rates

Asexual reproduction is the reproductive process that we have implemented in the platform, mostly common in unicellular organisms. As we saw in the biology section, individuals can simply create a copy of themselves whenever the moment is right. In the simulation, that moment arrives when an individual has enough biomass for it to split into two. The parent provides the new-born with half of its initial biomass. No additional genetic material is needed in this process, the parent's genome is simply copied and, after applying a certain mutation, it is passed down to the new organism. In nature, genomes have a very low mutation rate, meaning that they vary very little from generation to generation. The main reason is that, as the genome is responsible for absolutely everything in an organism, if the mutation rate was too high, too many individual's genomes would mutate too much and would become unstable, resulting in individuals that are not able to survive, or are simply born dead. Because this mutation rate is very low, the evolution of species is a slow process, taking millions of years until we can see significant changes in a species. There is no need to say that we don't have that much time. But there is an advantage in building this type of platform. Our virtual model of the genome is not as crucial for survival as it is in nature. Our model is a collection of

variables that control the behaviour of the individual in certain situations. That is why our genome model is much more resistant to high mutation rates, as it is also much simpler. Defining a higher mutation rate than in nature, we have significantly accelerated the process of evolution.

e) Plant Organisms and the Implementation of Wooden Material

As we will see in the last part of the chapter, we have designed an optimized platform capable of hosting a relatively high number of individuals in a simulation, specially without running the graphical interface. Even so, it doesn't even come close to real ecosystems. That was why we had to make sure that plant material wasn't going to go extinct, not even when there was a super population of herbivores. The measure that we took was to implement a new stage in the life of plant organisms. If the conditions are met, a plant organism will turn into wood. Wooden plant organisms will act normally, just as they did before the transformation. The only difference is that, from that moment onwards, they cannot be consumed by animal organism. The amount of plants that can transform into wooden plants is limited, otherwise they would end up taking control over the simulation. This limitation represents the photosynthetic needs of plant organisms and the space limitations of their roots, making it impossible for trees to grow too close together.

The concept as a whole comes from the mechanisms that plant species have to reproduce and endure. For instance, even if all the plant organisms would be eaten at the same time, there would still be many seeds, roots and traces that would be enough to repopulate the world with plants. Our implementation of the wood material is a simplification of this concept.

f) Species' Behaviour, Tendencies and Neural Networks

Connecting with some of the concepts in the last section about the genome model and how it impacts in the individual's behaviour, it is worth mentioning that, it is composed

by a collection of variables that control their most likely reaction in specific situations. These situations include the detection of another animal organism, a plant organism and no detections at all. Those detections are translated to a series of inputs that are passed to a simple implementation of a static neural network, that processes that information, extracts the data form the genome and returns the corresponding action that the individual will perform. Although this implementation is not complex, it fulfils its purpose, being able to generate individuals with a variety of different tendencies and behaviours. At the moment of writing this, we have been able to clearly differentiate herbivore, carnivore and omnivore species, with significant variations from one to another. We have found that certain species of herbivores roam the simulation space, looking for plant organisms, while others tend to move a lot less, and stay close to plant organisms that have turned into wood, in order to save the energy that movement requires. Carnivore species have taken a similar approach. Some species prowl the area, looking for prey and other species simply minimise their movement until they detect another organism.

3.3 Optimisation Principles

a) Introduction

As we have discussed in other sections, the design of the algorithms was crucial for this platform. The main reason is that the number of individuals in the simulation will impact directly in the richness of the results. If a minimum of individuals is not reached, the population won't even stabilise and life will go extinct. In this section, we will speak of the optimisation principles that we followed when designing the core algorithms of the simulator.

b) Computations vs Memory Usage

Since the invention of computers, technology has advanced a lot over the years, some fields more than others. For example, programmers in the 70s, 80s and 90s really needed to pay attention to the execution cost of the algorithms that they designed, both in terms

of computing power and memory usage. As an example, the Intel 4004 CPU was released in 1971 as a consumer product, with a clock speed of 740 kHz and supported a maximum of 640 bytes of RAM. Transistors have got much smaller since then, improving significantly the computing power of microprocessors, but the storage capacity of electronic devices has increased way more than the processing power. Nowadays, when optimising algorithms, the main target is the execution complexity, and we will try to reduce it as much as possible. In some cases, it is preferable to duplicate information, to take the optimisation further. Summarising, the first principle that we should always try to prioritise is the design of a low execution cost algorithm over a memory efficient algorithm, without neglecting the memory usage.

c) Algorithm Complexity

The second optimisation principle is the importance of the algorithm's execution complexity. The difference is insignificant when simulating small ecosystems, but depending on the way the algorithm is designed, if the execution cost is O(n^2), that means that the scalability is terrible, and we won't be able to simulate big ecosystems. This is a very extreme case, but if the execution cost was really O(n^2), it would relatively quickly require an amount of computational power that wouldn't be achievable, not even using our current supercomputers. Those are the consequences of designing an algorithm that scales exponentially.

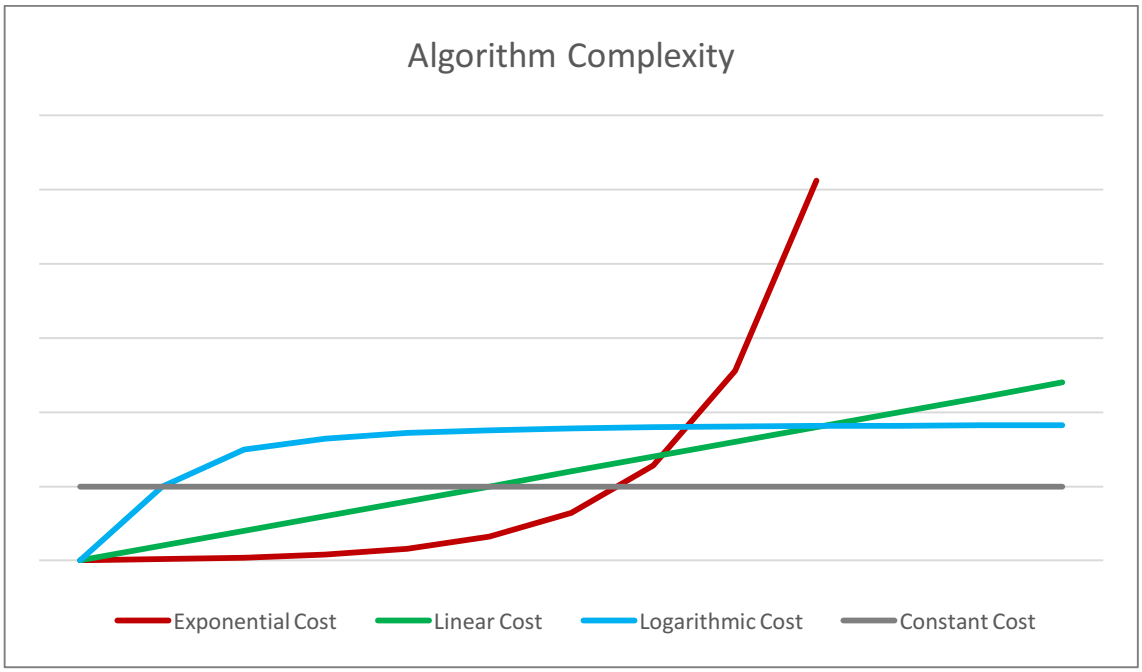


Chart 2: Algorithm Complexity

As we can see in the graph above, the ideal execution cost of an algorithm would be a constant cost. This, of course, is not possible for most problems, as the solution would be calculated in exactly the same amount of time, which means that the same amount of calculations are made, independently from the size of the input of the problem. The general tendency is to design algorithms with a logarithmic execution cost, if possible. An example of an algorithm with a logarithmic execution cost is the binary search, where the main requirement is that the input needs to be in order. Linear cost is also an acceptable execution complexity, where the calculations are made for every element in the input. It is worth mentioning that it is heavily impacted by the slope of the line. If the value of the slope is small enough, the complexity can be similar to the constant cost. The last algorithm complexity represented in the graph is the exponential execution cost, which should be avoided at all costs. To sum up the second principle that we followed when designing the main algorithms for this platform, the exponential cost should always be avoided. The goal should always be to avoid algorithm designs with a complexity of $\Omega(n)$[1].

[1] Algorithm complexity exceeds n (exceeds linear complexity).

3.4 Optimisation of the Platform

a) Introduction

As we mentioned in the last section, our goal when designing the platform algorithms was to achieve a complexity of $O(n)^2$.

We designed and built a total of three versions[3] of the main algorithm, each of them better optimised than the last. This algorithm goes over every active element in the simulation and performs the corresponding actions. The algorithm needs to preform those calculations for every tick of the master clock, that is then processed and rendered in a frame and printed on the screen. We have listed the algorithm's versions, and have included a short explanation of the main concepts in which they were based.

b) 1st Version

The first version was a prototype design. We have spoken about the grid based simulation, composed by a series of cells, where entities can freely move from one to the other. In this version, there is no such grid. Every entity had a pair of variables that indicated where the individual was in the simulation. As a first implementation, it was a stable, robust and an easy to develop algorithm, that we used to test concepts and to develop the main components of the genetic algorithms.

There was a big flaw in this implementation: the collision calculation, necessary to know whether an entity was able to attack another, or feed upon a plant organism. The only information that we had about the entities were the x and y axis in the simulation plane, which meant that we had to iterate the list of entities and compare their positions to the target entity. We had to iterate the list of entities once per every entity in the simulation, for every tick of the master clock. For example, if in the simulation, we had 4 plant organisms and 3 animal organisms, we would have to figure out if the any of those 7

[2] Algorithm complexity does not exceed n (does not exceed linear complexity).
[3] All versions can be found in the project repository.

entities were colliding with another organism, comparing every organism with the remaining 6 entities. This would result in 49 comparisons for this example. In small simulations, there would be no problem, but in a medium sized simulation of 300 individuals, it would result in 90.000 comparisons per frame, reaffirming the importance of the algorithm complexity that we talked about in the past section and making this approach unfit for our purposes. The complexity of the first version of the algorithm is $\Theta(n^2)$[4].

c) 2nd Version

In the second version, we redesigned the algorithms and defined new methods to adapt the simulator to the new data structure: the grid. Thanks to this implementation, we were able to calculate collisions simply by checking the nearby cells of an entity, instead of iterating over the whole list. In the same example that we presented in the same version, every entity would have to check its contiguous cells. The number of checked cells depends on the predefined detection range, defined before running the simulation. Even if the range is high, and every individual has to check 100 cells in every clock tick, this approach would scale better in the long run, even if it is more computationally costly than the last approach in small simulations. The algorithm complexity of the second version is $\Theta(n)$[5].

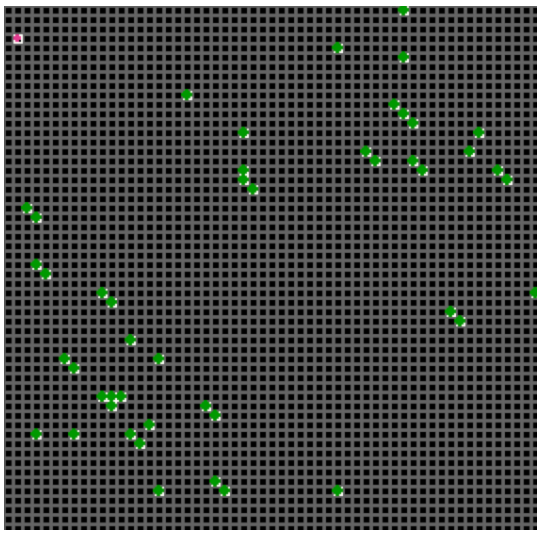


Figure 13: Grid data structure graphical representation

[4] Algorithm complexity is equal to n^2 (equal to exponential complexity).
[5] Algorithm complexity is equal to n (equal to linear complexity).

The downside of this approach is that it is harder to store and manage the positons of the individuals. A reference of the entity that is occupying the cell is needed in the cell's data structure and a reference of the cell that the entity is occupying also needs to be stored in the entity's data structure. This is a small sacrifice that we have decided to make, according to the optimisation principles seen in the last section. We have prioritised computation complexity over memory usage. When running this version of the simulator with the maximum amount of entities supported by the hardware, the CPU usage was close to 100% while the memory usage rarely reached 1 GB. From this test, we can prove that the optimisation principles are fit to the project and that, in our case, the bottleneck is happening in the CPU.

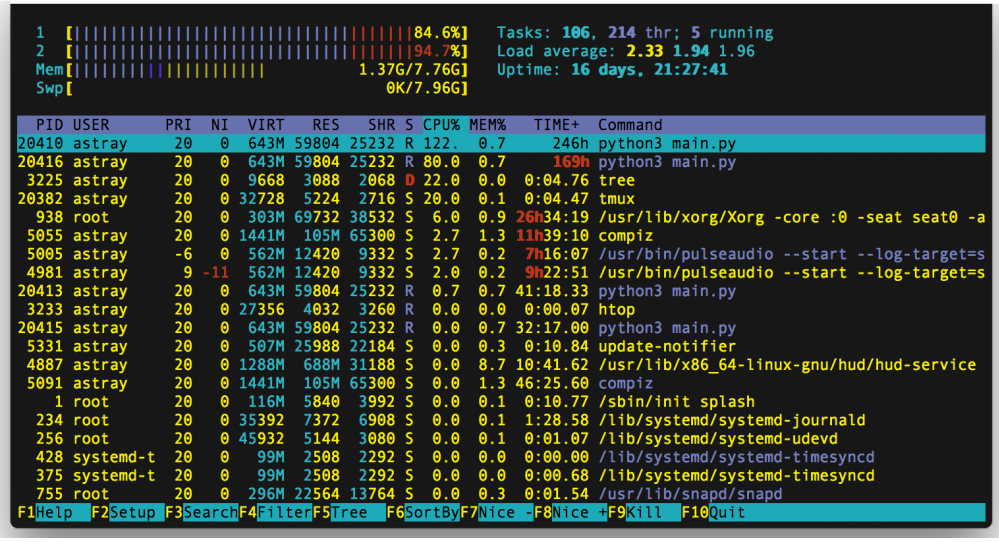


Figure 14: Htop program running on the server's terminal

d) 3rd Version

This version is closely related to the 2nd version, where we kept applying the first principle of optimisation. In the last version, the algorithm had to iterate a list of entities that contained plants organisms and animal organisms. Some of the methods had to sort them out and identify whether they were animal or plants organisms, to apply a specific method. In this version, we decided to create two extra lists of entities, one to store the animal organisms and another to store the plants organisms. These two new lists didn't

[6] Grid data structure. It is normally invisible, but can be shown as a debugging tool.

substitute the previous entities list, they are just two lists of references to the elements of that list, to optimise some of the methods and processes. We also solved minor bugs and updated some of the secondary algorithms to improve the general performance. We did some structural changes as well, which made the code clearer and easier to read.

In the next section, we will speak about the technologies that we used to design, build, train and evaluate the platform.

3.5 Technologies

a) Software

We have built the platform purely in Python. We chose it because it's a high-level programming language, that offers many tools and libraries like Numpy, very helpful in the development of artificial intelligence. Python's syntax is simple, which made the development agile and easy. We also used the library Pygame to build the graphic interface. The main drawback was performance, where python is generally not as fast as C/C++ or Java, but we still considered using it worth it. As a version control, we have used GitHub, that has been crucial in the development of the different releases of the platform. This is the link to the platform's repository:
https://github.com/astraey/AI_Ecosystems_Evolution.

b) Hardware

In the next chapter, we will delve into the details of the simulations. From now, we will say that they can take from a day to a week, depending on the mutation rate, and even longer if the value is set low. We used a MacBook Pro 15″ 2016 with a quad core Intel i7 processor, running at 2.6 GHz, with 6MB shared cache and 16 GB of RAM LPDDR3 running at 2133 MHz to develop the platform. And while that was plenty of computing power to build the platform, it wouldn't be practical to also run the simulations on it, as it is my main computing device. That is why we decided to acquire a server to run the simulation for as long as we needed to. Luckily, I had some spare computer components, and I was able to put together a decent system to use as a server. The main components were an Intel Pentium CPU overclocked up to 3GHz, a graphics card AMD ATI Radeon HD 4670 and 8 GB of RAM, running at 1333MHz. We installed Ubuntu 16.04, and all the necessary libraries to be able to run the project, as well as SSH capabilities, to access is remotely.

Figure 15: Server 1 exterior shot

c) Network Design

As the server was going to be permanently installed in my house, and I needed to be able to access it from anywhere, and not just from my home´s local network, we also acquired a free domain that we directed to the public IP of my domestic router.

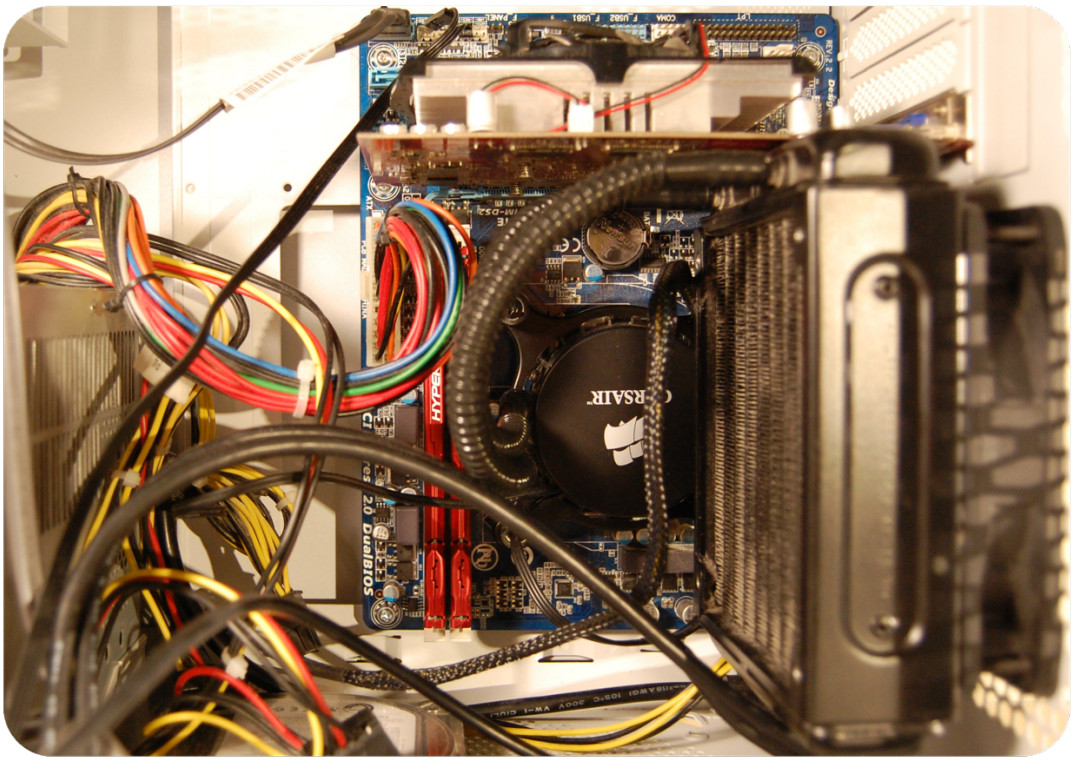


Figure 16: Server 1 interior shot

As you may know, domestic router's public IPs change quite often, which is why we had to point the domain to the router dynamically, so when it changed, the IP where the domain was pointed changed as well. We also had to redirect some ports to point at the server internally, so when the router received a SSH request through a predefined port, it redirected it to our server in my domestic network.

d) Overclocking and Cooling

As you might have seen on the picture of the internal components, the server is water cooled by a simple all-in-one solution, composed by a water pump and a radiator. Initially, it used a stock Intel CPU fan-based cooler. The reason to replace the stock CPU cooling solution by the water cooling system has a lot to do with what we spoke about in the last section about optimisation. As we mentioned, the idea was to simulate an ecosystem with as many individuals as possible, for potentially long periods of time. We also identified that the main bottleneck was clearly the CPU, even using my Apple laptop with a significantly more powerful CPU than the server where we intended to run the

simulation. It was out of our reach to acquire a more powerful CPU for the server, and if the simulations were going to be running for weeks, using my personal laptop was not a viable option. That is why we decided to overclock the server's CPU.

The process of overclocking is pretty simple. The methodology consists on setting the CPU´s frequency to a higher value, and increase the electric voltage that feeds the processor. This combination of two values is highly unstable, and the computer would crash if they are not set to a compatible combination. We could have tried different values until we got a stable one, but it's a long and slow process, as you not only have to find a stable frequency and voltage value, but you also have to put the computer under stress tests that certify that the overclocking was indeed successful and the computer is stable. Because of this, and the risk of damaging the components, always present when overclocking, we simply searched on the internet for a combination of voltage and frequency that were certified to be stable, and we overclocked the system.

The main advantage of overclocking the system is that, as the frequency of the CPU is higher, it is able to perform more operations in the same amount of time, making the computer effectively faster. But not all are advantages. In order to overclock, we need to set the value of the voltage higher, which translates into a higher CPU temperature. When the temperature of the CPU reaches a certain limit, the processor starts throttling and its performance significantly decreases. The technical word for this phenomenon is thermal throttling. And here is where our CPU cooling solution comes in. The stock fan-based cooling solution that Intel provides is enough to run the processor at the stock frequency and rarely reach thermal throttling. Because we overclocked the system, as the CPU load increased, the processor started to throttle and the performance decreased, performing even worse than it did when running at the stock frequency. That is why, if we wanted to be able to run bigger simulations on the server, we had to replace the stock CPU cooler by a better solution. There are plenty of options in the market that would have done the job just right, but we happened to have a spare Corsair Water Cooling solution, that is able to keep the processor under 60º Celsius even under full load.

At some point in the middle of the development of the project, we realised that it would be optimal to dispose of a second server, in order to simulate two ecosystems in parallel. Again, we put together a second server, using spare components even older than the first

one, an AMD Athlon CPU from the year 2000. We also mapped a new port in the router and redirected the SSH requests to the new server's static IP. This way, we were able to access both servers remotely, and run two simulations at the same time.

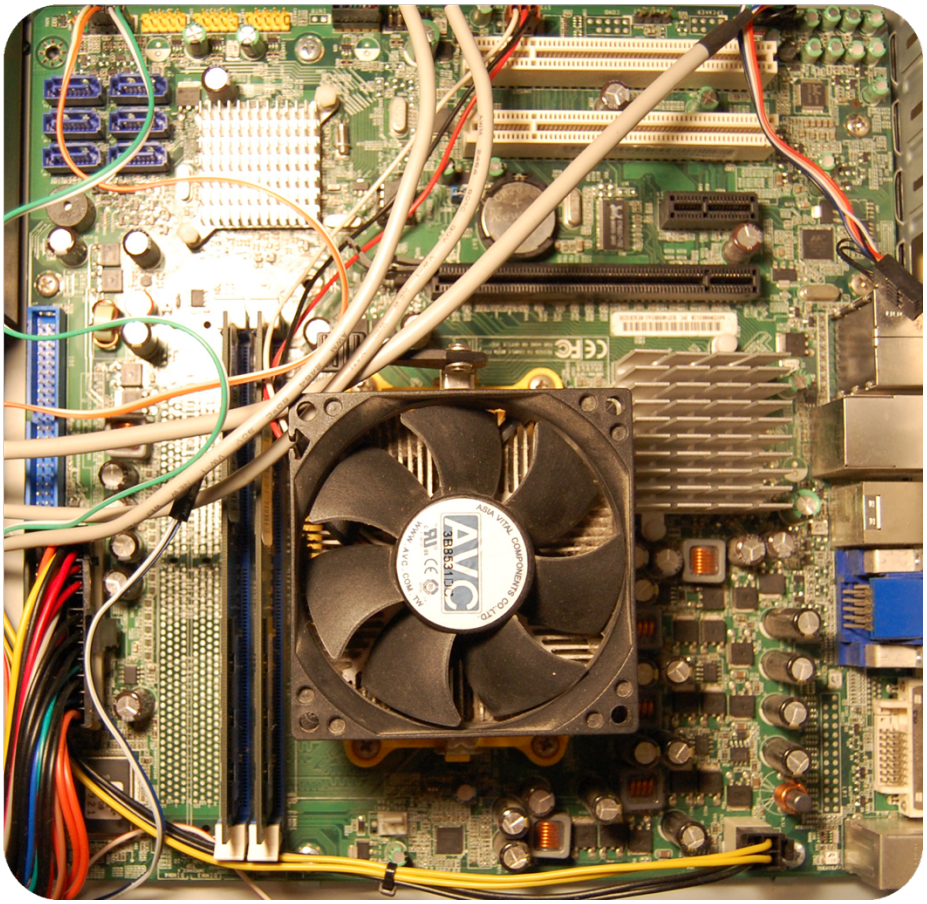


Figure 17: Server 2 interior shot

As this second system was less powerful than the first one, we used it to simulate smaller ecosystems, that had run for months, to observe unexpected evolutionary mutations, once the species had clearly been established and it seemed like they reached the optimal solutions for each of their different approaches. The first server was used to perform shorter simulations, that had varied from an hour to a week, as we implemented new features in the platform and we solved minor bugs.

4. Training and Simulations

4.1 Context and Expectations

In this section, we will delve into the simulation dynamics and the way in which the different algorithms are trained. We will also go through the technical details of the simulations and the biological relation of the results.

Prior to the design and implementation of the platform, our expectations were to be able to simulate the evolution of different species, as well as represent different types of environments. We wanted to both learn about how these types of algorithms are able to adapt and learn from themselves, and whether it is optimal or not to use genetic algorithms in other types of applications. We also wanted to learn about biology and genetics, and the impact that scientific and medical advancements have on algorithm implementation, as we talked about in the neural networks section. The goal was to build an interdisciplinary tool that combined artificial intelligence algorithms with biological concepts in order to simulate biological evolution.

We expected to be able to observe a biological-like natural selection process, where different species adopt specific strategies in order to survive. As the first generation is randomly generated, we also expected that the set of better fitted individuals by chance would endure and be able to reproduce enough to be the origin of a new species, based on their genome.

In the second section of this chapter, we will speak about the technical details of the algorithm training and the repercussions that this has in the simulation. In the third section, we will present some biological and evolutionary theories and statements, and we will run specific simulations to emulate the theory conditions, and compare their conclusions with our results.

4.2 Training and Outcomes

a) Introduction

We have really focused on the optimisation of the platforms, in order to be able to perform simulations as big as possible. Ideally, if we chose not to run the graphic interface, the resource consumption would be reduced significantly, allowing bigger simulations. And while this is useful to calculate the resultant genomes at a certain point, we miss information, like seeing the strategies that certain species adopt at certain situations, that we can easily observe with the graphical interface of the simulation. That is the reason why we chose to run simulations of about 200 individuals on average (including plants and animal organisms), going as high as 500 and as low as 50, depending on the ecosystem evolution.

b) Example Simulation

We have designed a simple simulation which is representative of the general processes that individuals go through, in order to illustrate the different concepts of the platform. We ran the simulation in a grid, 195 cells high and 195 cells wide. At the start, we included 500 plant organisms and 30 animal organisms, whose genome was randomly generated. Most of them died of starvation or were consumed by other animal organisms, as they were not naturally inclined towards feeding upon other organisms or they didn't escape or fight when they were attacked. As we included plenty of plant material in the environment, the few individuals with a strong tendency towards hunting other organisms or consuming plant material (typically 5% - 10% of

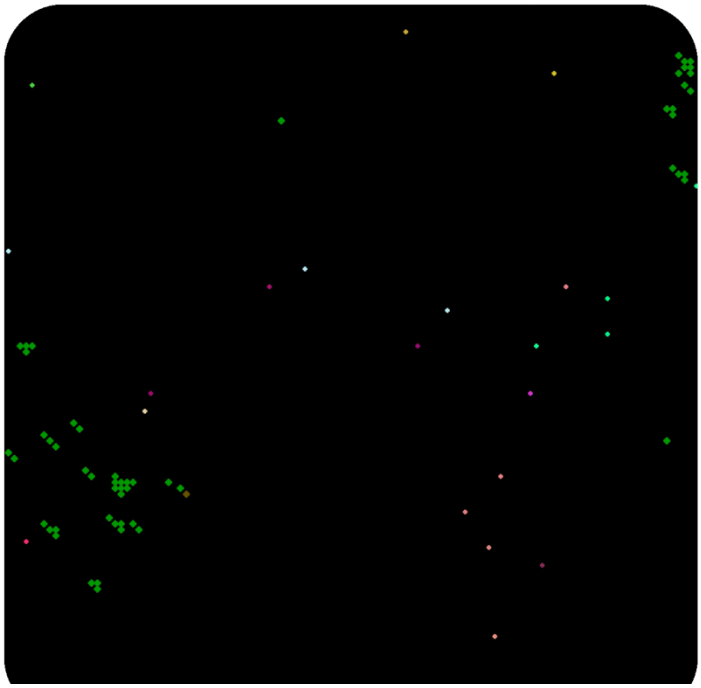


Figure 18: Example simulation

the randomly generated animal organisms) had a very good chance of reproducing in this first stage of the simulation. Out of the 30 initial individuals, around 2 to 4 organisms are usually able to survive and reproduce, resulting in the origin of a new species based on their genome. We can easily differentiate every species by the colour of the organism. As it is related to the genome, an individual's offspring will inherit the parent's colour, with a potential small variation.

These few species fought each other for resources and space, refining their genome as the simulation advanced, giving birth to new species and adapting new strategies in order to survive. To illustrate this evolution of the genomes, we have included a collection of graphs that represent the different genome variables for a sample of generations and species. It is worth mentioning that the species are just a selection of the most representative ones. We considered it better to measure evolution based on generations rather than based on time. If we measure evolution by generation, we can simplify it to the mutation rate (1%) and mutation chance (30%) indices, and we don't have to take into account factors like the average lifespan of the species.

The graph below shows the digestive adaptations, where a value of 20 represents a fully herbivore organism and a value of -20 a fully carnivore organism. Middle values correspond to omnivore organisms, with carnivore or herbivore tendencies, depending on how close the values are to 20 or to -20.

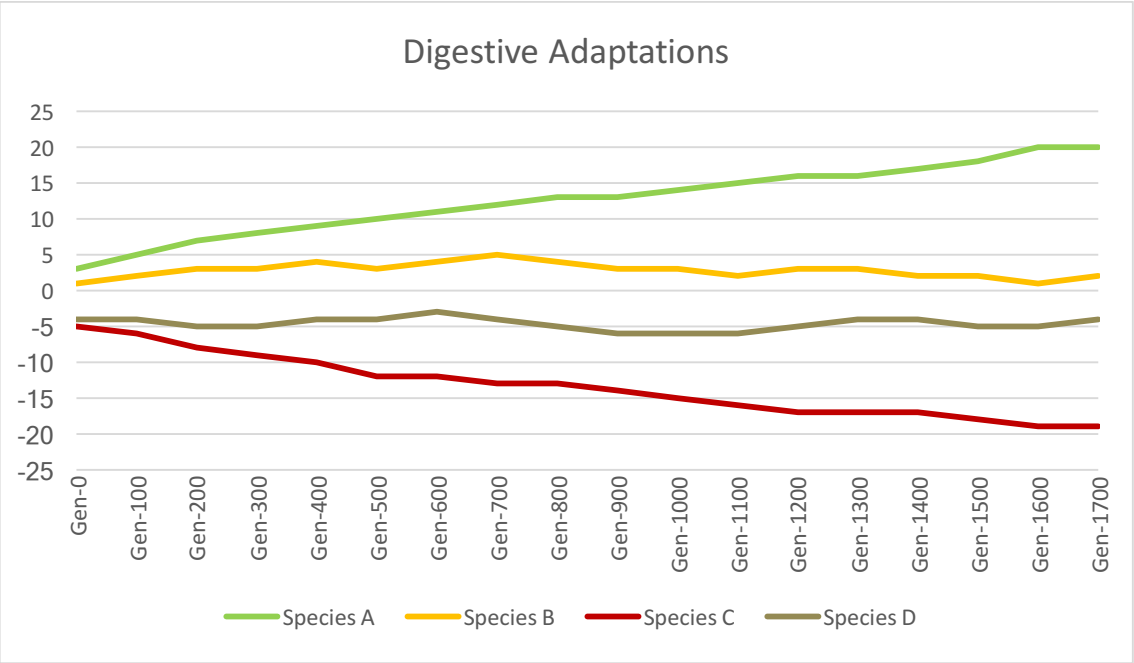


Chart 3: Digestive Adaptations of various species

The next graph shows the organisms' tendency to avoid or approach other animal organisms. As in the last graph, a value of 100 represents an organism that strongly avoids animal organisms, and a value of -100 represents an organism that approaches other animal organisms when they are in its detection range. As before, middle values correspond to organisms that combine both behaviours.

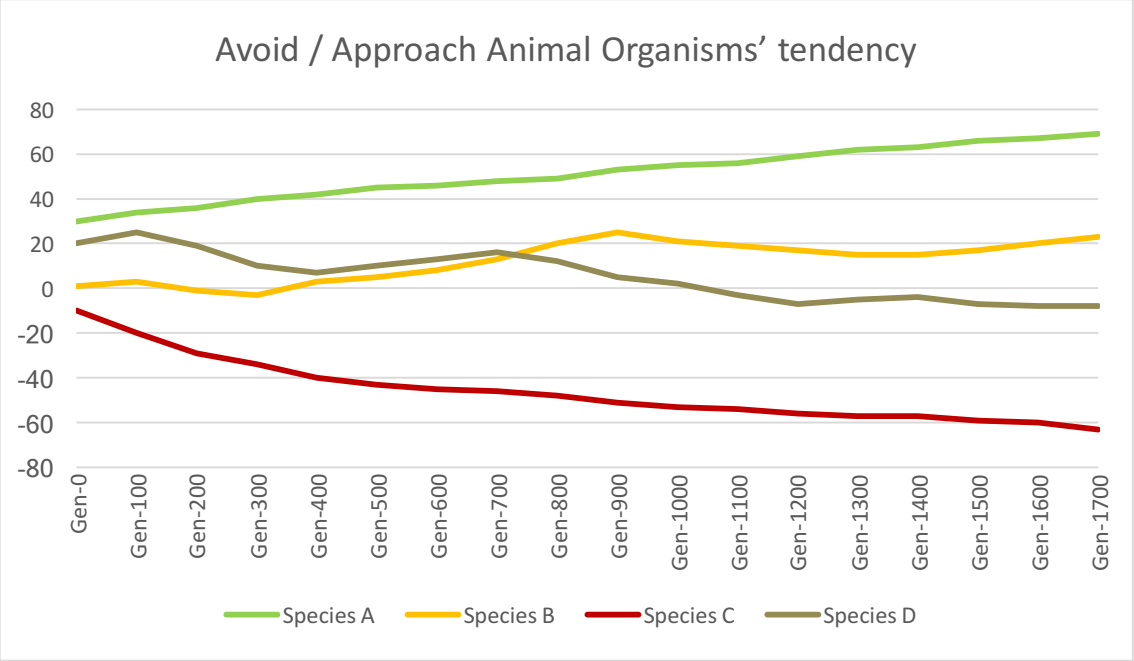


Chart 4: Avoid / Approach Animal Organisms' tendency

In the next graph, we can see the number of individuals in each generation, distinguishing plant and animal organisms. As we can see, with our current design, a population of plant organisms of a certain size can support a population of animal organisms a third of the size.

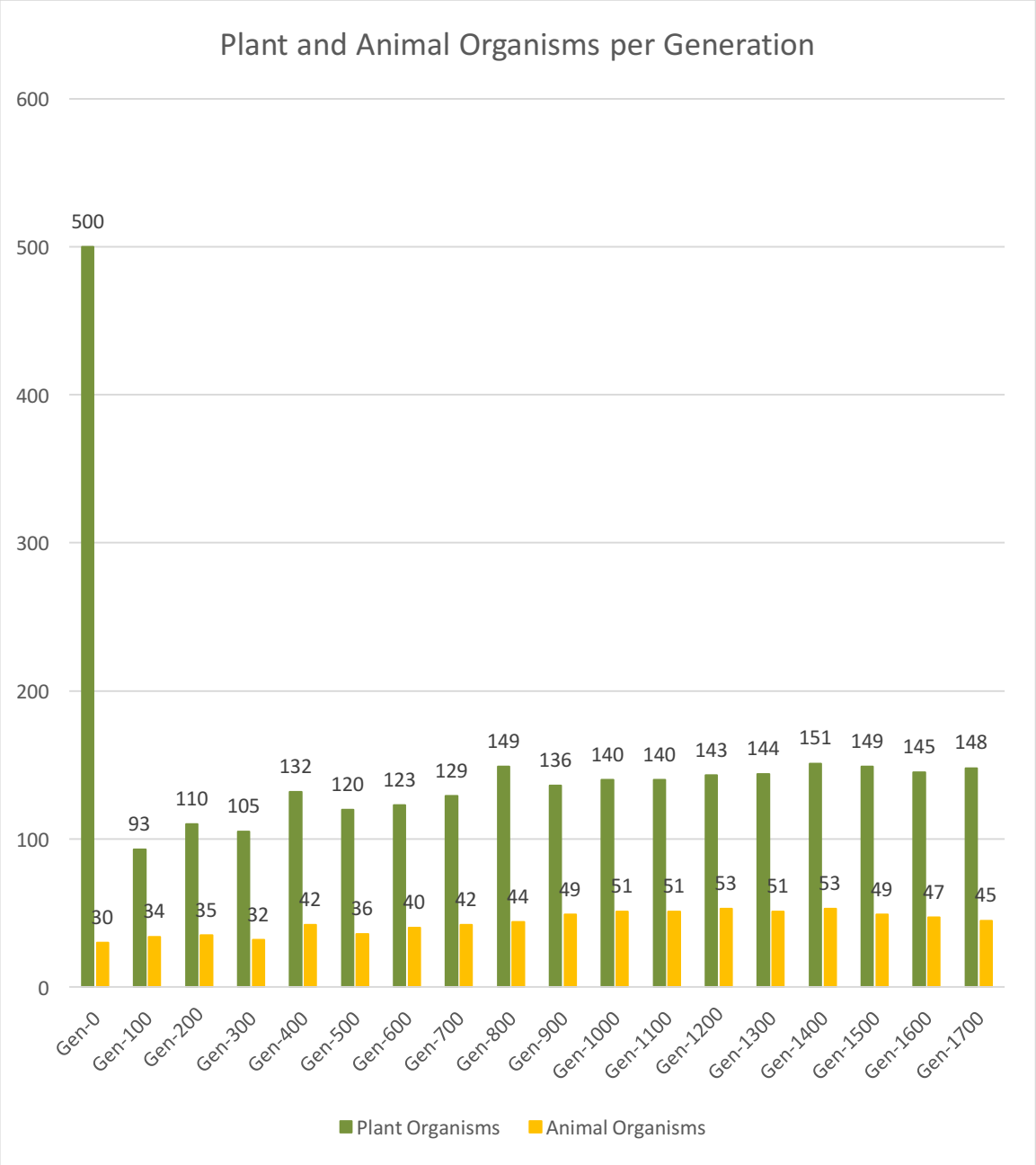


Chart 5: Plant and Animal Organisms per Generation

The following graph represents the number of individuals per generation, classified depending on their alimentary preferences. In order to identify each individual's group,

we measured their digestive adaptation value (the classification by species can be seen in the first graph), and considered it herbivore if it was in the upper part of the graph, carnivore if it was in the bottom part and omnivore if it was closer to the middle.

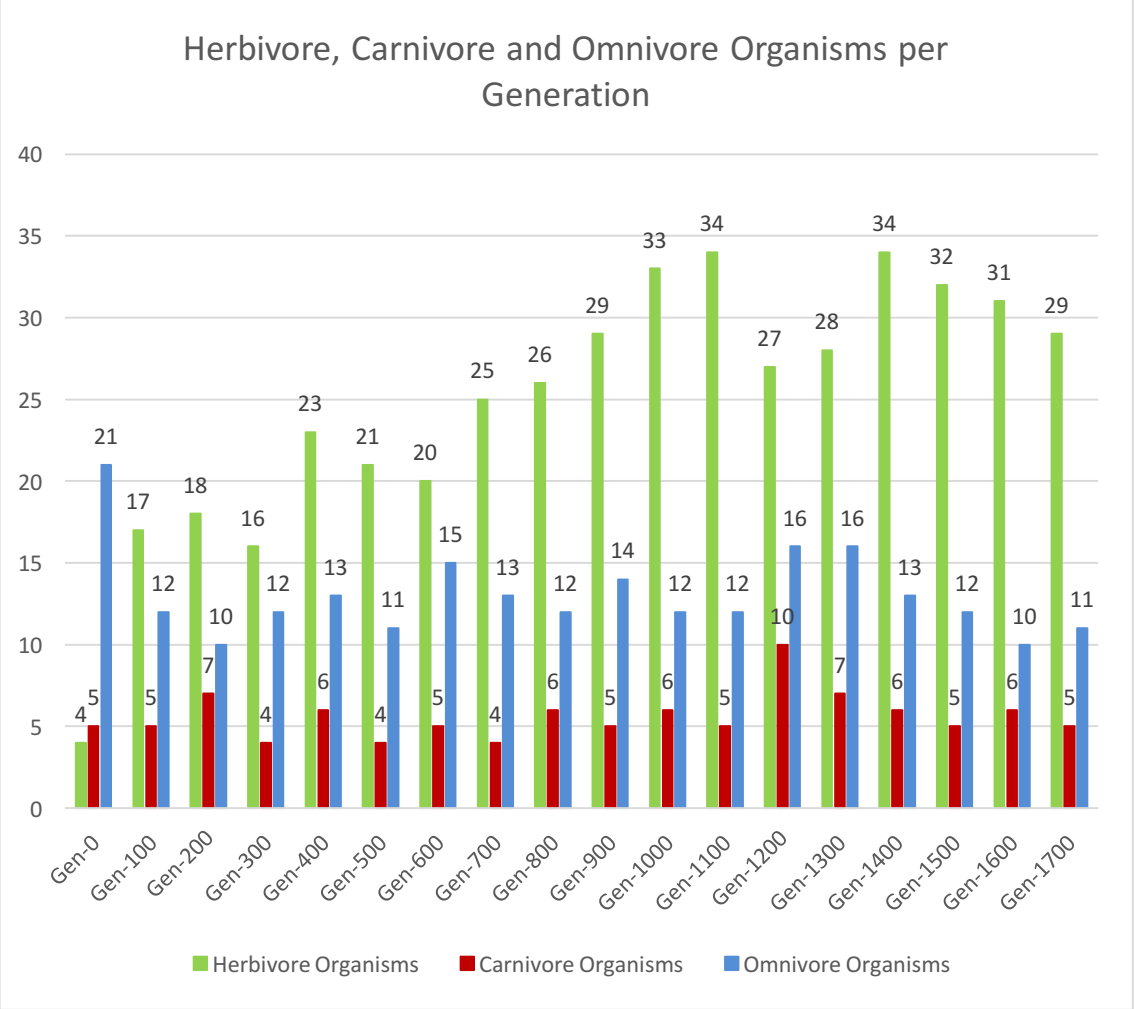


Chart 6: Herbivore, Carnivore and Omnivore Organisms per Generation

It is worth mentioning that these graphs are very simplified, as we only show the general tendency of some species that made it to the end of the simulation. If we examine the evolution process more closely, and we focus on the individuals, we can see that at any point, random mutations occur to fill the gaps left by other species. For example, if the population of the main herbivore species decreases, the mutations which potentially will be more successful in the next period are the ones better adapted to digest plant material. The evolution process that an individual goes through to fill those gaps heavily depends on how far it was from the target solution. Coming back to the example, if the population

of the herbivore species decreased dramatically, the individuals that most likely would fill those gaps would be the omnivore's offspring that mutated towards consuming plant material. The individuals less likely would be the carnivore's offspring with this same mutation, as it would be too far to begin with, and it would not be worth it.

The next graph is very illustrative, as it is a clear indication of the evolution of the ecosystem as a whole. It is a representation of the type of death that the individuals in a certain generation have suffered, where death of starvation means that the individual had no access to biomaterial, and the living cost caused that it ran out of biomaterial itself. The death by consumption represents that the individual was consumed by another organism, most likely an omnivore or a carnivore. In the graph, we can see how, as the simulation advances, the deaths by starvation are reduced, mainly because individuals learn and perfect how to feed. We have represented the total number of deaths of each type for every one of the selected generations.

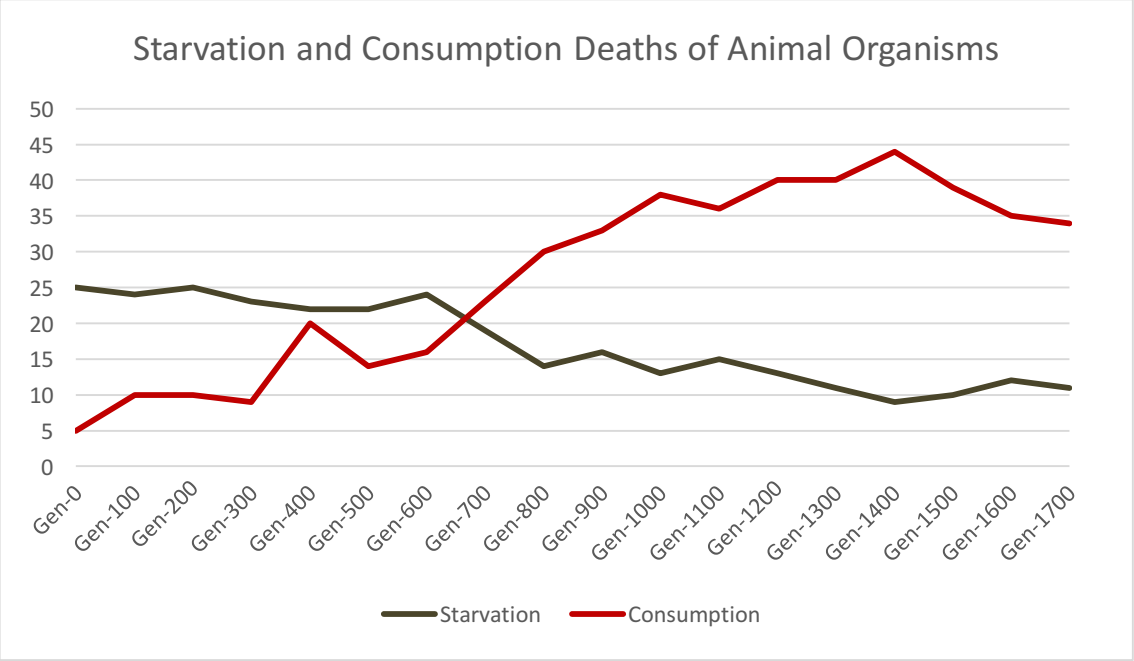


Chart 7: Starvation and Consumption Deaths of Animal Organisms

As we have seen in the graphs above, around the generation 1700 the different attributes stabilise, and from a genetic algorithm standpoint, we reach the solutions for the environment (the genomes of every specimen of a species). As we mentioned, these solutions are strictly dependent on each other, just like in nature. If carnivores are left

with no herbivores, they will die of starvation before long. If herbivores are left without carnivores, the population will grow too much, they will consume all the plants, and they will end up dying of starvation, or greatly reducing their numbers. That is the reason why we don't consider this a solution of the problem, but rather a collection of genomes that conforms a solution: a stable ecosystem with a set of fully adapted and co-dependent species.

4.3 Theory-based Simulations

a) Introduction

In this section, we will present a collection of evolutionary theories and we will design specific simulations that will be representative of the environment and conditions presented, in order to verify the results.

b) Limited Lifespan and Non-Limited Lifespan

There are many theories where the trade-offs of unlimited and limited lifespans are discussed. Yaneer Bar-Yam addresses in many of his papers that lifespans are genetically controlled, according to the resource limitation of a given environment. The goal is to leave sufficient resources for their offspring, and to accelerate the adaptation process. Shorter lifespans mean more generations in the same amount of time, and more potential genome mutations.

We designed a new simulation where every individual had a limited lifespan, and compared the results with the simulation presented in the last section, and the results confirmed Yaneer's theory. Indeed, when limiting the lifespan, the adaptation process accelerates. For instance, he says that the longest possible lifespan tends not to be naturally selected. It needs to be long enough for the individual to be able to reproduce, but not too long so the individual consumes the offspring's resources.

The following graph represents the adaptation acceleration when limiting the lifespan, where a higher value means a higher adaptation level to the environment.

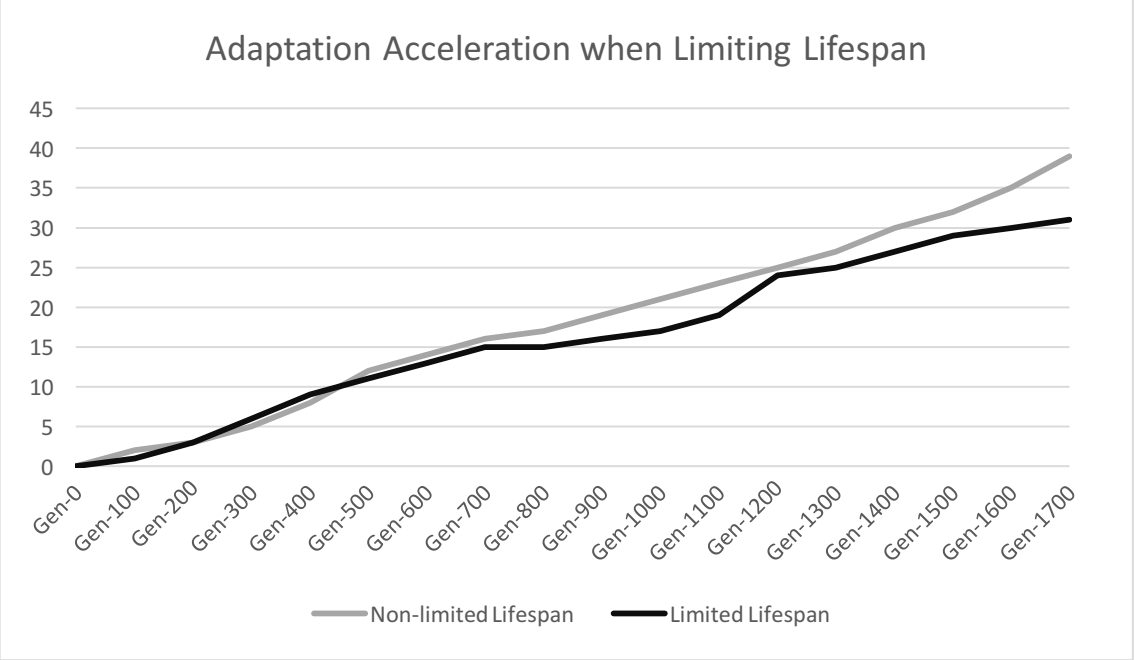


Chart 8: Adaptation Acceleration when Limiting Lifespan

Even though the data for this graph has been simplified, as it is rather hard to quantify the adaptation of a species to an environment, especially in omnivore species, we can still see a clear adaptation advantage when we limit the lifespan.

c) Lifespan's Impact on Biological Adaptation

The next theory is closely related to the last one. It states that there is a relation between lifespan and adaptation rates, meaning that the lifespan needs to be long enough for the individual to reproduce but not too long so it consumes its offspring's resources.

We designed a collection of simulations where we defined a certain lifespan in each of them. The results can be seen in the following graph, where a higher value means a higher adaptation level to the environment:

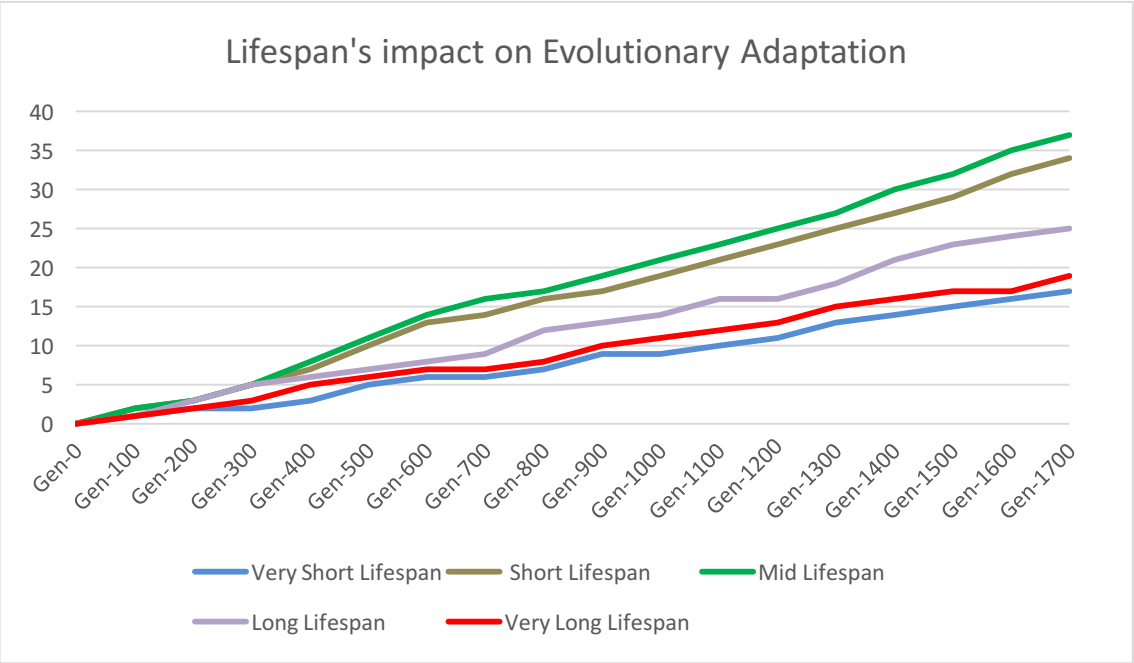


Chart 9: Lifespan's impact on Evolutionary Adaptation

We can clearly see that the best approaches are mid and short lifespans, where the better adapted individuals have time to reproduce, and leave enough resources for their offspring. It is interesting to see that the worst performer is the shortest lifespan, closely followed by the longest lifespan, even though the order in which the two perform is heavily dependent on the type of simulation and the way reproduction has been implemented. As stated in the theory, an approach not too long nor too short is a good solution.

d) High-resource and Low-resource Environments

Another topic of discussion among biologists is the potential adaptation of species fit to survive in environments with low resources when taken to a high resource environment, as opposed to the opposite process. The general opinion is that the species adapted to survive with very little resources when its environment is replaced will have it easier than the species used to live in a high resource environment. However, there are many people that disagree with this, and believe that it will mean an equal struggle for life.

For this theory, we ran a total of 4 simulations. We used the first two to generate both collections of solutions that represent the different species developed in each of the two

environments. We emulated this resource density by controlling the rate in which plants reproduce. This is an easy way of simplifying the resource availability, as it has an impact in each level of the ecosystem pyramid. The parallelism with nature would be that typically, water and soil composition are the two main factors that potentiate biodiversity and resource richness in an environment. This is due to the fact that they are the main energy source for plants, the base level of the natural pyramid.

The next step was to collect those results and run two more simulations, where we exchanged the environments, meaning that neither of the set of solutions were in the environment that they evolved to survive in. In the graph below, we show a comparison between the adaptation potential in both cases. We decided to measure the struggle based on the deaths by starvation per generation, for both collections of solutions, the ones that had gone from a low resource environment to a high resource environment and the ones that had gone from a high resource environment to a low resource environment. It is worth mentioning that, the total number of individuals capable of surviving in the same grid in the high resource environment is way higher, so comparing the total amount of deaths made no sense. That is why we have represented the data in percentages, meaning that a value of 30 for a generation would mean that 30% of that generation has died of starvation.

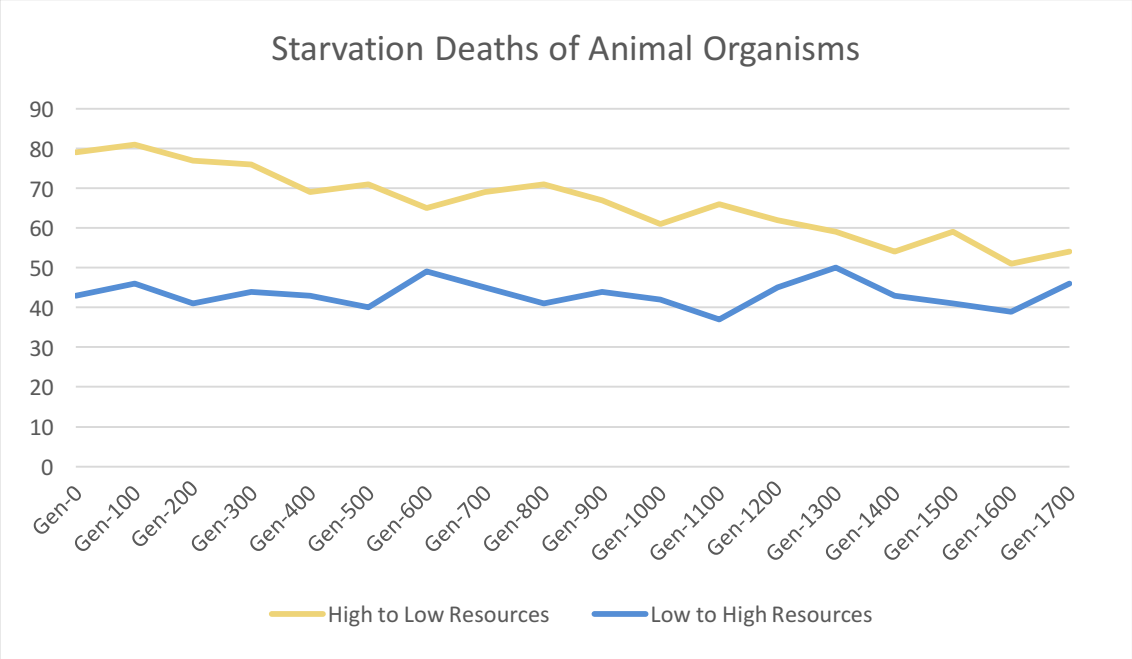


Chart 10: Starvation Deaths of Animal Organisms

As expected, the species moved from a high resource environment to a low resource environment struggles the most, although we can see a clear evolution and adaptation as the simulation advances. On the other hand, the species moved from a low resource environment to a high resource environment remain the same, with no apparent tendency. These results confirm the initial assumption.

5. Conclusions

A year ago, we stated that the main goal of the project was to build an ecosystem simulation platform through genetic algorithms and neural networks, and today, I can confirm that we have achieved that goal. We have built a system, although biologically simplified, that includes the core evolutionary principles, where species evolve and adapt to the environment. We are proud of the results, but we can't help but feel like there is still so much more that could be done to improve and expand the platform, so many ways of extending its functionality and getting closer to biology with every step.

Our plan is to keep working on the project, and the following are some of the ideas that we would like to implement in the future. The next step will be to develop a more complete genome, that also controls the shape of the individuals. We also want this shape to have an impact on the way the individuals interact with the environment and with other organisms.

We also want to model more biological concepts and processes, like other reproduction methods, more ways in which individuals can interact with each other, or the modelling of a method so the colour of the individuals has an impact on their visibility and interaction, depending on the environment that they are in.

The graphical interface needs improvements. At the moment, it is quite rudimentary and does not add a lot of functionality to the simulator.

We will also keep exploring the broad field of artificial intelligence, potentially including new algorithms or optimising the current ones.

Additionally, we have spoken about taking this project as a base, and building an ecosystem simulator in 3D. This would certainly open a new dimension for biodiversity, combined with the implementation of a more complex genome that contemplated organism shapes.

I can proudly say that I have learned a lot not only about artificial intelligence but also about biology, history, algorithm optimisation and most importantly, I have enjoyed building the platform. I have found the process of creating something from scratch fascinating, turning an idea into something real, capable of simulating living organisms and their evolution.

Bibliography

[1] Darwin, C. (1859) *The Origin of Species by means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*. Available at: http://darwin-online.org.uk/converted/pdf/1861_OriginNY_F382.pdf (Accessed: 2 September 2016).

[2] *Global Optimisation and Algorithms, Theory and Application*. Available at: http://www.it-weise.de/projects/book.pdf (Accessed: 25 October 2016).

[3] *Physical Review Letters*, Stanford University and Center for the Study of Language and Information (U.S.) (1997) *Stanford encyclopedia of philosophy*. Stanford University. Available at: https://plato.stanford.edu/entries/equivME/ (Accessed: 16 November 2016).

[4] *Introduction to Genetic Algorithms* (1996). Available at: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html (Accessed: 28 November 2016).

[5] *Artificial Intelligence Overview*. Available at: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_overview.htm (Accessed: 3 December 2016).

[6] *What is DNA? - Genetics Home Reference*. Available at: https://ghr.nlm.nih.gov/primer/basics/dna (Accessed: 8 December 2017).

[7] Griffiths, A. J., Gelbart, W. M., Miller, J. H. and Lewontin, R. C. (1999) 'Darwin's Revolution'. W. H. Freeman. Available at: https://www.ncbi.nlm.nih.gov/books/NBK21378/ (Accessed: 21 February 2017).

[8] Randy L. Haupt and Sue Ellen Haupt (no date) *Practical Genetic Algorithms.* Available at:
https://archive.org/details/PracticalGeneticAlgorithms
(Accessed: 20 February 2017).

[9] Algorithms, E. (1957) 'Genetic and Evolutionary Algorithms', *Encyclopedia of Computational Chemistry. John Wiley*. Available at:
http://www.wiley.com/legacy/wileychi/ecc/samples/sample10.pdf
(Accessed: 3 March 2017).

[10] *Complexity and Ecosystem Management. The Theory and Practice of Multi-Agent Systems, Edward Elgar*, (November 2015). Available at:
http://s3.amazonaws.com/academia.edu.documents/41904420/Multi-agent_systems_and_role_games_colle20160202-31574-ffoaks.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1497201998&Signature=C4yLweHwlv3v2461SrwDoi2%2FFns%3D&response-content-disposition=inline%3B filename%3DMulti-agent_systems_and_role_games_colle.pdf
(Accessed: 22 March 2017).

[11] *Genetics and Molecular Biology* (1996), Chan, Lawrence
Ehnholm, Christian. Available at:
http://gene.bio.jhu.edu/bm2whole.pdf
(Accessed: 22 March 2017).

[12] *Ecosystems, flow of energy and matter.* Available at:
http://www.wiley.com/legacy/Australia/PageProofs/SQ9_AC_VIC/c05EcosystemsFlowOfEnergyAndMatter_WEB.pdf
(Accessed: 1 April 2017).