A computational approach to Bacterial Response Mechanism: Evolution in Artificial Cells driven by Two-component Regulatory System

Department of Informatics

Luis Morales Layja

26/04/2024

Abstract

In this research project, a computational model has been developed to emulate the Two-component Regulatory System, a response mechanism to external stimuli observed in bacteria. The stimulus-response process has been abstracted and implemented in an agent-based model to simulate bacterial behavior. Factors such as longevity and offspring are considered as direct indicators of fitness, and a genetic algorithm is used to explore the evolution of this mechanism in response to the simulated environment. Additionally, the resulting behavior is compared with that generated by a Feedforward Neural Network Controller (FNN) applied to the same agents. The results show a clear trend of the mechanism to produce emergent solutions for environmental challenges, producing satisfactory behaviours in relationship with the FNN. Even though, a more detailed exploration and more challenges would be needed to test the model made beyond this study.

1 Introduction

The understanding and comprehension of biological organisms are widely applied across various fields, from medicine and health—particularly in combating infections and diseases [8], where synthetic biology plays a crucial role—to the development of bio-inspired models for autonomous and computational systems [9]. Moreover, computational biology has deepened our knowledge of processes occurring in biological systems through simulations. These simulations range from modeling animal behaviors, such as collective decision-making [11], studying behaviors in animals for which the underlying causes are not yet fully understood [4] [7], to the modeling of each part of a cell and the modelling of its behaviour [6]. What's more, the study of Adaptive and Evolutionary Sytems can intersect this area. According to [3], evolution is the genetic and hereditary change that occurs transgenerationally due to natural selection, while adaptation is a process by which an organism learns to live under circumstances that threaten its survival, this process could involve physiological changes, but not genetic changes, as occurs in evolution. Computationally speaking, it refers to the modeling of systems or agents capable of adapting to an environment and finding ways to solve problems through both evolutionarily inspired algorithms and algorithms based on modifying their own characteristics. All living organisms are evolutionary and adaptive by nature [3]. Therefore, the study and simulation of these not only serve to develop robust biologically inspired systems, such as autonomous and control systems, and robotics but are also useful for advancing the understanding and comprehension of any such organism in question.

Consequently, this research work proposes to develop and implement a biologically inspired mechanism present in most bacteria. Given that bacteria possess a high degree of adaptability to new and stressful scenarios, it is hypothesized that this mechanism could serve as a robust and highly useful option for the creation of biologically inspired adaptive or autonomous agents, whether physical or virtual, while also encouraging the imitation of these types of mechanisms directly extracted from living organisms to advance in their understanding and prediction of behavior through computationally accurate representation of them. Specifically, it will implement the Two-component Regulatory System, which involves a sensor protein called "histidine kinase," which detects external stimuli, and a "response regulator protein," which receives the signal and mediates changes in gene expression, thus affecting cellular functions such as motility and metabolic responses, among others [5]. Additionally, these systems are critical for the adaptation of bacteria to changing environmental conditions and various stressors, making bacteria extremely adaptable and capable of surviving in hostile environments. This underscores their importance, and why they are considered potential targets for the development of new antimicrobial strategies [5]. Although there is a good amount of information in the literature on how this system is constituted and possible positive uses [2] [5] [1], there are hardly any computational models that represent this system. For example, [10] investigates how reciprocal regulation within Two-component systems with a bifunctional sensor kinase can contribute to ultrasensitivity in the signaling behavior of the system. Thus, there is little or no exploration in the literature on models of this system.

An agent-based model is proposed, in which the attributes of the agent (bacteria) such as entropy, offspring quantity, stored energy, and an environment with zones conducive to extending its life such as its ability to reproduce, as well as zones that affect its integrity, among other things, are used. The fitness measure is represented by a proportional relationship between the number of offspring obtained and the amount of time lived. Once the Two-component Regulatory System is abstractly implemented so that the agents act based on the stimuli entering and the responses exiting this system, and through a process of evolution (genetic algorithm), the parameters of the mechanism are adjusted, this same process will be analogously performed with a Feedforward Neural Network for comparison. The following data from the population evolution process will be collected to evaluate the performance of both mechanisms and contrast them: 1) Average fitness per generation. 2) Average lifespan. 3) Average number of offspring. 4) Standard deviation of average fitness per generation. 5) Standard deviation of average lifespan. 6) Standard deviation of average number of offspring. 7) Genetic diversity per generation. This aims to provide a clear evaluation of the performance and validity of this mechanism in terms of the performance of a well-established controller in the literature, such as the Feedforward Neural Network.

2 Methods

2.1 Flow of the program

Figure 1 illustrates the flow of the simulation. As can be observed, the genetic algorithm calls upon the environment, which contains all the objects and cells. Within this environment, the cell receives stimuli represented by the inputs; these inputs are the entry point to the Two-component Regulatory System, which then produces an output signal. This output dictates potential movement directions for the cell and its ability to reproduce. Subsequently, these actions are updated in the environment, and the cycle repeats. Only at the end of the generation is the cell evaluated, and its fitness assessment is obtained by the genetic algorithm. This algorithm assigns a genotype to the Two-component Regulatory System and calls the environment again for the next generation, until the number of generations is complete.

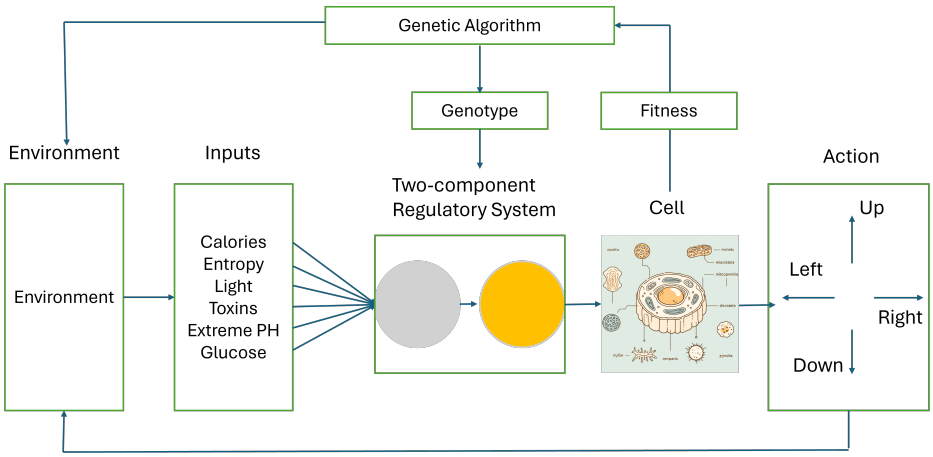

Figure 1: Flow of the program.

Once the cells die, they change from their natural color (blue) to red, and do not perform any further actions. It's also important to emphasize that in the first generation, the controller's genotype is randomly initialized, and that it's not possible to perform more than one action at a time. Therefore, if a cell reproduces, it cannot move.

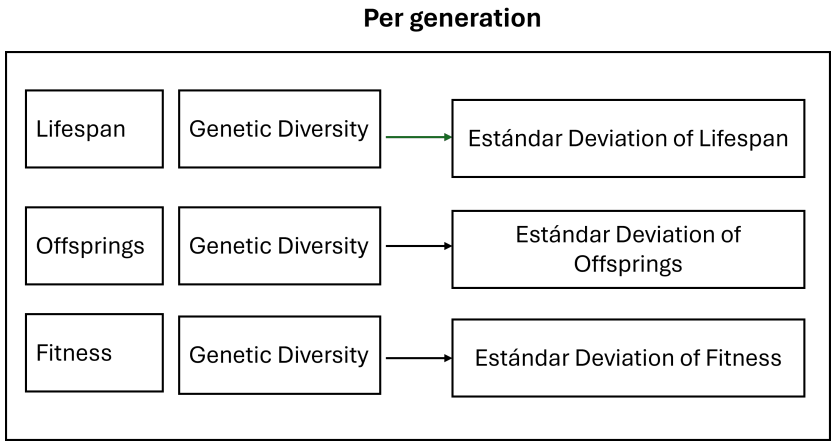

Figure 2: Metrics to analyze after simulation.

At the end of the simulation, the metrics of the entire population shown by Figure 2 are automatically obtained. As said, the metrics are gathered by generation, so they are an average by generation.

2.2 The environments

For this project, the behavior of agents in two different environments is analyzed and evaluated. Environment number one is represented by Figure 3, while environment number two by Figure 4.

Environment number one is easier to inhabit and is designed so that agents can find a simple solution to the challenge of survival. The yellow circles represent light sources that promote longer life for the cells, which the cells might seek, while the red ones represent toxins, which negatively affect the agent's integrity. Lastly, green circles represent units of glucose, giving energy to the cells.

Environment number two is really difficult to inhabit, and the only way to have longevity is by staying in the center of the map or at least not on the edges. It is expected that the agents...

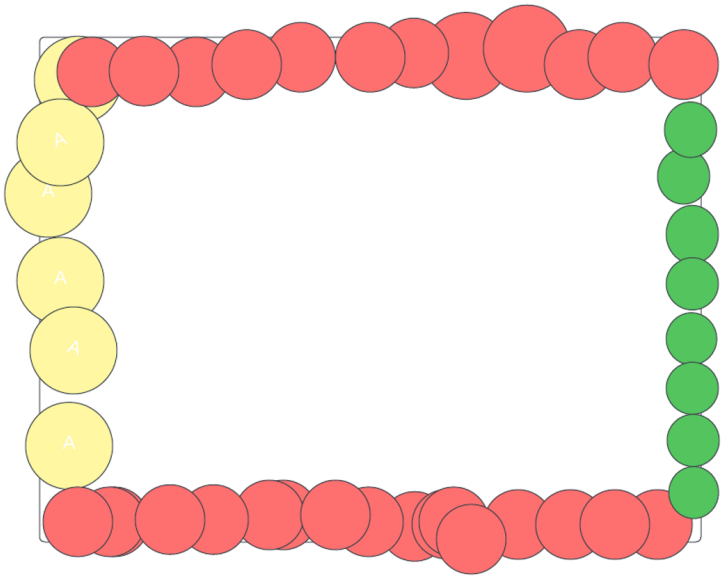

Figure 3: Environment 1.

2.3 The Cell: fitness, offspring and lifespan

All cells are represented by blue circles which have five possible actions: move up, move down, move right, move left, and reproduce.

The most important attributes of each cell are shown in Figure 5. To simplify visualization, these attributes are categorized into four different groups. The cell has constant attributes such as metabolism and the maximum amount of entropy it can sustain. Essentially, entropy refers to the degree of disorganization within the cellular structure, and maximum entropy is the greatest amount the cell can withstand before dying. In the simulation, entropy is synonymous with life. What distinguishes a living cell from a dead one is its level of entropy. Basal metabolism is the amount of calories the cell must expend just to continue living. In purple are the attributes related to the cell's senses, and these encompass everything that enters as input to its controller. In orange are the attributes of its controller, which could be a "neural network" or "TCRS controller." This means that the cell may have either a Feed Forward Neural Network or the Two-components Regulatory System instantiated, and the attribute 'layer sizes' refers to the initial random weights of its controllers. Lastly, the attributes in green are those directly related to its reproductive success and longevity, which change over time.

2.3.1 Fitness

Fitness is an attribute that changes with each timestep, and its formula is:

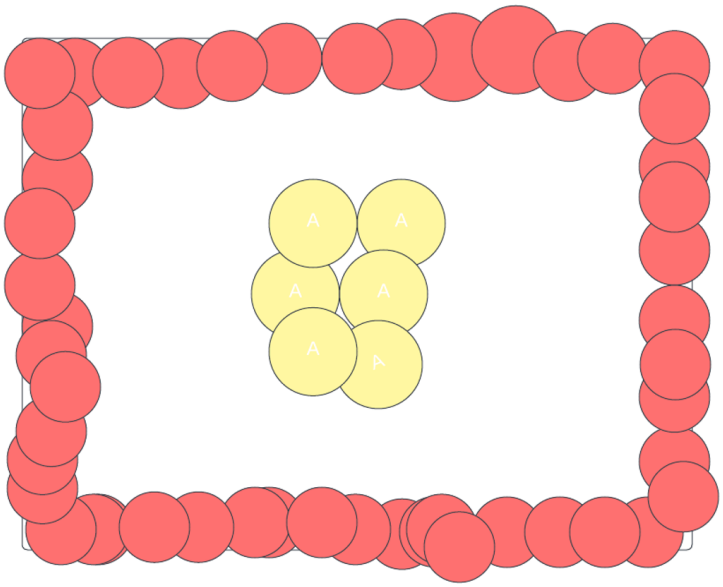

Figure 4: Environment 2.

$$fitness = (weighttimealive * timealive) + (weightnumoffsprings * numoffspring) \quad (1)$$

Where "weight time alive" and "weight num offspring" represent constants within the function, subject to the degree of fitness that should be added for each step of life or offspring produced.

2.3.2 Offspring

The action of producing offspring adds one to the attribute self.num offspring and, to simulate the cost of having a child, it adds +35 to the entropy and subtracts -400 from the calories that the cell has at that moment.

2.3.3 Lifespan

Lifespan refers to the average life duration of the cells in a generation. The attribute self.time alive of the cell, like the other attributes mentioned in this section, is dynamic and calculated at each time step. Each time step equates to adding +1 to the self.time alive attribute, which represents the total lifetime that a cell has had so far.

2.4 Feedforward Neural Network Controller

A Feedforward Neural Network is used for comparison with the controller of interest, the Two-component Regulatory System. The network consists of three layers. The input layer has six neurons, the hidden layer also has six neurons, and the output layer contains five neurons. The inputs and outputs of the neural network are subject to the specifications discussed in section 2.1. The visual representation is shown in Figure 6.

$$y = \sum_{i=1}^{n} w_i x_i + b$$

where y is the output of the neuron, x_i are the inputs, w_i are the weights associated with each input, b is the bias term, and n is the number of inputs.

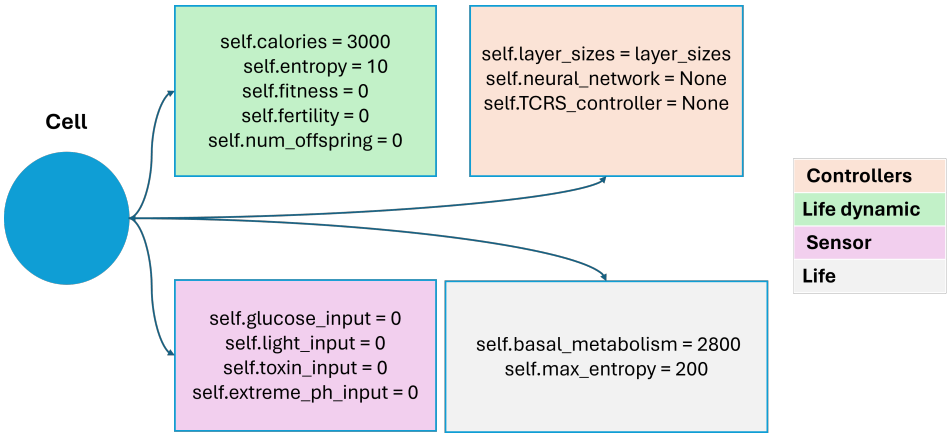

Figure 5: Most important attributes of the cell.

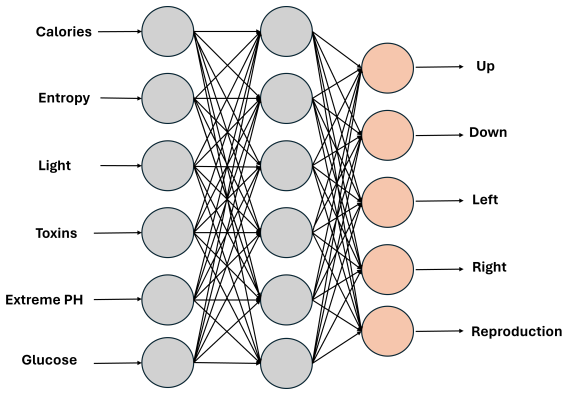

Figure 6: Feedforward Neural Network Controller used in as part of the project.

2.5 Two-component Regulatory System (TCRS)

The Two-component Regulatory System (TCRS) implemented in this model is inspired by the biological control systems found in bacteria, which use these mechanisms to respond to environmental stimuli. This computational analogue aims to mimic the way such systems process information through a sequence of sensor and regulator components.

In this model, the TCRS consists of two main components:

1. **Sensor**: This component receives external inputs and processes them to form an internal state. This state captures the combined effect of various environmental factors on the cell [5].
2. **Regulator**: Based on the internal state provided by the Sensor, the Regulator component outputs decisions that guide the cell's actions, such as movement or reproduction [5].

The mathematical representation proposed in this methodology and research of the TCRS is as follows:

$$S(t) = \tanh\left(\mathbf{W}_s \cdot \mathbf{x}(t)\right) \tag{2}$$

$$O(t) = \frac{\exp\left(S(t) \times \mathbf{W}_o\right)}{\sum \exp\left(S(t) \times \mathbf{W}_o\right)} \tag{3}$$

Where:

- $\mathbf{x}(t)$ represents the input vector at time t.
- $\mathbf{W}_s$ are the weights of the sensor component, mapping inputs to the sensor state.
- $S(t)$ is the sensor state at time t.
- $\mathbf{W}_o$ are the weights of the regulator component, which uses the sensor state to compute the output probabilities.
- $O(t)$ is the output vector at time t, representing action probabilities.

The sensor uses a hyperbolic tangent function (tanh) to ensure that the internal state remains within a manageable range, mimicking the saturation effects seen in biological systems, and also allowing for negative signals representing inhibition. The regulator part employs a softmax function to convert the linear activations into a probabilistic output, ensuring that the sum of probabilities for all possible actions equals one. This setup allows the model to probabilistically choose actions in a way that resembles decision-making processes in natural organisms.

2.6 Evolution

The evolution process was such that cells that did not reproduce in one generation could not advance to the next. This approach is intended to more faithfully reflect real-life processes where the only way to maintain progeny is through reproduction. As depicted in Algorithm 1, which is a pseudocode of the genetic algorithm presented in this research, "reproducers" is the vector of individuals that reproduced, sorted in descending order by fitness. The algorithm iterates over this vector and mutates the individual at the current position (asexual reproduction) to generate the individual for the next generation. This is done once per individual. Since there will typically be fewer individuals than in the original population, if the current population matrix is not filled, this process is repeated with individuals who have more offspring than the current round number until it is filled. This repetition is managed with the variables current round and reproduction rounds. Current round is the number of the current "round" or iteration, and reproduction rounds is the maximum number of offspring produced by any individual. This means that if it is the second round, and an individual only reproduced once, it cannot reproduce again; only those who reproduced more can. Furthermore, if the matrix is still not filled and no more individuals are left to reproduce, then this process is carried out with each individual in "reproducers".

Algorithm 1 Genetic Algorithm for Evolutionary Simulation

```
for generation from 1 to max_generations do
    Collect cells with offspring into reproducers
    Sort reproducers by fitness and number of offspring
    if reproducers is empty then
        return                                          ▷ End if no cells reproduced
    end if
    Initialize new_generation as empty list
    Find maximum number of rounds based on offspring numbers
    Set current_round to 1
    while new_generation is smaller than original population size do
        for each cell in reproducers do
            if cell's offspring more than current_round then
                new_cell ← cell.reproduce()
                Add new_cell to new_generation
                if new_generation reaches original population size then
                    break
                end if
            end if
        end for
        if current_round equals reproduction_rounds then
            Set current_round to 1                      ▷ Restart reproduction rounds if needed
        else
            Increment current_round
        end if
    end while
    Remove all old cells from the environment
    Add new_generation to the environment
    Evaluate fitness of each cell in new_generation
    Collect data for analysis and logging
end for
```

3 Results

It's important to highlight that simulations were conducted using different mutation parameters with the goal of ensuring that potential values for mutation rates or mutation strengths that might yield better results are not overlooked. Therefore, various metrics were measured for the same scenario and the same controller, with the difference being the mutations used. It should be emphasized that the mutation rate is the frequency with which mutations occur, and the strength is the intensity of these occurrences. This means that the difference in the weight of the neural network before and after mutation is ten percent.

As established in the introduction, there are six metrics: The first is the average fitness per generation. The second metric is the standard deviation of the average fitness per generation, which provides a visualization of the amount of genetic diversity in the population. The third metric is the average lifespan per generation, and similarly, the fourth is the genetic diversity in terms of the standard deviation of the average lifespan. Lastly, the average number of offspring per generation.

3.1 First Environment

3.1.1 Agents with Feedforward Neural Network Controller

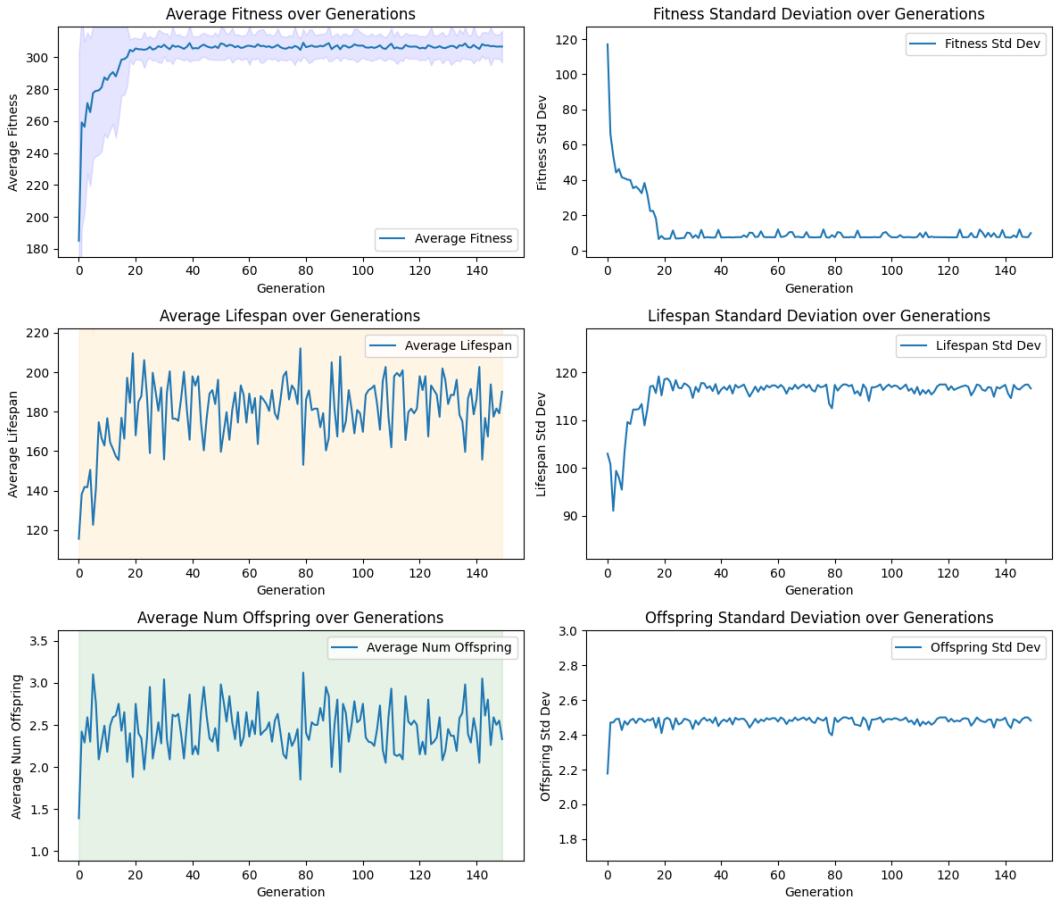

Figure 7: First environment FFN perfomance metrics with mutation rate = 0.1 and mutation strength = 0.1.

As can be seen in Figure 7, the average fitness per generation evolves correctly, and it can be observed that the maximum peak occurs around generation number twenty and remains stable until generation number 140. This could indicate that the mutation rate is lower than optimal, or simply that the agents found a fairly optimal solution to the challenges of the environment. Similarly, we can observe that the genetic diversity in terms of the standard deviation of the average fitness per

generation is around less than 10. This could suggest the previously mentioned issue: the mutation rate or mutation strength might be low.

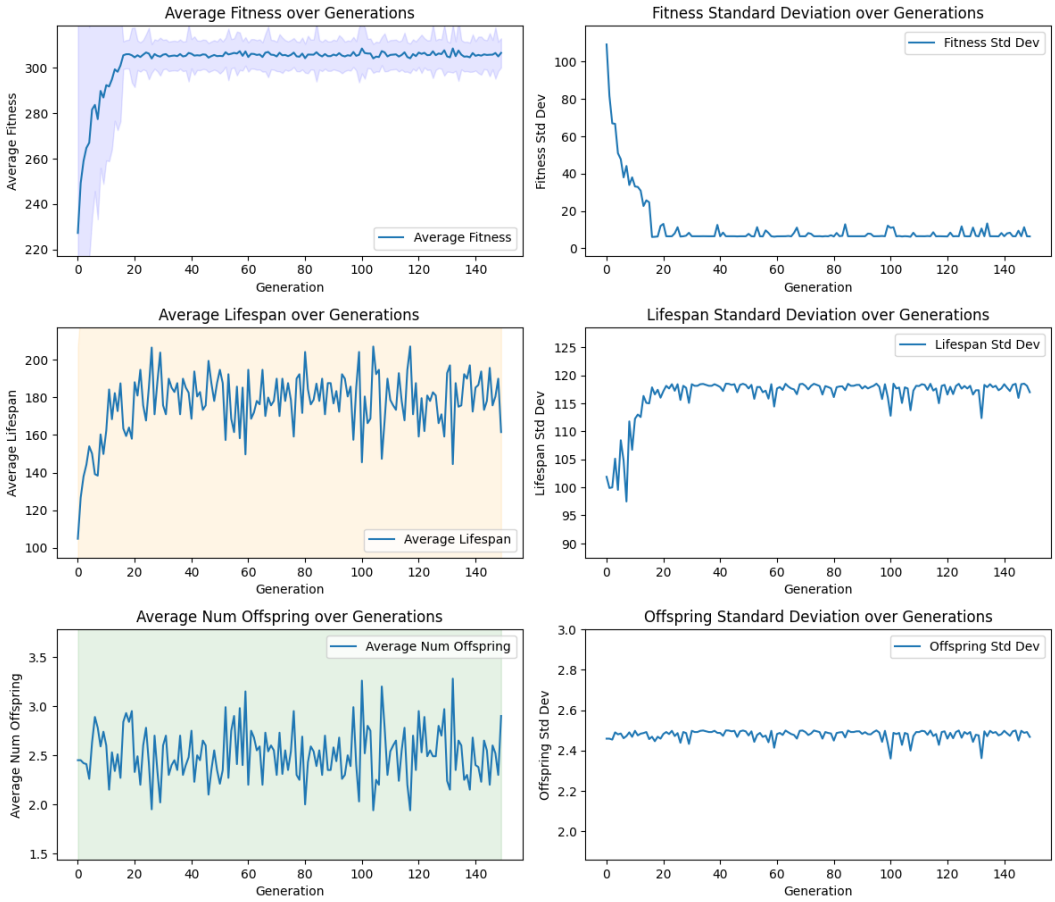

Figure 8: First environment FFN perfomance metrics with mutation rate = 0.15 and mutation strength = 0.12.

In Figure number eight, the mutation rate and strength are significantly increased, moving from a value of 0.1 and 0.05 respectively (equivalent to 10 percent and five percent) to 0.15 and 0.12, respectively. As observed, both the genetic diversity and the average fitness do not appear too different. It is noted that the standard deviation of the average fitness remains very low. Regarding the average lifespan and the average number of offspring, a similar behavior is observed in both situations.

3.1.2 Agents with Two-components Regulatory System

As shown in Figure number nine, the performance of the Two-components Regulatory System (TCRS) is observed under the same scenario and mutation rates as with the FNN. As can be seen, this controller also evolves successfully. There is a huge difference in genetic diversity using this controller compared to using the FNN, even though the mutation rate and strength are the same. In this case, the genetic diversity is around 20.

It is also very important to highlight that, although the evolution appears correct, the fitness value started at 80 and its highest point is around 220. This shows a clearly inferior performance compared to others. Despite this, in Figure number 14 a very similar behavior can be noted. It starts from around 80, up to around 2020. Considering that in Figure 14 the same controller was used for a much more challenging scenario, it is possible that Figure number nine was "erroneously" labeled.

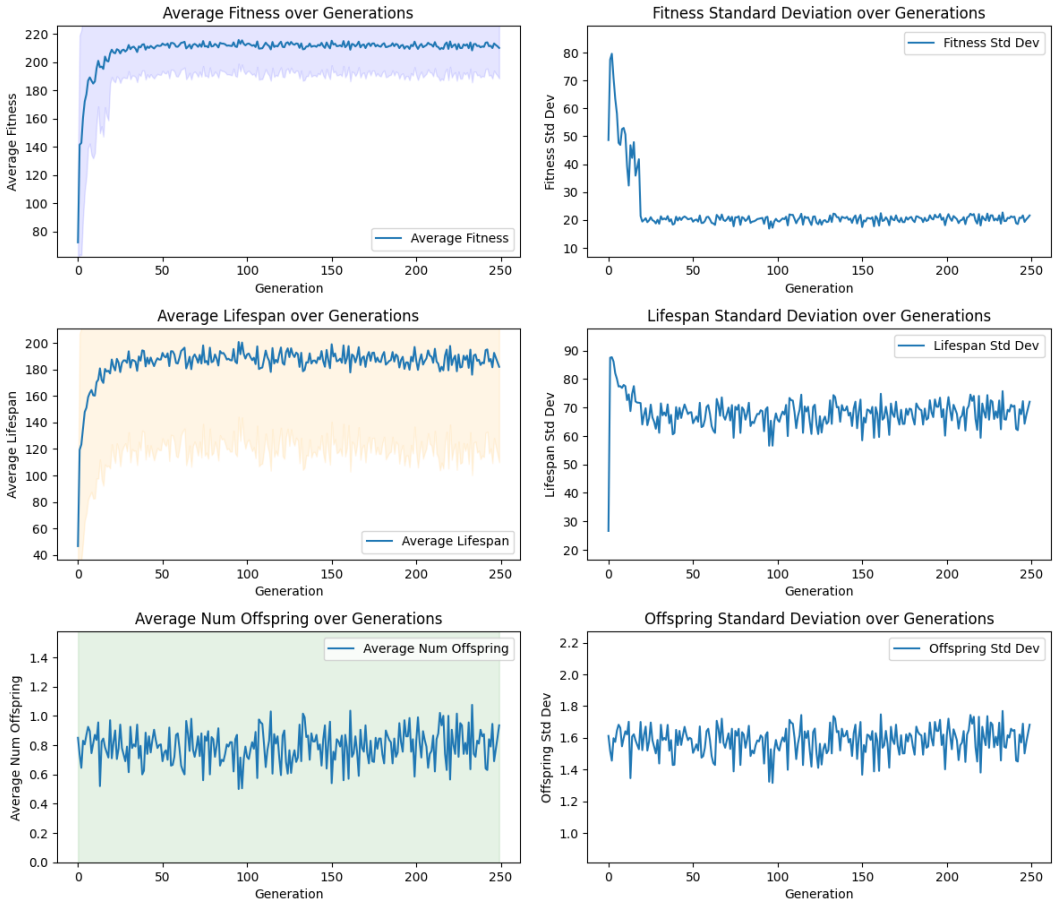

Figure 9: First environment TCRS perfomance metrics with mutation rate = 0.1 and mutation strength = 0.1.

3.2 Second Environment

In this section, the performances of both controllers are evaluated in the same way as for the first environment. In this case, the challenge is greater, and a lower overall fitness is expected, but good evolution is nonetheless anticipated.

3.2.1 Agents with Feedforward Neural Network Controller

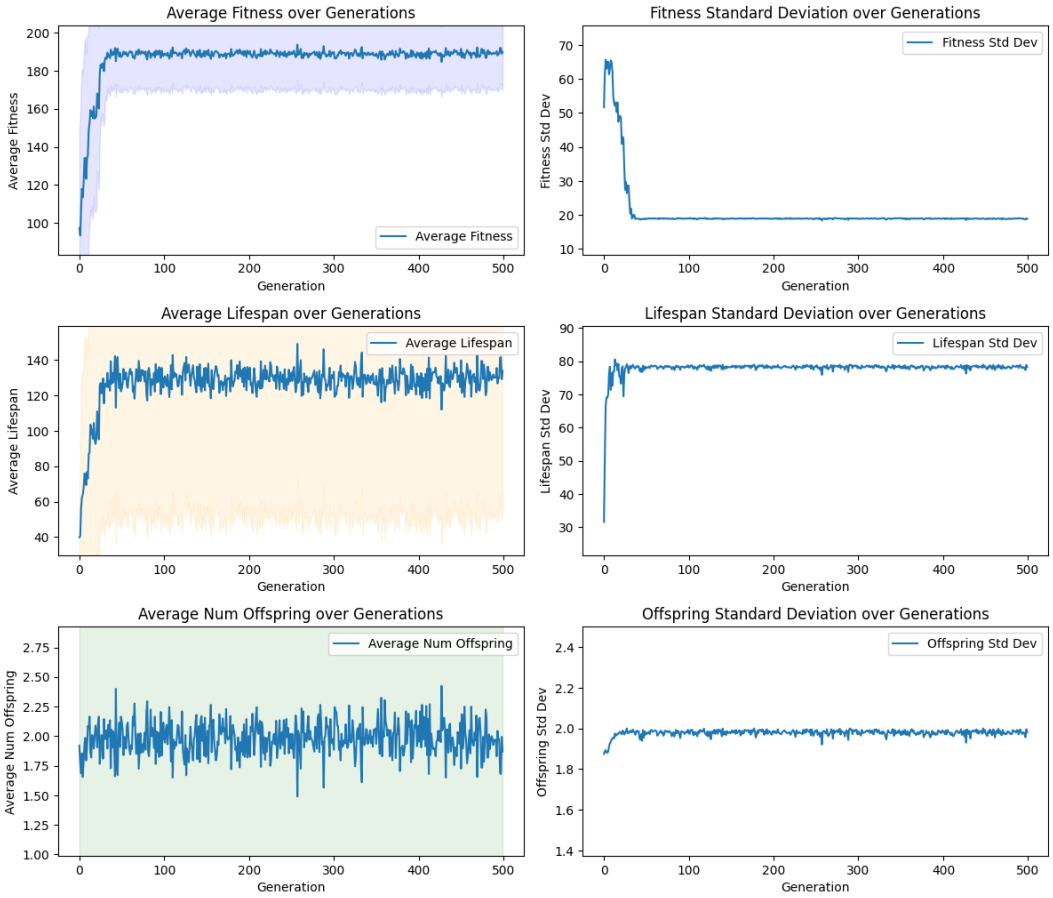

Figure 10: Second environment FFN perfomance metrics with mutation rate = 0.1 and mutation strength = 0.1.

It is notable how the FNN's total fitness decreased, as expected. Despite this, its evolution over time increased from around 100 to 200 in fitness, nearly doubling the initial performance. It is important to highlight that, in general, the genetic diversity is greater than in the first environment. On the other hand, the TCRS actually achieved a slightly better performance in two of the tests compared to the FNN. This superiority occurred when the mutation rate and strength were 20 percent and 10 percent respectively for the TCRS. It is possible that in Figure 16, the TCRS also slightly surpassed the FNN, although it is not entirely clear.

3.2.2 Agents with Two-components Regulatory System

As shown in the metrics evaluated between TCRS and FNN, es posible apreciar que en el caso del segundo environment, cuando la tasa y fuerza de mutacion en el TCRS llega al veinte por ciento, el fitness no se ve tan ruidoso como el del FNN. A pesar de esto lo mas notorio es que los comportamientos a nivel de fitness y diversidad genética se mantienen muy similares.

Overall, the results of the TCRS show a clear tendency to evolve in the same way as the FNN does, and it demonstrates the same performance or even slightly better in some cases.

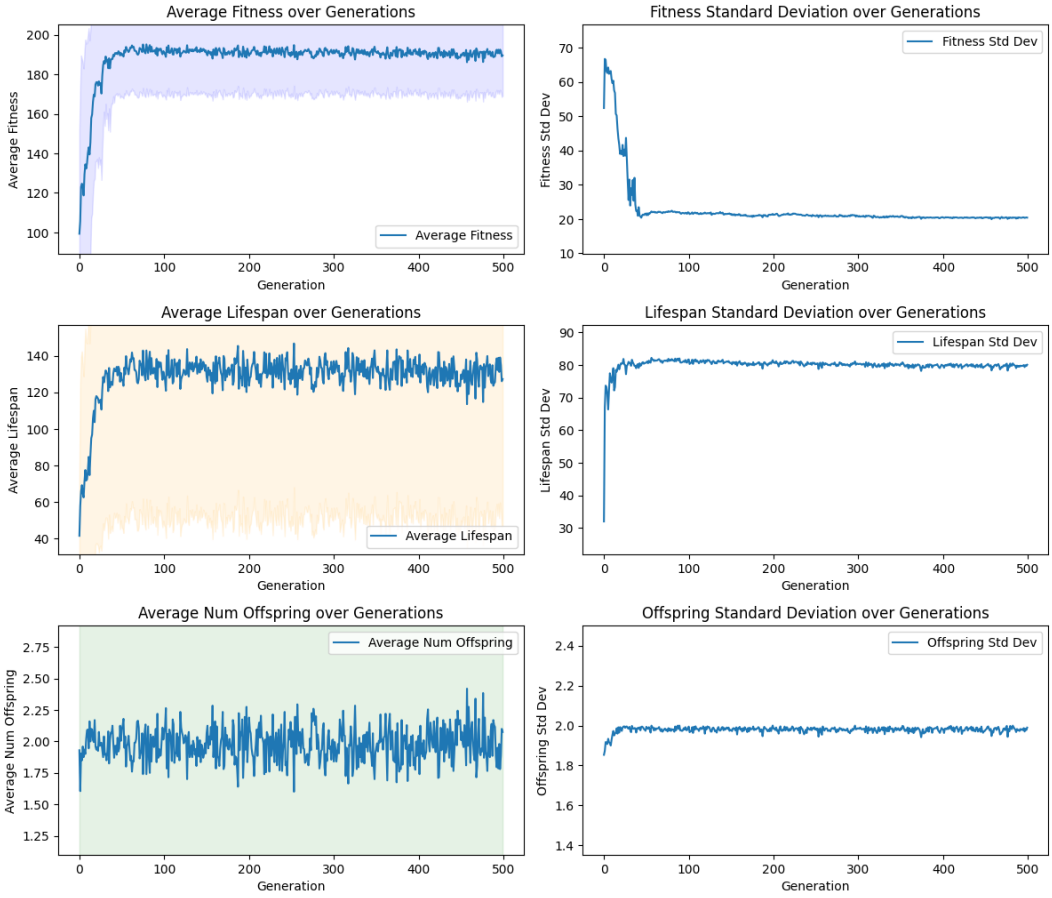

Figure 11: Second environment FNN perfomance metrics with mutation rate = .15 and mutation strength = 0.12

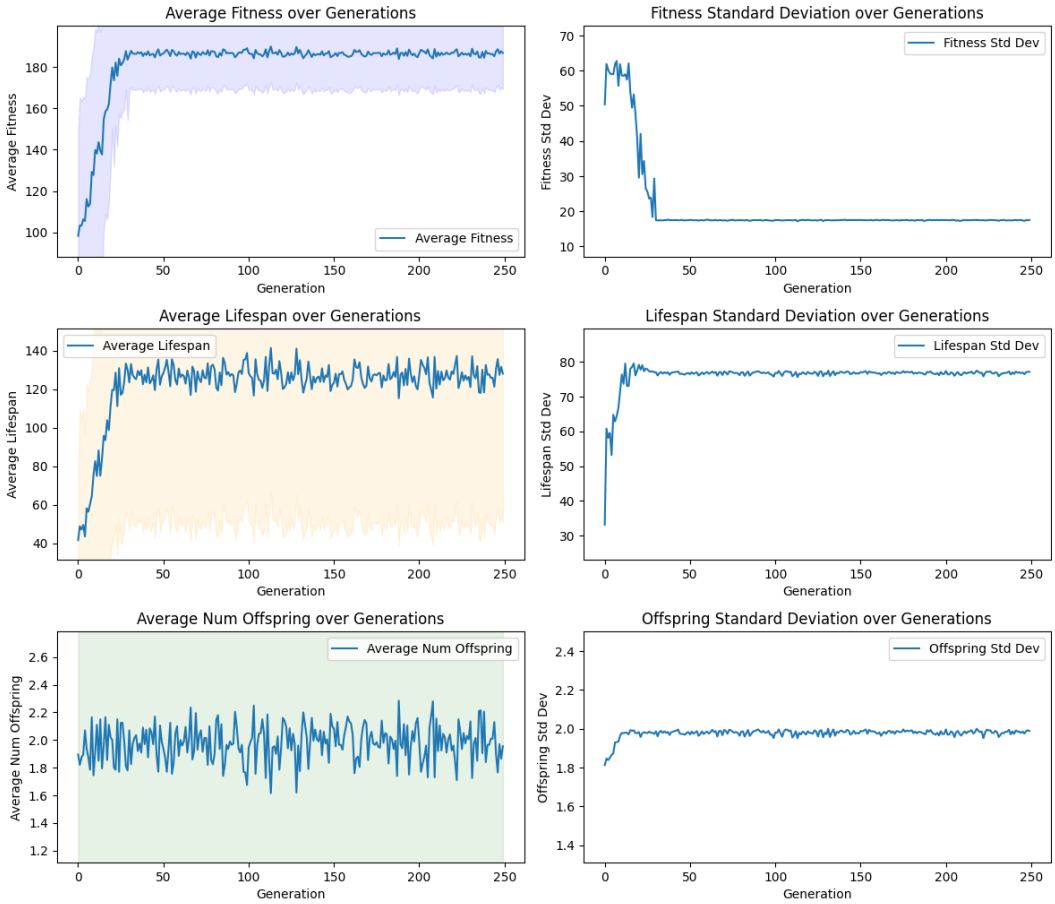

Figure 12: Second environmnet FNN perfomance metrics with mutation rate = .15 and mutation strength = 0.2

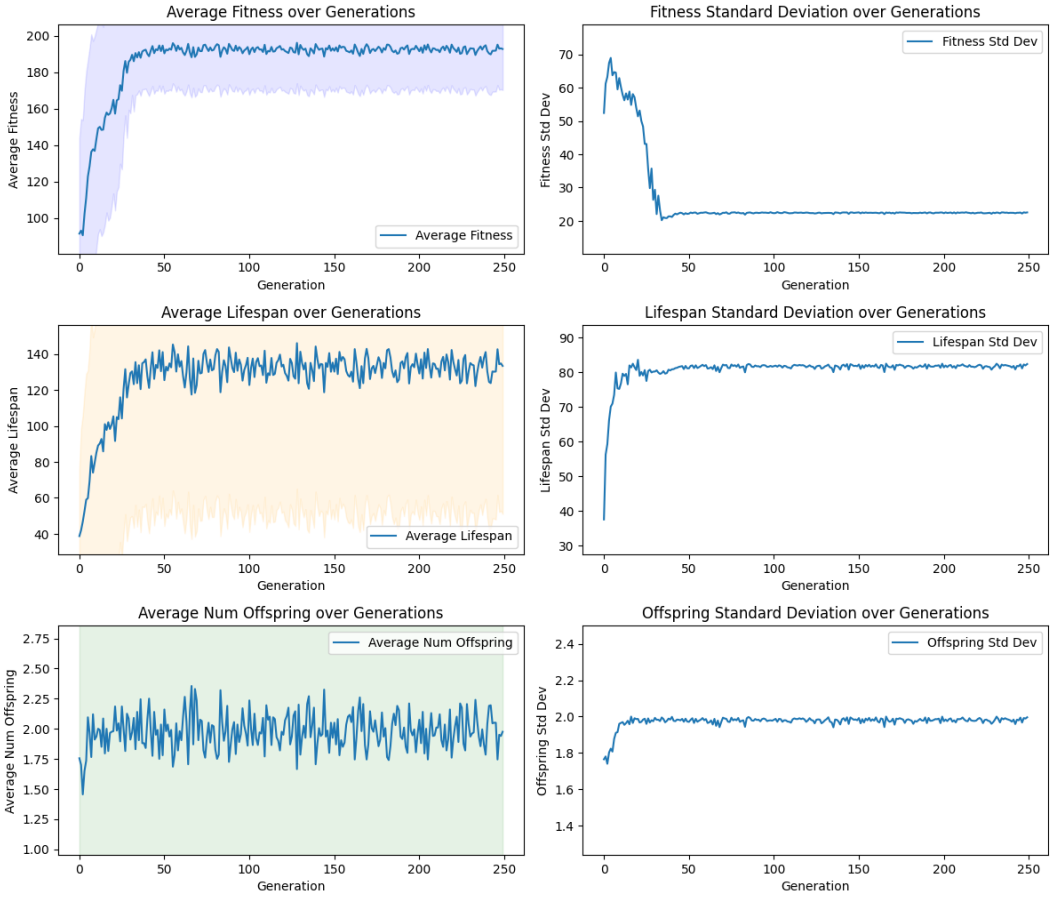

Figure 13: Second environment FNN perfomance metrics with mutation rate = .2 and mutation strength = 0.2

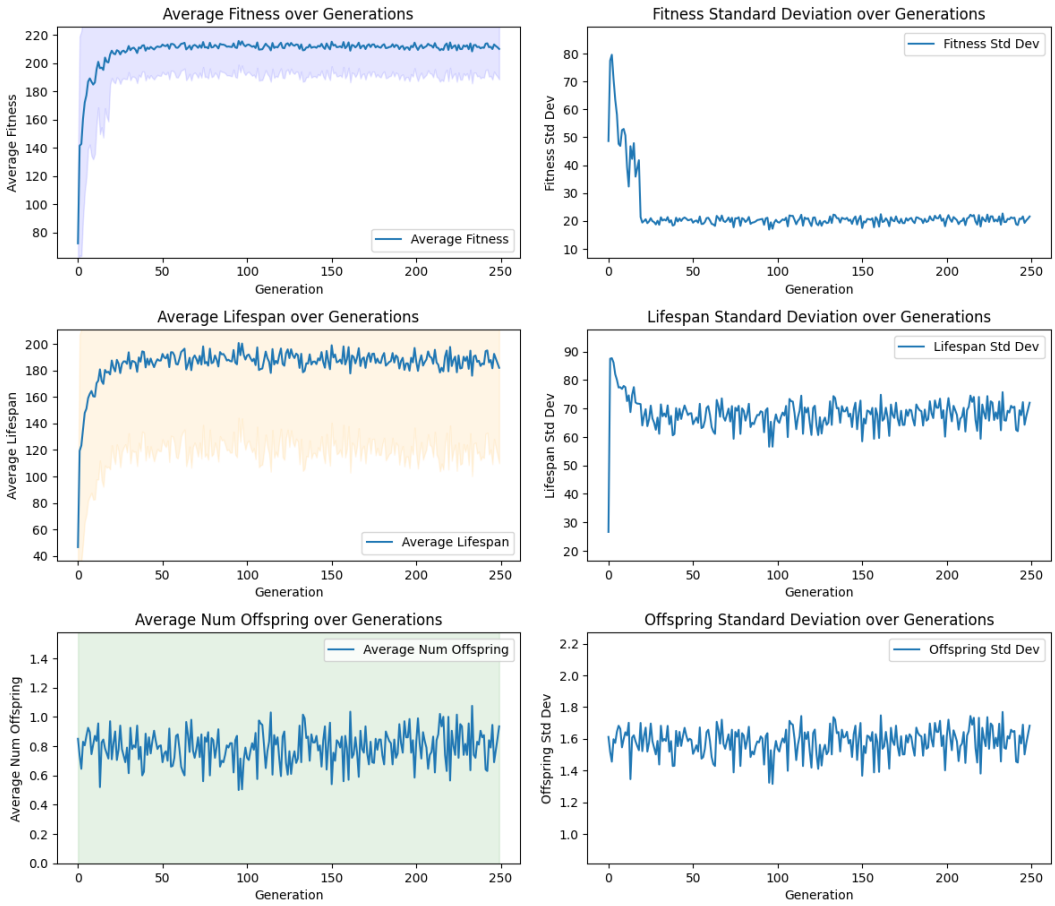

Figure 14: Second environment TCRS perfomance metrics with mutation rate = .1 and mutation strength = 0.1

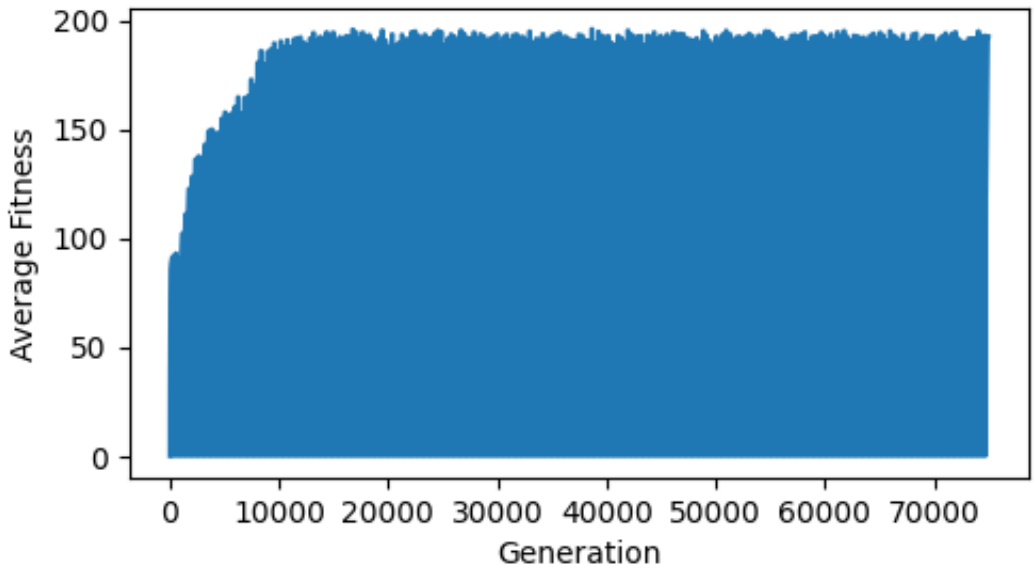

Figure 15: Second environment TCRS perfomance metrics with mutation rate = .1 and mutation strength = 0.05

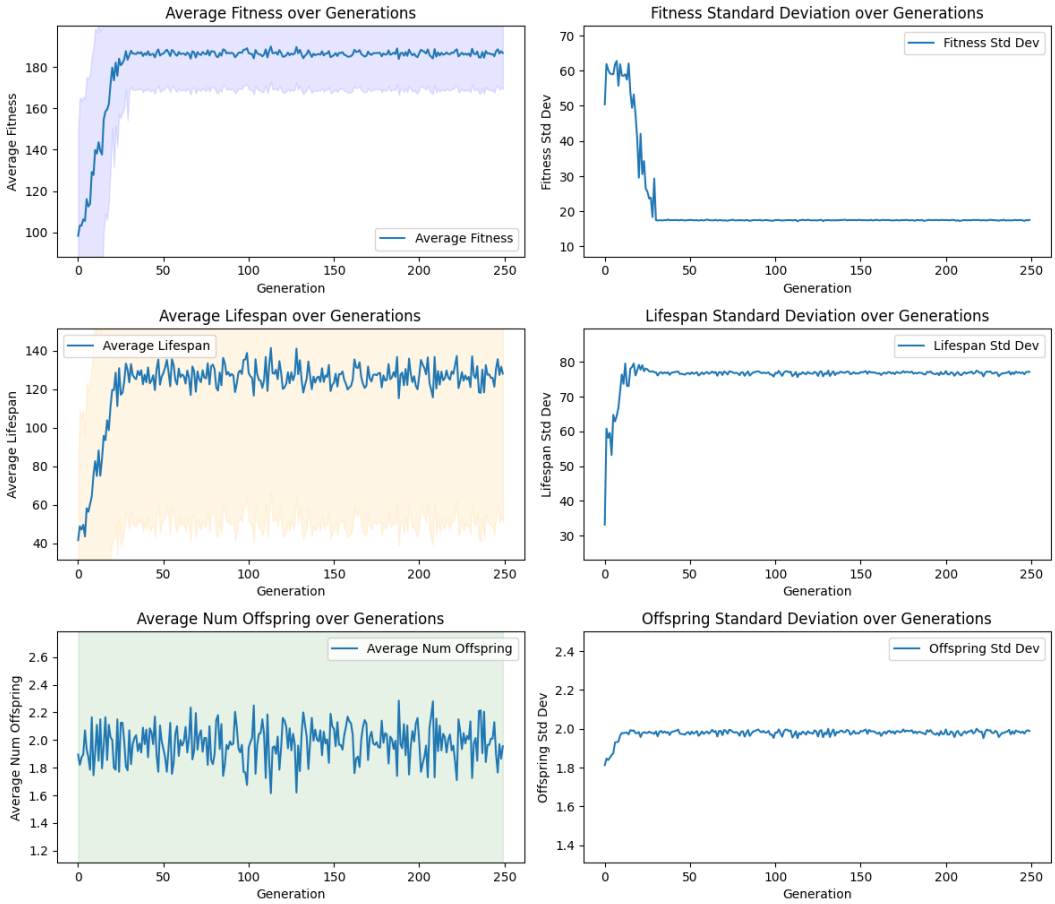

Figure 16: Second environment TCRS perfomance metrics with mutation rate = .15 and mutation strength = 0.20

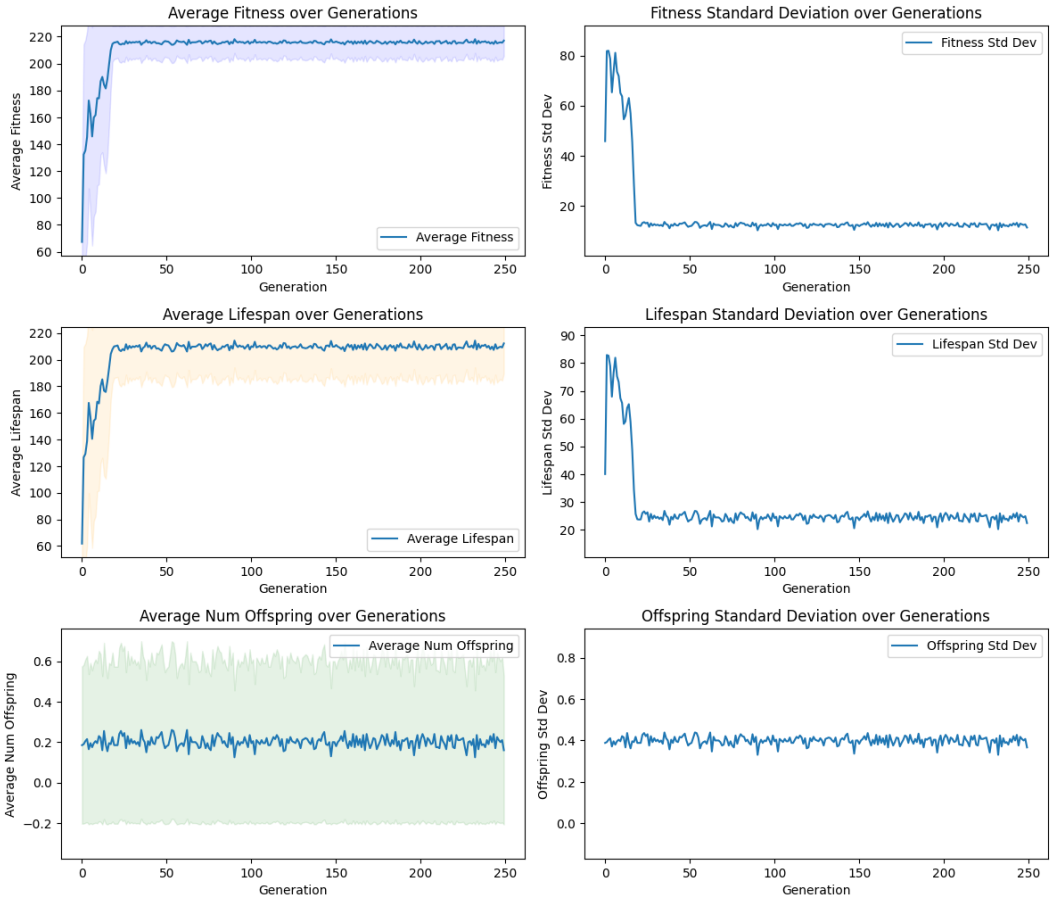

Figure 17: Second environment TCRS perfomance metrics with mutation rate = .2 and mutation strength = 0.2.

An important point to highlight is that it seems the mutation rate affects both controllers somewhat differently. While the TCRS appears to perform better with higher mutation rates, where the controller can be seen evolving quite well, as was the case in Figure 17 under the second scenario, where its mutation rate was 20 percent as well as the mutation strength, and its genetic diversity increased from above around 80, which was not common in any other case. It seemed to perform better in this scenario. In the case of the FNN, there does not appear to be an improvement by increasing such mutation.

4 Discussion

In this research work, computational modeling of the "Two-component Regulatory System" found in most bacteria was carried out, serving as a controller that processes environmental signals through a sensor protein and, through another protein, in this case, regulatory, takes actions to ensure its integrity and survival. Likewise, an agent-based model was created to simulate the behavior of bacteria in a challenging environment. The purpose was to compare this modeling of the mentioned mechanism with the results of a Feedforward neural network controller placed in the same agents to evolve in the same way through a genetic algorithm. The results of this work showed that the controller functioned at least at the same level, and in some cases slightly better than a Feedforward Neural Network. These findings give the green light for more detailed implementations of this mechanism to explore its possibilities, one of which could be the exploration to fight against other infectious agents as shown in [5], by being able to simulate and predict their behavior, or as shown in [8] where through synthetic biology methods the genome of a bacterium was modified to kill pathogens. In this case, progressing towards realistic and detailed simulation could complement synthetic biology in simulations of highly complex events that may not always be feasible in real life, but which can show predictions of their behaviors. Likewise, the reader is encouraged to take this idea since the mechanism in general appears to be less computationally and energetically costly than that seen through neural networks. Perhaps it could be a possibility for future implementations in autonomous systems and robotics, and bio-inspired electronic systems, as discussed in [9], having the additional advantage that the computational load of this model promises to be lower due to its simplicity and reduction of variables in modeling.

Despite the excellent results of the novel controller, there appears to have been a "leak" in one of the Figures of the results, since the TCRS showed a performance practically equal to that of the second environment in the first environment suddenly, which could have happened by mistake when placing the Figures. It also seems that the behavior of the controller in the presence of higher levels of mutation is not as affected as with the FNN, possibly suggesting that due to the particular configuration of this controller's structure, higher mutation rates should be sought.

Also, in this present work, there were many limitations in various aspects. First of all, this model is very simplistic and does not capture in detail the biological process carried out in bacteria and their molecules. Rather, it was an attempt to carry out an abstraction of the general mechanism and of the most important parts, rather than a detailed simulation of the interaction of its components. Possibly a more detailed model of this mechanism could result in more robust behaviors, at the cost of increasing computational cost. Another limitation was the lack of variability in the test environments. While it is suggested that this controller has been robust enough in the two environments presented to evolve correctly in other environments. It is always important to use a controller in different contexts, as there could be non-intuitive details that occurred. Similarly, an attempt could have been made to test with more variability in mutation. However, computational and time limitations did not allow for many more exhaustive tests. Likewise, technical knowledge may have limited the model, perhaps a detail was omitted that the author was not aware of, given the non-specialization in microbiology. On the other hand, one of the most important areas where the start of agent-based simulations specifically designed to simulate the internal process of the agent being explored can shed light on behaviors of, in this case bacteria, as in [1], although it can be expanded to any other biological agent as long as its internal biological process is modeled correctly, precisely, and in detail. This type of model can serve to predict the behavior of biological agents, making it possible to prevent diseases and even epidemics by being able to predict them.

References

[1] Ataman M. Hatzimanikatis V. Or D. Borer, B. Modeling metabolic networks of individual bacterial agents in heterogeneous and dynamic soil habitats (indimesh). *PLoS computational biology*, 2019.

[2] Pelletier A. Durmort C. Faure A. Kanonenberg K. Freton C. ... Orelle C. Diagne, A. M. Identification of a two-component regulatory system involved in antimicrobial peptide resistance in streptococcus pneumoniae. *PLoS Pathogens*, 2022.

[3] J. Fleagle. *Primate adaptation and evolution*. Academic press, 2013.

[4] Farina W. M. Grüter, C. The honeybee waggle dance: can we follow the steps? *Trends in ecology evolution*, 6(5):242–247, 2009.

[5] Kurushima J. Hashimoto Y. Tomita H. Hirakawa, H. Progress overview of bacterial two-component regulatory systems as potential targets for antimicrobial chemotherapy. *Antibiotics*, 2020.

[6] Sanghvi J. C. Macklin D. N. Gutschow M. V. Jacobs J. M. Bolival B. ... Covert M. W. Karr, J. R. A whole-cell computational model predicts phenotype from genotype. *cell*, 2012.

[7] Luis Morales Layja. Impact of private information factors leading bees to ignore the waggle dance when foraging. University of Sussex, Intelligence in Animals and Machines.

[8] M Peplow. Engineered bacterium hunts down pathogens. *Annual Reviews in Control*, 2013.

[9] A.; Dehnavi M.M.; Chhabra R. Pham, M.D.; D'Angiulli. From brain models to robotic embodied cognition: How does biological plausibility inform neuromorphic systems? *Brain Sci*, 2023.

[10] Straube R. Reciprocal regulation as a source of ultrasensitivity in two-component systems with a bifunctional sensor kinase. *PLOS Computational Biology*, 2014.

[11] Franks N. R. Ellis S. Okuda S. Marshall J. A. Robinson, E. J. A simple threshold rule is sufficient to explain sophisticated collective decision-making. *PloS one*, 6(5), 2011.