Predator-Prey Coevolution Simulation

Mateusz Kocot Łukasz Pitrus

2022

Abstract

Predator and prey interactions are common in nature, therefore there is a lot of research carried out to simulate these interactions. Here we present a highly configurable application capable of running such simulations. Although our project was inspired by the NetLogo application, we added a few important changes, e.g. we allow many animals on a single tile. We also simulate simple genomes. We ran a lot of variously-configured simulations and managed to achieve interesting results. While some of the simulations were balanced, the others ended up with one or two species extinct. We managed to achieve the simulations where the population graphs were compatible with the Lotka-Volterra model. We also proved the Darwin's theory of evolution by analysing the genomes of animals throughout the simulations.

Keywords: Lotka-Volterra, coevolution, predator-prey, simulation

1 Introduction and review of related research

In this section we introduce and describe the predator-prey coevolution problem. We formulate our research goals and review related works.

1.1 Introduction

Predator and prey interactions are common in nature. There are numerous examples of these interactions, including wolf and sheep, lion and zebra, fox and rabbit, etc. Each animal species has its own place in the food chain. It is well known that predators affect the number of their prey by eating them, and it may not be easy to realise, but prey affect the predator population as well. Increasing the number of predators decreases the number of prey, but when there are too few prey, predators start to die of starvation, so their population decreases. This relationship can be represented as a cyclical pattern, which can go on indefinitely if only none of the populations drops to zero. If prey become extinct, predators loose their food source, so they are doomed as well. However, when predators die out, the population of prey does not go to infinity because their food source (plants) is limited.

Since the problem is prevalent, there is a lot of research being carried out in order to simulate predator and prey interactions properly. A basic example that does not include a food source is described in [1]. The predator-prey population plots are shown in Figure 1. The cyclical pattern in Figure 1, right, can be modelled using the Lotka-Volterra equations. An example of a predator-prey system simulation using the Lotka-Volterra model is described in [2].

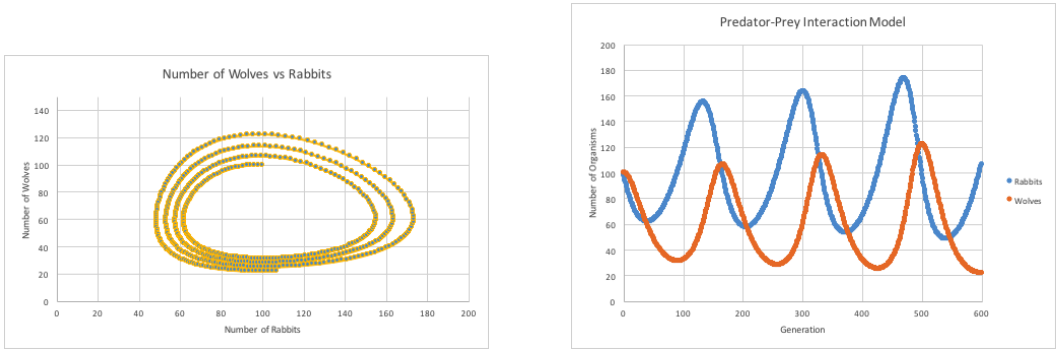

Figure 1: Example plots showing the interactions between predators and prey (source: [1]). The plot on the right side is an example of the Lotka-Volterra model of predator and prey interactions.

1.2 Research goals

Our goal was to create a configurable tool capable of running and observing a predator-prey simulation. Our work was inspired by the NetLogo application [3] (see Figure 2) which, among other simulations, includes the predator and prey interactions.

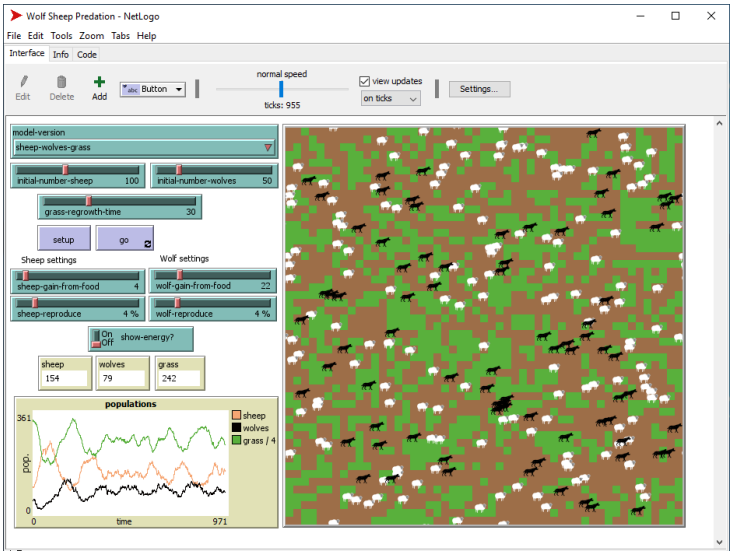

Figure 2: NetLogo [3] predator-prey simulation GUI

We planned to base on the functionalities present in the NetLogo application. Our goal was to iteratively add new features including attributes for animals (e.g. view range or energy consumption efficiency) and possibility of their evolution when reproducing.

We formulate the problem and describe our solution in the second section. Then, in the third section we test our application and observe its behaviour in various settings. Finally, we conclude the project in the fourth section. The code of our project can be found in a repository on GitHub [4].

2 Problem formulation and proposed solution

In this section we describe our simulation model and algorithms we use for the simulation. We also present the Graphical User Interface (GUI) and describe configurable parameters.

2.1 Simulation model

In principle, our model is similar to the one used in the NetLogo application. We use a square map and place two types of animals on it – prey and predators. The animals need energy in order to survive, therefore we also add plants which provide the energy for prey. Predators take their energy from eating prey. Animals move and perform interactions which reduce their energy. If it drops to zero, an animal is presumed death.

While in NetLogo each map tile can host only one animal, we enable many animals on a single tile. Only animals that are on the same map tile can interact with themselves. We store them in a list whose order determines the sequence in which the animals eat plants or interact. The list is sorted by the time an animal has entered the tile. An interaction can be either eating a prey by a predator or mating between two animals of the same species.

We also modify the simulation of plants necessary for prey to survive. In our model, density of plants on a tile is simulated by a floating point number which is increased every turn. We assume that the number of plants that a prey can eat is the density of plants rounded down to an integer. Plants on a single tile are eaten by all prey present on it. If there are not enough plants, they are eaten by individuals that came to the tile first (each eats one plant). If there are more plants than prey individuals, the first one eats more. For instance, if the density of plants on a tile is 7.3, and there are 5 prey, the first one eats 3 plants, and the others – 1 plant. After the eating phase, the density of plants on the tile is 0.3.

Finally, every animal has its own genome which contains the gene types listed below.

- View range – view range in which animal perceives its surroundings (predators, food and mates detection). After analysing its surroundings, an animal has to choose the direction of the next move.
- Energy consumption ratio – energy efficiency of animals; the higher the ratio, the more energy an animal consumes.
- Max animal energy – maximal energy an animal can accumulate.
- Fear of predators – the higher the value of this parameter, the smaller the probability of a prey going towards one of the predators nearby. This parameter does not have impact on predators.
- Eating over mating ratio – the higher the value of this parameter, the bigger the chances of an animal choosing the direction with eating feasibility instead of mating feasibility.

A child inherits mean values of their parents' genes with slight, random changes (mutations). One can configure each gene's ranges and defeault values in the GUI.

It is possible to turn off the genome simulation. In that case, view range is set to 0 (all moves are random), and other genes are not taken into account.

2.2 Simulation scheme

We start the simulation with an empty, square map. Then we place first prey and predators on it. We do not place two animals on a single tile though. Plant density on every tile is set to 0. Then the simulation begins. Each turn is divided into five phases described below.

1. Clear dead animals – animals that died in the previous turn are removed from the map.

2. Move animals – each animal makes a move, i.e. changes its position on the map to one of the neighbouring tiles.

3. Process interactions – when there are multiple animals on the single tile, they interact in pairs with each other (the first with the second, the third with the fourth, etc.).

4. Put newborns on the map – after the previous phase, which includes mating interactions, there are newborn animals, which are placed on the map. This includes inheriting parents' genomes and applying mutations.

5. Process plants (eating and growing) – plants on tiles occupied by prey are eaten and all vegetation on map grows.

The simulation is not automatically stopped, so it is running even if one or two species become extinct.

2.3 Configurable parameters

Apart from modifying default gene values and ranges, our simulator provides many configurable parameters briefly described below.

- Map size – the simulation map size
- Predator count – the predator population count at the beginning of the simulation
- Prey count – the prey population count at the beginning of the simulation
- Base animal energy – the starting energy of all animals at the beginning of the simulation
- Grass regeneration ratio – the amount of plants (usually a floating point value) added to each tile every turn
- Maximum grass units – the maximum amount of plants on a tile (a fully-grown tile)
- Minimum reproduction energy – the minimal energy value required for animals to produce an offspring (if an animal does not have enough energy, it cannot mate with others)
- Predator food efficiency ratio – a multiplication factor for the energy gained by eating a prey
- Max energy check multiplier – animals do not eat and try to find food within their view range if their energy is bigger than the value of their max energy gene multiplied by the value of this parameter.
- Child energy denominator – after mating, each parent loses 1/3 of their energy. The energy of their child equals to the sum of the parents' energies lost during coupling divided by the value of this parameter.
- Mutation ratio – Maximum gene value shift for a newborn (not present in GUI)

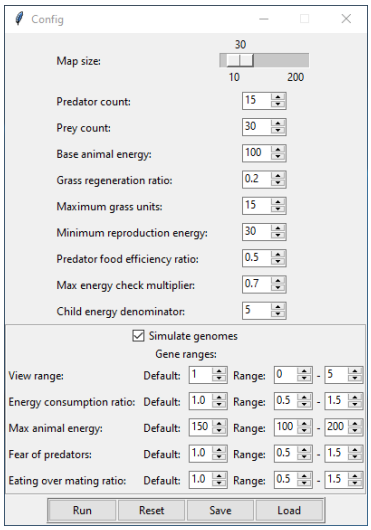

Figure 3: The configuration window. It allows setting the values of configurable parameters and ranges of gene types. The default values are present in the picture.

2.4 GUI

The simulation can be configured and observed in the GUI of our application.

We allow a high level of configuration so as to adapt the simulation to the user's needs. The simulation can be configured instantly after running the application. The configuration window is shown in Figure 3. GUI also enables saving and loading the configuration from a file.

The base simulation window (Figure 4) includes the map visualisation and the population graph presenting the histories of the species' populations and total plant count. The user can manually apply next turn or use a timer with configurable speed. The simulation can be paused at any point in time. The user can restart the simulation and set new values of the configurable parameters. In case of bigger maps, it is possible to turn off refreshing of the map which significantly increases the performance.

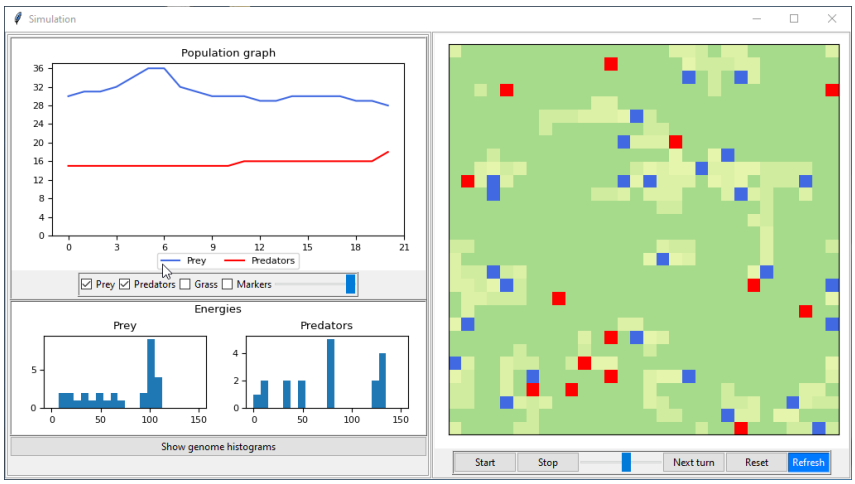

Figure 4: The base simulation window. It includes the population graphs, the energy histograms and the map.

The user can also turn on showing the gene histograms which present the current genomes on the map (Figure 5).

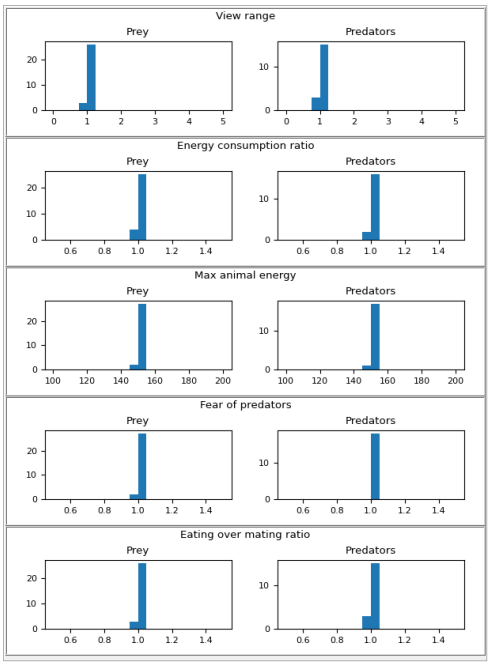

Figure 5: The gene histograms frame

2.5 Implementation specification

In our project we used the Python 3 language with some common libraries such as matplotlib, numpy and tkinter. We based our initial code on another project with basic functionality only [5], but the final code was implemented mostly from scratch.

3 Experimental research results

Below we present and describe a few experiments we have run. Some of them have small mutation ratios which means that the mutation is negligible. On the other hand, others have higher mutation ratios, so genomes evolve, and we can check if Darwin's theory of evolution works in the case of our simulation.

3.1 Basic stable scenario

At first we present a basic simulation on a small, 30×30 map, with negligible mutation ratio. The detailed configuration is presented in Figure 6. Figure 7 shows the population graphs of prey and predators. The simulation is stable and none of the species became extinct. The dependence between prey and predators is clearly visible, especially between 3500th and 5500th turn, where the graph resembles the Lotka-Volterra model.

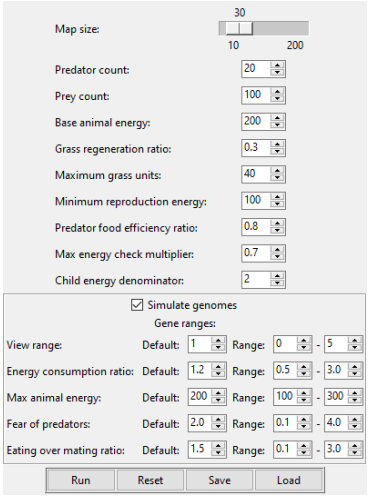

Figure 6: Simulation configuration

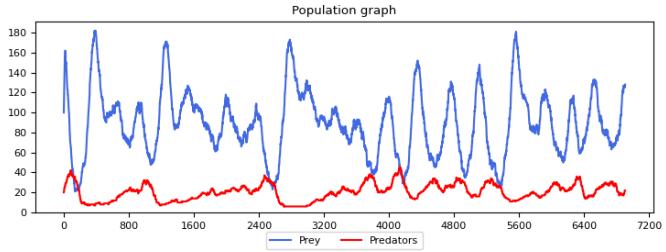

Figure 7: Population graph

3.2 Stable scenario with more mutations

In this scenario, the mutation ratio was changed from 0.05 to 0.15. Other settings were slightly modified too (look at Figure 8). Figure 9 shows a really stable simulation, with many changes in population sizes of both prey and predators. The higher mutation rate allowed animals to evolve much faster. Genes' histograms in Figure 10 show that some of them changed overtime. Fear of predators increased in prey population and eating over mating increased in predators' population.

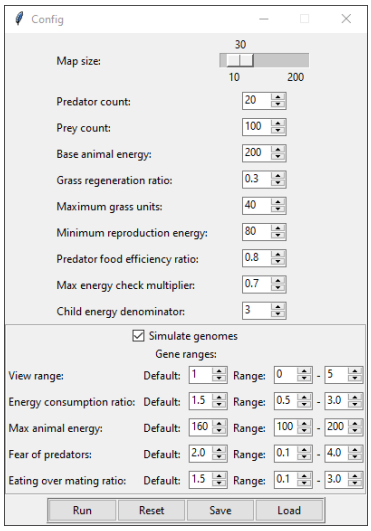

Figure 8: Simulation configuration

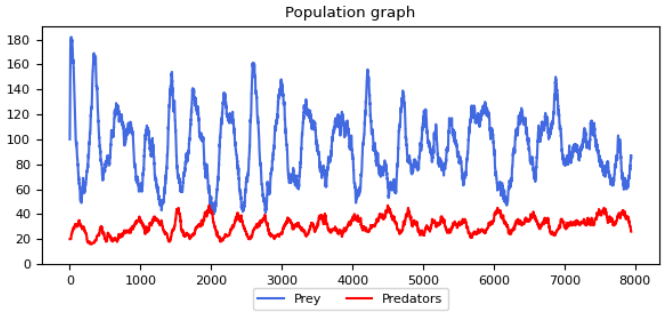

Figure 9: Population graph

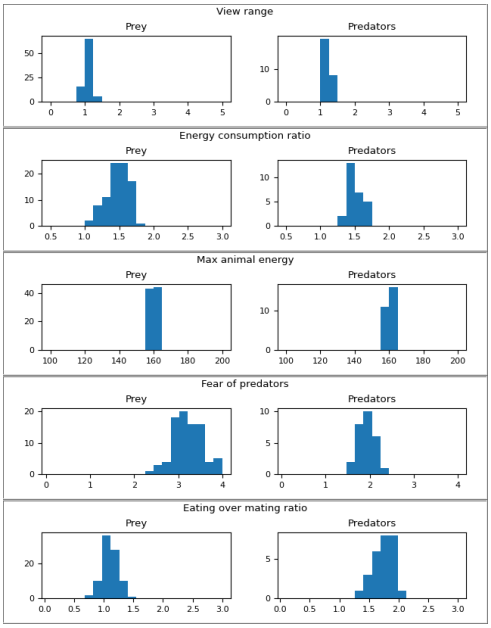

Figure 10: Population genes

3.3 Stable scenario - death of one population

During experiments we ended up with some simulations looking stable with one of the populations dying out at some point. As an example of prey's death we used configuration in Figure 11 (left) an got graph presented in Figure 12 (top). Due to predators being dependent on prey, they would also die in a matter of few turns, but we ended simulation due to knowing the future result.

Another simulation run with configuration in Figure 11 (right) resulted in death of predators, shown on Figure 12 (bottom). This simulation was also ended after death, because it would only show the increase of the prey population after which it would stabilise at some level.

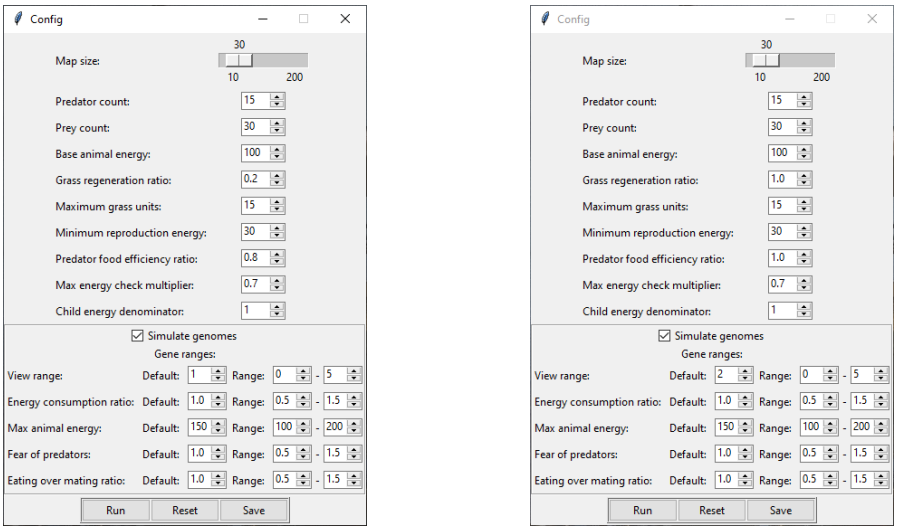

Figure 11: First (left) and second (right) simulation configurations

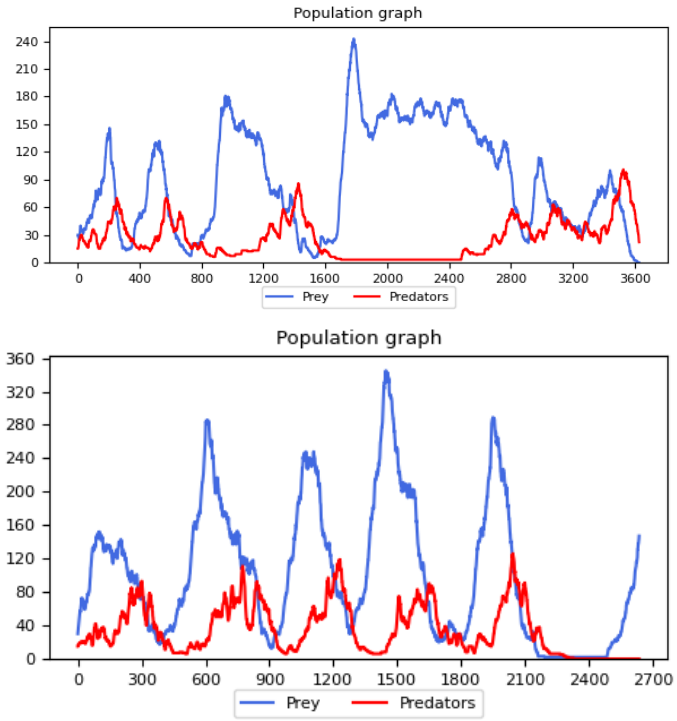

Figure 12: First (top) and second (bottom) population graphs

3.4 Bigger map, small mutation ratio

In this case, we increased the map size to 99×99. Now, animals have more space and they can form clusters. In this case, the mutation ratio is negligible. We present the configuration in Figure 13 and the population graph in Figure 14, top. Figure 14, bottom shows the plant count graph. In this configuration, we managed to achieve a clear Lotka-Volterra model with similar amplitudes of prey and predators. What is interesting, the amplitude is not constant – it raises for about 3000 first turns. Then it decreases. The shape of the plant count graph looks as expected. It is inversely proportional to the prey graph.

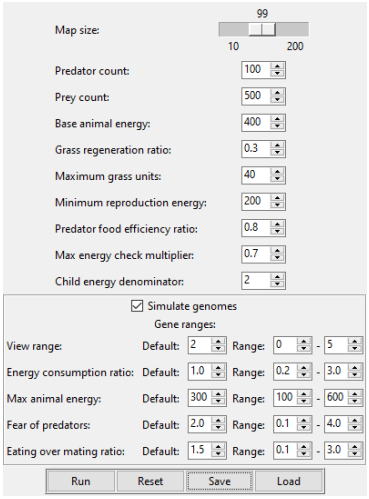

Figure 13: Simulation configuration

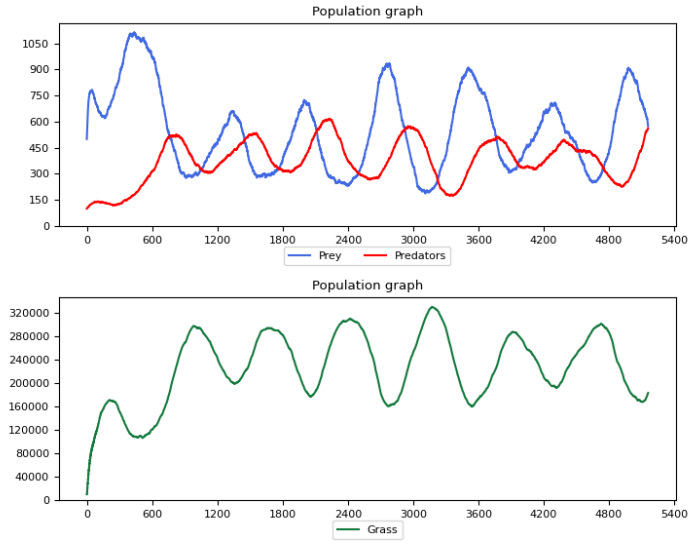

Figure 14: Population (top) and plant (bottom) graph

We can also analyse the map at different moments in the simulation. In Figure 15 we present a few more interesting moments. Below we analyse them top to bottom, left to right.

At first, after 700 turns, the simulation stabilises after the initialisation and enters the Lotka-Volterra fluctuations. Some parts of the map are inaccessible for prey. Then, the prey count drops and starts to return in the second map, after 1100 turns. The jungle is overgrown. The third map presents the state after 1350 turns in which the prey count is in its peak while the

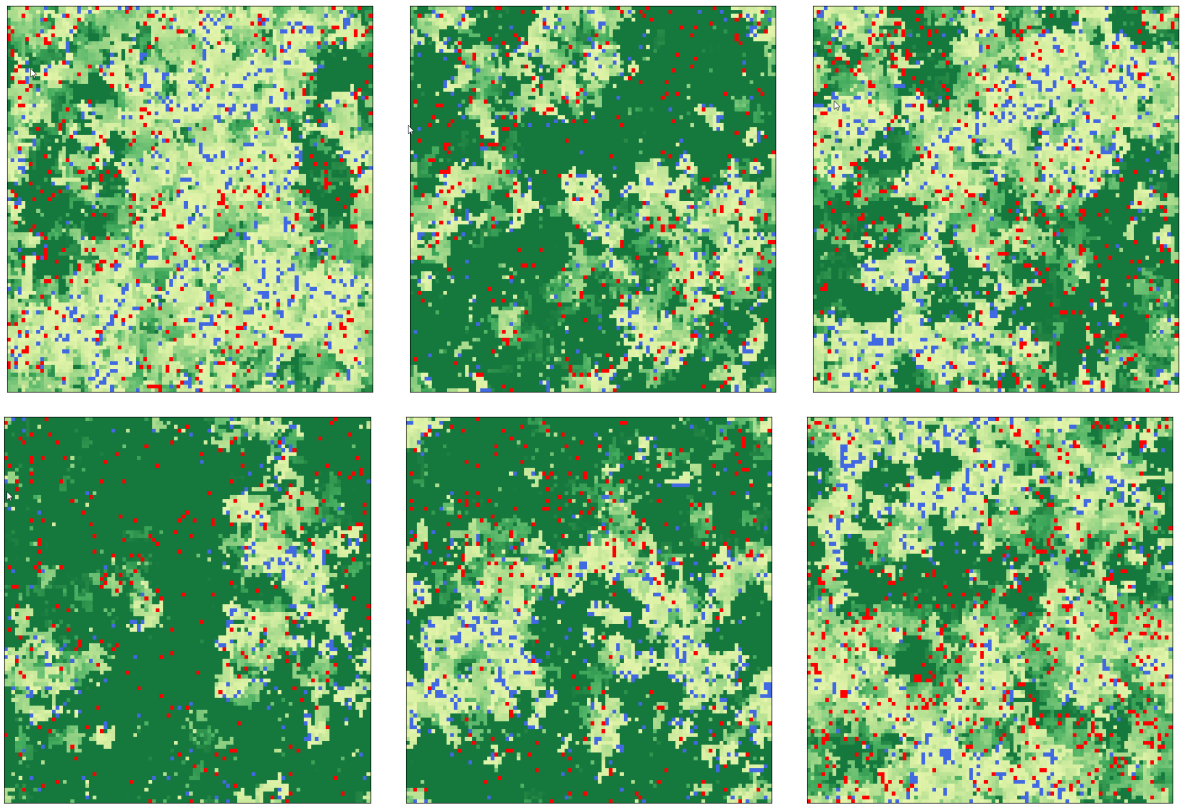

Figure 15: Maps after different numbers of turns; respectively, top to bottom, from left to right: 700, 1100, 1350, 3300, 5000, 5100

predator count is slowly raising. The fourth map (3300 turns) presents the all-time low of predators and start of the raising of the prey count. This is one of the moments in which the jungle is extremely overgrown. We have chosen the fifth map (5000 turns) to show one of interesting patterns formed by the populations on the map. In this moment prey are regaining their population, but the predator count is still decreasing and letting prey take control of next chunks of the map. The sixth map is the final state of the simulation which lasted about 5100 turns.

3.5 Medium-sized map, very high mutation ratio

In the last simulation, we used a medium-sized map (60×60) and increased the mutation ratio to 0.8 which is a very high value. We wanted to see the Darwin's theory of evolution in action. The configuration is shown in Figure 16. The population graph is presented in Figure 17.

We can see that for about 4800 epochs the simulation is stable. After this moment, predators are starting to win and prey are slowly dying out. Therefore, we have chosen two moments: the first one after circa 5400 turns (Figure 18), and the other at the end of the simulation, after about 5700 epochs 19).

After 5400 turns the prey are a few turns after their peak – they start becoming extinct. We can see that both species have evolved and adjusted their genome to the situation. The view range was increased and the energy consumption ratio decreased. Maximal animal energy changes are not visible on the histogram due to small changes and the widespread range. What is important, fear of predators has reached its maximal value – 8 – for many prey. It also seems

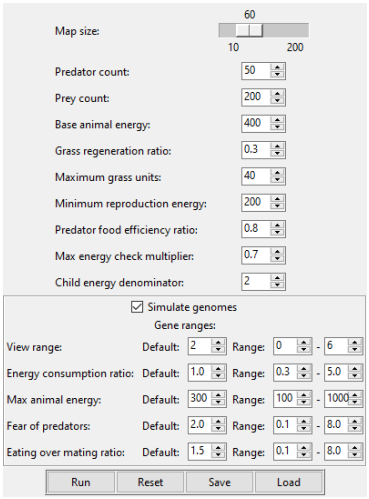

Figure 16: Simulation configuration

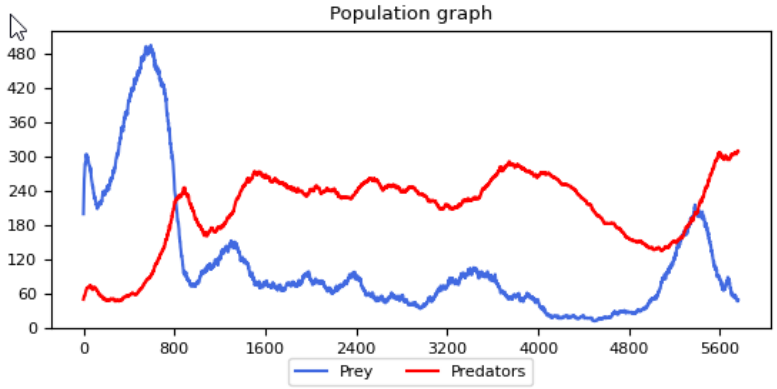

Figure 17: Population graph

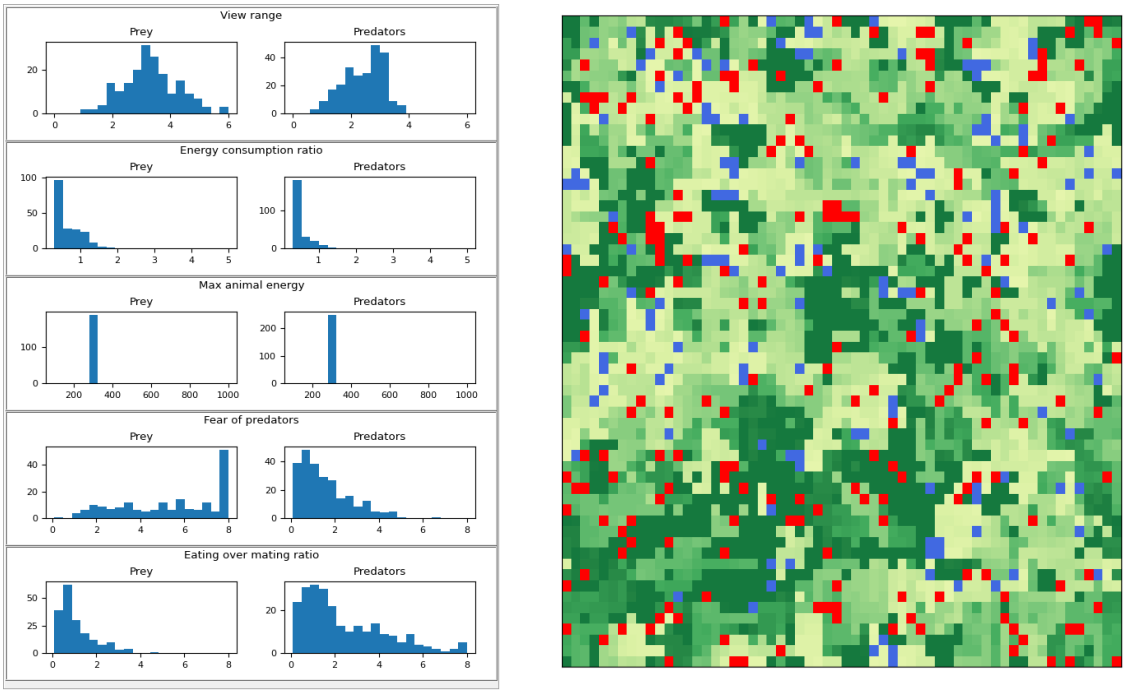

Figure 18: Gene histograms and the map after circa 5400 turns

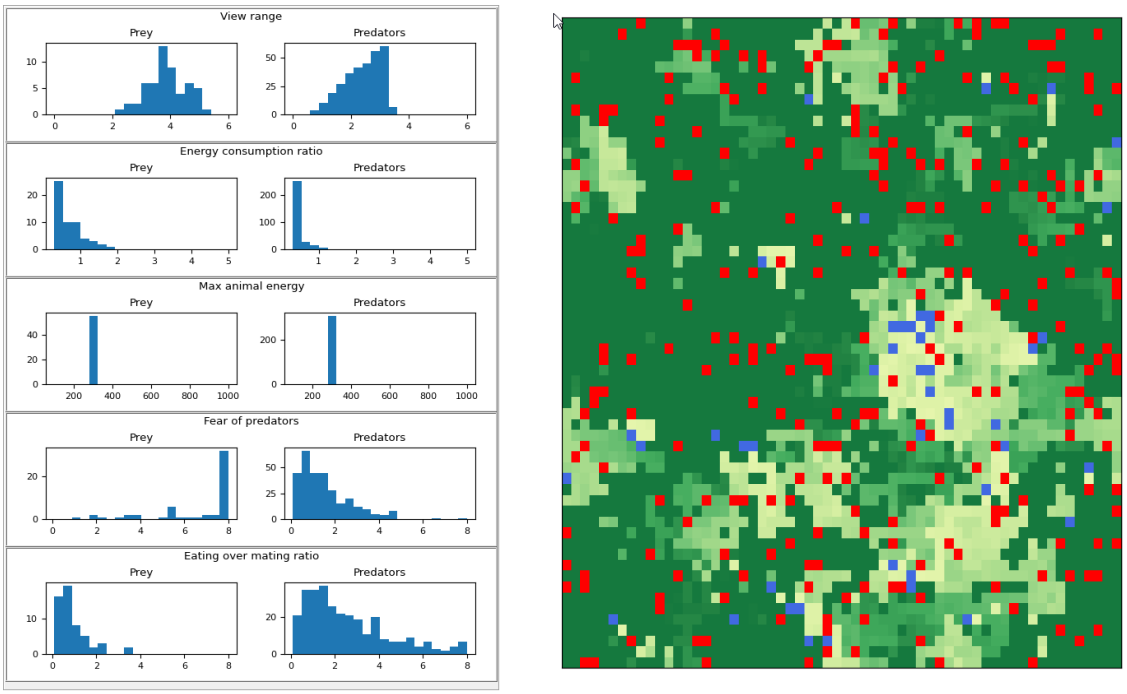

Figure 19: Gene histograms and the map after circa 5700 turns

that prey prefer mating to eating. They do not need to care about food, because plants grow everywhere. Furthermore, prey individuals preferring to mate produce more offspring which then increases the probability of mating preference in the next generation. Although this gene seem to have little impact on predators, some of them reached 8, which means that they prefer eating to mating. We can compare this histogram with a completely random fear of predators histogram for predators. A bias towards higher values is clearly visible.

After a few hundred turns, only the strongest prey have survived. Those with smaller view range, fear of predators or with higher energy consumption ratio died out throughout the simulation. Therefore, the theory of Darwin is working. Unfortunately, in this configuration of the simulation, predators have the upper hand, and thus, ultimately, both species will become extinct.

4 Summary and conclusions

To conclude, we find the project successful. We managed to develop an extensive application with a clear user interface. Due to its high configurability, it enables achieving various simulation results including preserving balanced populations compliant with the Lotka-Volterra model or proving the Darwin's theory of evolution.

There are a few possibilities of expanding the project. The most significant flaw in the application is its performance – it does not perform well for big maps. Since the operations are thread-safe, we could add multithreading. We could also implement more operations like matrix additions or multiplications instead of regular loops. Another expansion could be adding more genes, e.g. speed or attractiveness.

References

[1] *Predator-Prey Interaction*. URL: https://www2.nau.edu/lrm22/lessons/predator_prey/predator_prey.html (visited on 03/26/2022).

[2] T. A. Obaid. "The predator-prey model simulation". In: *Basrah Journal of science* 31.2 (2013), pp. 103–109.

[3] *NetLogo*. URL: https://ccl.northwestern.edu/netlogo/.

[4] *p1003/Predator-Prey-Simulation*. URL: https://github.com/p1003/Predator-Prey-Simulation.

[5] *rdmorel/predator-prey-simulation*. URL: https://github.com/rdmorel/predator-prey-simulation.