Artificial Intelligence Fundamentals

Practical work no 2

The Slow and The Calm:

Darwin's Edition

Report

2021/2022 – 3rd year, 2nd semester

Coimbra, 21st May 2022

Tiago Filipe Santa Ventura | 2019243695

Sancho Amaral Simões | 2019217590

1. Index

2. Goal

This report is related to the second practical work of the Artificial Intelligence Fundamentals subject, titled *The Slow and The Calm: Darwin's Edition* and aims to develop skills related to the analysis, development and implementation of adaptive agents. Therefore, it is based on the study of an Evolutionary Algorithm, and several scenarios with different parameters of evolution should be tested. In the *Unity* package provided, there are structures and primitives that integrate a simulation scenario that basically consists in sets of individuals, represented as cars, that attempt to finish a race circuit with, obviously, with some obstacles.

3. Introduction

In the context of the Artificial Intelligence subject, the project “The Slow and The Calm: Darwin’s Edition” was developed, recurring to the *Unity* platform, as a simulation environment. Adaptative agents were used as the main source of results. An agent is considered adaptive if it is capable of responding to other agents and/or its environment to some degree. At a minimum, this means that an agent must be able to react to a simple stimulus to make a direct, predetermined response to a particular event or environmental signal. More advanced forms of adaptation include the capacity to learn and evolve. These agents can change their behaviour based on experience. Common software techniques for learning are neural networks, *Bayesian* rules, credit assignments, and classifier rules. Examples of learning agents would be agents that can approve credit applications, analyse speech, and recognize and track targets. A primary technique for agent evolution usually involves genetic algorithms and genetic programming. Here, agents can literally be bred to fit specific purposes. For example, operation plans, circuitry, and software programs can prove to be more optimal that any product that a human can make in a reasonable amount of time.

A *Darwinian*-style evolution technique was used. Here, agents can be bred using genetic algorithms and then compete in a survival-of-the-fittest mode. In *Lamarckian*-style evolution, features acquired and information learned by an agent can be passed on to its offspring. The use of memes, or culturally transmitted information, is also a popular technique.

At its simplest, Darwin’s theory states that evolution is driven by small variations in traits that are amplified by natural selection. Organisms with beneficial traits are more likely to reproduce and thus pass on those traits to offspring than are organisms that have detrimental traits.

Darwin was unaware of how traits were passed from parents to offspring (making his insights all the more impressive), but we now know that an organism’s genotype, along with its developmental environment, is responsible for its phenotype (physical and behavioural traits). In general, new genotypes arise in offspring via random mutation of parent DNA, a mixing of genes from multiple parents (sexual reproduction), or both.

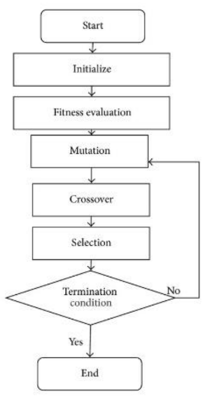


Fig. 1 - Flowchart of an adaptive agent

4. Implementation

Even though most of the structure that supports the genetic algorithm was implemented, such as the simulation scenery, phenotypes and population generation, some fundamental parts were missing. Those fundamental parts integrate the steps described in the previously presented flowchart: *recombination* (*crossover*), *mutation, parent and survivor selection*, and *fitness*. Also, it was made possible the parametrization of the genetic algorithm in terms of *crossover* and *mutation* probabilities and number of individuals to use in *parent* and *survivor selection*. The algorithms for the enumerated operations except the *fitness*, were considerably easy to implement, since their correspondent pseudocode was provided, so, the only thing left to do was to transpose de pseudocode to the *C#* programming language. As for the *fitness function*, the work group was responsible for implement it from scratch, having in mind that some of the environment properties such as the car *max distance*, *max distance time*, *max velocity*, *number of wheels*, *mass* and *completion* of the circuit were available.

In this project, the fitness function revealed to be the key factor to success, being this success the completion of the road present in two provided scenarios: *Gap Road* and *Hill Road*. As learned from the theoretical classes, the fitness function is a multivariable function that allows to evaluate the relevance of a certain individual contained in a population of a given generation *n* of the genetic algorithm. Since this function intervenes in the *selection* steps, it acts as a maximization function, with the goal of finding individuals with the best performance. With the intent of sharpening the performance of the *fitness function*, the group proceeded to a study of the environment the individuals are subject to, which will be explained ahead, in the *A priori analysis* section.

With the goal of having a better coding environment, some documentation, structuring, modularization of the existing/implemented code was performed.

In order to allow the choice of the fitness function in the genetic algorithm configuration, some additional logic was implemented: a static class that registers most of the developed fitness functions under the form of constants was created. In the genetic algorithm configuration file, one of these constants is chosen and then is applied in the selection moments.

The group came to the conclusion that some essential information about the environment and the individuals themselves was missing: properties such as the *max road length*, *max mass*, were not accessible from the scope where the fitness function is implemented. So, in order to fix that issue, additional infrastructure was implemented.

Since this project is something that takes time, due to the large number of experiments that have to be performed in order to retrieve relevant results, the developers tried to accelerate these, by altering the variables *Time.timeScale* and *Time.fixedDeltaTIme* to values greater than *1*. Unfortunately, this could not work because the physical calculations related with individual interactions with the environment (road) were greatly affected.

In spite of this sad conclusion, it was a fine attempt of increasing the efficiency of the simulation process.

5. *A priori* analysis

As mentioned in the previous section, an analysis of both simulation scenarios, *Gap Road* and *Hill Road*, was vital in order to have a better and superficial understanding of which individuals would be ideal to complete the road, and, therefore, allowing the implementation of a *fitness function* that better serves the maximization purpose of the genetic algorithm. So, the group proceeded to the analysis of the road in each scenario in the first couple of experiments and performed some logical and valid deductions.

5.1. *Gap Road*

This scenario consists of a *656m* road which presents pits that grow larger and larger with distance.

→ Small cars will easily fall into the pit.

→ Cars with low velocity will probably get stuck in the pit borders and may then fall into it.

→ Large cars will easily go above the pit.

→ Cars with high velocity will spend less time above the pit hence have a lower probability of getting stuck or falling into it.

→ Cars with wheels at different levels are less likely to get stuck.

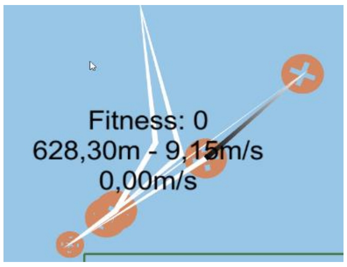


Fig. 2 – Example of a potentially ideal individual (long, fast, wheels at different levels)

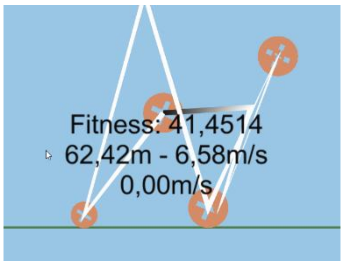


Fig. 3 – Example of an individual that is not ideal (short, slow)

5.1. *Hill Road*

This scenario consists of a 360m road with four main hills: the first and the second are relatively gentle, while the third and last are quite steep. The peaks of both first and second hills are a platform and the peak of the third hill   The goal is contained in the last climb.

→ Heavier cars will less likely to climb the hills due to effect of gravity on their weight.

→ Shorter cars will have a harder time bending the peaks of the first two hills.

→ Slower cars will not be so capable of fighting the effect of gravity.

→ Longer cars with an opening in the middle will find it easier to bend the peaks of the first two hills.

→ Lighter cars will experience less of the effect of gravity on their weight when going up hills.

→ Speedy cars will surpass the effect of gravity more easily.

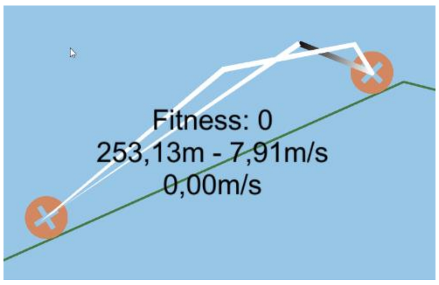


Fig. 4 – Example of a potentially ideal individual right before the third hill peak (light, speedy, longer and with an opening in the middle)

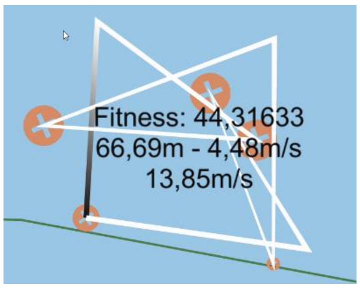


Fig. 5 – Example of an individual that is not ideal (small, heavy, slow)

6. Experimentation

The first experiments in both scenarios of the genetic algorithm were conducted with configurations specified in the following table (3 trials per configuration):

Parameter / Experiment	Mutation prob.	Elitism (survivors)	Tournament size	Crossover prob.	Generations num.
1	0.05	0	5	0.9	30
2	0.2				
3	0.05	2			
4	0.2				
5	0.05		2		

Fig. 6 – Initially used configurations of the genetic algorithm

5.1. *Gap Road*

For this scenario, a *fitness function* that simply returned the *max distance* travelled by the individual was sufficient in order to get cars which would cross the finish line. In other words, the larger the distance travelled by the individual the fittest he is. Since the total road length is accessible in the *fitness function*, thanks to the code improvements done in *4.* the value returned by it could be a normalized one (in the [0, 1] interval).

$$F_1 = \frac{MaxDistance}{RoadLength}$$

Other fitness functions were implemented but did not bring relevant results.

$$F_2 = \frac{1}{NumberOFWheels}$$

$$F_3 = MaxDistance + \begin{cases} NumberOfWheels * 25 \; if \; NumberOfWheels < 2 \\ \\ (NumberOfWheels - 4) * 10 \; if \; NumberOfWheel > 4 \end{cases}$$

The group decided to implement a more complex fitness function in order to possibly observe more interesting results:

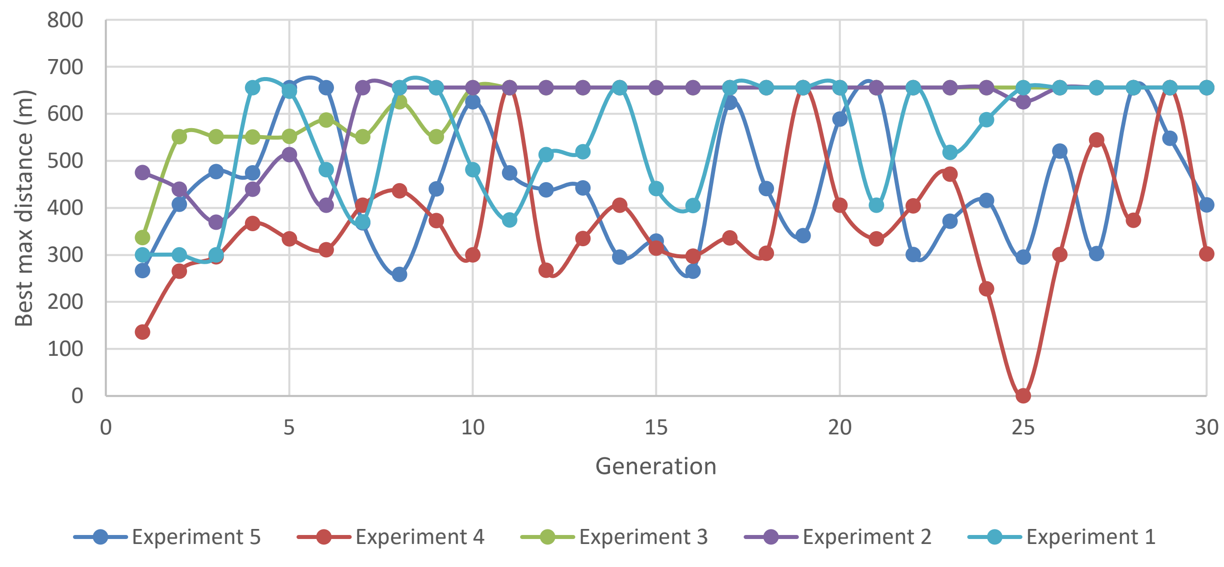
$$F_3 = MaxDistance * 4 * IsRoadComplete + 7 * MaxVelocity - 0.5 * Mass - 5 * MaxDistanceTime - \begin{cases} 300 \; if \; NumberOfWheels \leq 1 \\ 50 \; if \; NumberOfWheels == 2 \end{cases}$$

This fitness function would then compensate individuals with greater distance travelled and give a greater bonus to those who reached the goal. In addition, it would favour individuals with greater speed. Individuals with large masses and long travel times would be penalized. Regarding the logic of the wheels, it was assumed that a number of wheels less than or equal to one would be greatly penalized, as it would greatly impair the vehicle's ability to move. A number of wheels would be reasonable but could bring some limitations to the vehicle in terms of getting out of the pits. Above two wheels, there could benefits or there could be no effect in case the wheels were not able to come into contact with the surface.

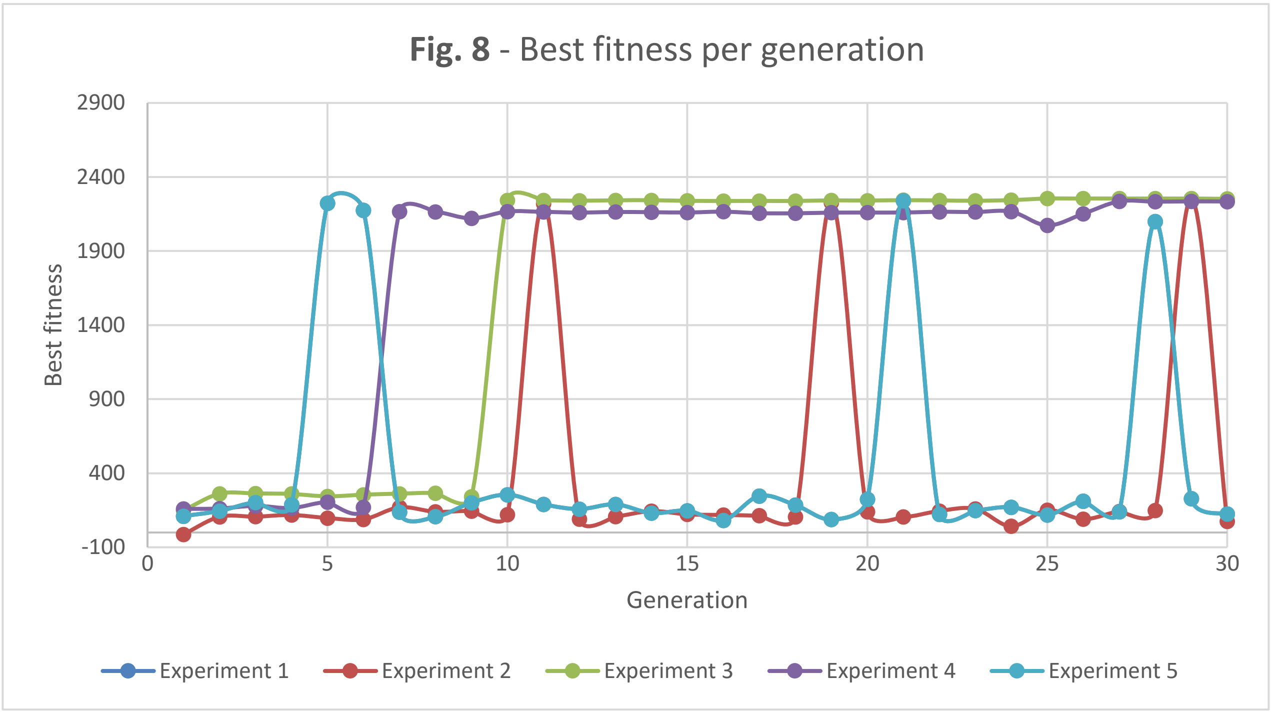
It is important to point out that the values of the coefficients represented in the above formula were obtained in an ad-hoc way, by carrying out additional experiments and inspecting the attributes of the individual in question through the debugger tool.

This fitness function returns values in an approximate range of *[-50, 2500]*. This interval is not exact, since at the time this formula was developed, the notion of the variability of the parameters that integrate it was not well understood.

Fig. 7 - Best max distance per generation



In the graph of Figure 7, the distinction between executions of the genetic algorithm that use or do not use *elitism* is very clear. When this is used, from the moment an individual is reached who travels the entire length of the road, the m*aximum distance* stabilizes at the value of *656* (experiences 3, 4, 5). Otherwise, much more oscillation is observed (experiments 1, 2).



In the graph of Figure 8, due to the compensation given to individuals who pass the goal, it is quite evident the moment when one of these same individuals is extracted. The fitness from then on may vary due to small variations in the completion time of the track or, perhaps, because individuals with a higher fitness appear.

5.2. *Hilll Road*

In this scenario, the vast majority of individuals show great difficulty in going over the third hill, namely its top (in the form of a V in reverse). This zone covers approximately the 230m mark. For this reason, a fitness function was initially developed that greatly rewards individuals whose maximum distance covered exceeds this value.

$$F_4 = \begin{cases} 0 \; if \; MaxDistance > 360 \; and \; not(IsRoadComplete) \\ \\ MaxDistance + \big((MaxDistance > 230) * (MaxDistance - \; 230) * 30\big) + IsRoadComplete * 1000 \end{cases}$$

The first branch of the fitness function is due to the fact that, in some experiments, namely those in which the number of individuals is very high (simulation grid), totally unconfigured and uncontrolled vehicles are occasionally generated and they fall infinitely, not passing the target, even if they have exceeded the 360m mark. These "mutants" are some of the outliers present in this genetic algorithm and, if they were not treated, they would continue to spoil possible optimal results, as their fitness would grow indefinitely.

The group initially detected that this fitness function was not showing very satisfactory results, most likely due to the fact that it was ignoring the individual's parameters that are particularly relevant in this scenario, such as their mass and max velocity. Therefore, the developers proceeded to the construction of a last fitness function, more complex, well-founded and, as it will be possible to observe below, a fitness function that will give much more interesting results.

$$F_5 = \begin{cases} 0 \; if \; MaxDistance > RoadLength \\ \\ (IsRoadComplete * \; 0.5 + not(IsRoadComplete) * 0.5 * \left(\frac{MaxDistance}{RoadLength}\right) + 0.3 * \left(1 \; - \frac{Mass}{MaxMass}\right) + \\ 1.5 * (MaxVelocity \; [n \; - 1] == 0) * 1 * not(MaxVelocity \; [n \; - 1] == 0) * \\ Clamp\left(\frac{MaxVelocity[n]}{MaxVelocity[n-1]}, min = 0, max = 1\right) + \; 0.05 * Gaussian(NumberOfWheels, mean = \\ 3, std = 2)) * 100 \qquad \text{(n is the time instant of measure of the max velocity)} \end{cases}$$

This fitness function appears to be overly complex, but it is actually quite simple. Each function of a parameter is assigned a weight so that the sum of the weights equals 1: the distance travelled is given a weight of 0.5, for the mass 0.3, for the maximum speed 0.15 and for the number of wheels 0.05. These weights were estimated based on the analysis of the Hill Road scenario and on the importance that the group considered for each individual parameter. For the logic used for the speed parameter, additional code was implemented that allowed the registration of the previous and current maximum speed. Regarding the function of the number of wheels, a gaussian curve of mean three (ideal number of wheels) and standard deviation of 2 is used.

It is important to note that the maximum mass is calculated by adding the maximum number of wheels weighed by their radius. maximum (number of bits reserved for the spoke of each wheel) and the sum of the maximum number of vectors weighed by its maximum length (number of bits reserved for the length of each vector).

This fact is particularly important since, in this way, it would be easier to make comparisons between the results obtained, that is, they would be commensurable, since they would always be comprised between the same values.

Fig. 9 - Best max distance per generation

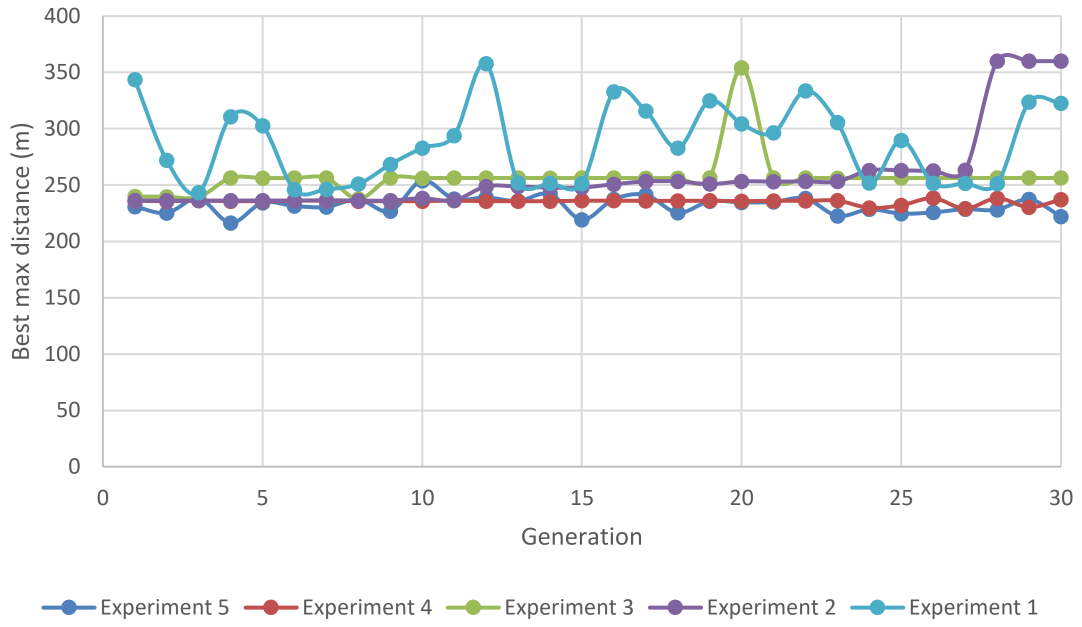
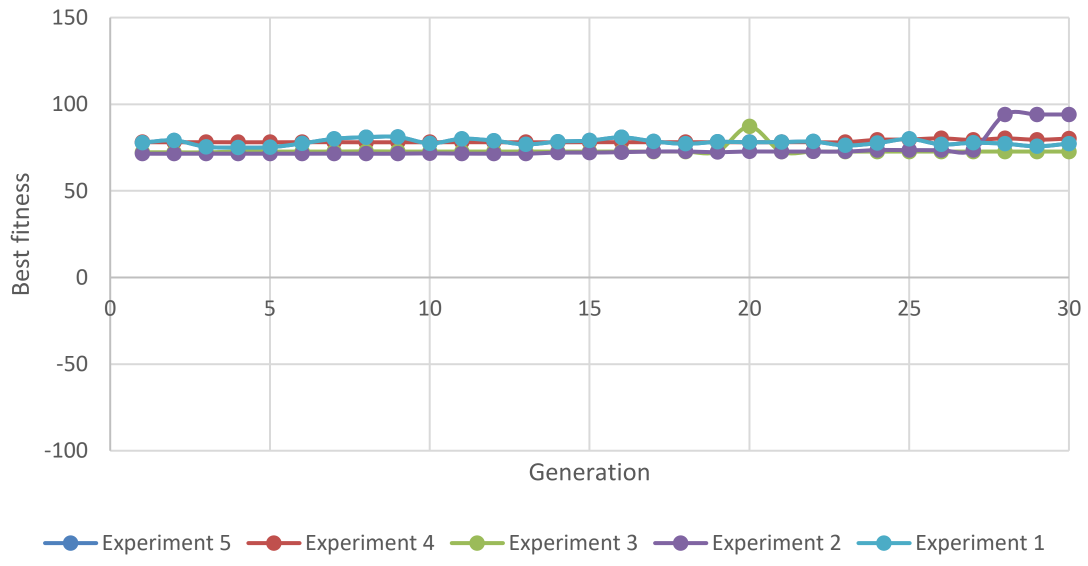


Fig. 10 - Best fitness per generation



The graph above shows the effects of the normalization of the fitness function, since the trend lines remain practically constant over the number of generations.

7. Final results

Through the use of the F_3 and F_5 fitness functions and, of course, with a lot of experimentation and some luck, the developers were able to find individuals capable of finishing both the Gap Road and Hill Road scenarios. Below are the characteristics of these same individuals.

Gap Road

Number of wheels: 8

Mass: 167.5

Time: 66.91882

Hill Road

Number of wheels: 8

Mass: 257

Time: 51.56127

6. Conclusion

In conclusion, it should be noted that the draft relating to this report has enabled for us to develop fundamental competencies in the area of AI through the design, implementation and testing of adaptive agents. Initially appeared to us to be quite complex but that, with the development of the work, we came to realize that through the modelling of the genetic algorithm it is much simpler to develop an adaptive agent in various test environments and that the ability of genetic algorithms in relation to development of the agents is quite high. This project allowed us to understand how the application of natural phenomena such as natural selection can be applied to computing, the importance of fitness function as a decisive factor in the selection of skilled individuals, the fundamentals of artificial intelligence of agents capable of adapting to the environment through recombination and mutation of genes. It is also noticeable that artificial intelligence still has much to develop both scientifically and technologically and the adaptive agents themselves as a target of study have a great capacity for evolution.

7. Bibliography

[1] Inteligência Artificial: Fundamentos e Aplicações, Ernesto Costa, Anabela Simões

[2] Artificial Intelligence - A Modern Approach., Russell, S., Norvig, P.

[3] Wikipedia, a enciclopédia livre, Evolutionary algorithm, https://en.wikipedia.org/wiki/Evolutionary_algorithm. Acedido em 16/05/2022.

[4] Evolutionary approaches towards AI: past, present, and future, Matthew Roos, https://towardsdatascience.com/evolutionary-approaches-towards-ai-past-present-and-future-b23ccb424e98 , Acedido em 15/05/2022.

[5] Adaptive Agent , James Odell, https://wiki.c2.com/?AdaptiveAgent Acedido em 17/05/2022.