University of Sussex

Artificial intelligence and adaptive systems

Adaptive systems

Evolutionary optimization of neural networks for autonomous robot navigation in webots simulation

Author:
Andres Salazar

Lecturer:
Dr. Chris Johnson

April 13, 2024

UNIVERSITY OF SUSSEX

Contents

List of Figures

List of Tables

Abstract

Robot exploration has a wide range of applications such as space, deep water and hard to reach places exploration. One task that these explorer robots must do perfectly is object avoidance. Due to the fact that if a robot crashes with any object and needs any kind of assistance or loses important features given the accident, is possible that the robot's mission fails losing the robot and the money invested in the expedition as there is no human to do maintenance to the robot. These potential scenarios motivated this experiment. The objective is to create a simulation with a single robot that must avoid all the objects in the environment and explore the map as much as possible given the highest fitness to the solutions that explore the most. A feed forward neural network was chosen as controller because of its effectiveness in multiple applications. Moreover, to optimize the neural network parameters and find the best configuration possible a genetic algorithm is proposed.

1 Introduction

Within robot exploration important features. Object avoidance is a relevant one as the ability of a robot to deal with objects in an unstructured environment is key to perform a task effectively and safely with low risk of collisions. Some applications to these robots are travel aids for blind pedestrians [1], human-robot collision avoidance to improve the coexistence of humans and robots [2] and industrial applications [3] [4].

Given that the environment, sensors, and tasks vary in the experiments and that the objective is to get the best performance possible. A genetic algorithm (GA) optimization is used to find the best neural network parameters set. A lot of experiments have been carried out testing the efficiency of using a GA to optimize neural network parameters. Applications are as complex as soccer robot avoidance studying the method of automatic extraction and optimization of fuzzy rules of fuzzy path planner by GA [5]. On the other hand, GA are used to tune the structure and parameters of a neural network [6] [7]. Furthermore, [8] brings together object avoidance and GA optimization, the environment was a simulation of the robot in the space, the results evidenced that the algorithm is effective in avoiding objects and improve the control performance of the robot. The previous related work motivated the question: Could a speed control based on a neural network be optimized effectively to explore as much as possible of an environment using a GA to optimize the neural network parameters?

To solve this question, a neural network is used and a simulation in Webots was created to test the different sets of parameters in the same environment. Also, the solutions that explore the environment the most are given high fitness. Due to the difference in the environment and sensors, the parameters optimization of the speed control based on a neural network is fundamental to attaining the best performance.

2 Methods

2.1 Proposed methodology

A neural network is proposed to be used in the controller of the robot as its viability has been proven multiple times [6] [7]. The optimization algorithm chosen to find the best parameters was a genetic algorithm. Additionally, the fitness function is the count of the squares explored by the robot in the whole trajectory.

Figure 1 shows the workflow of the experimentation. The GA used to create and evolve the genotypes population and used it as parameters for the neural network that is the agent controller. Then, the genotype is tested and the fitness is sent back to the GA.

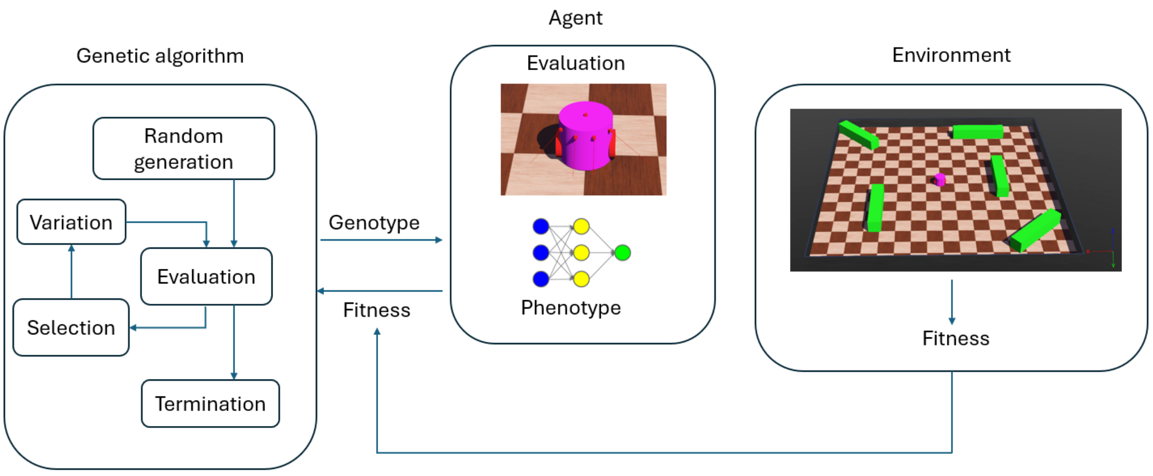


Figure 1: Experiment workflow.

The experiment was carried out with three different approaches:

- Evolving all the weights and bias of the neural network.
- Fixed bias values and evolving the weights.
- Increase the mutation probability and generations.

Moreover, the number of generations tested were 100, 200, 300 and the mutation rate were 0.1 and 0.2.

2.2 Simulation

The application used to simulate the environment and the robot was Webots. Webots is an open source and multi-platform desktop application used to simulate robots. It provides a complete development environment to model, program and simulate robots. It has been designed for a professional use, and it is widely used in industry, education and research. Cyberbotics Ltd. maintains Webots as its main product continuously since 1998 [9].

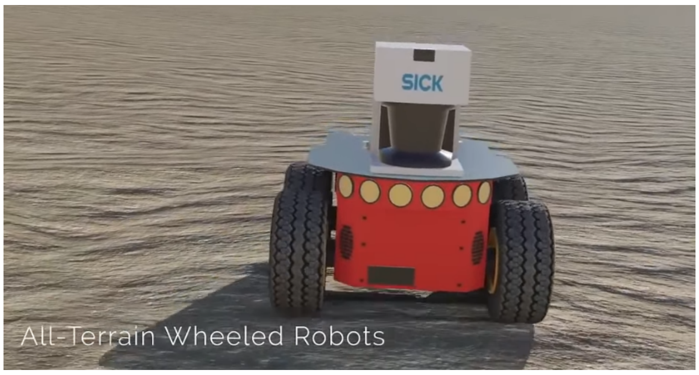


Figure 2: Webots simulation example [9].

The robot used to test the algorithm was a custom one(Figure 3). The robot has four distance sensors and two wheels. Also, is equipped with a GPS at the top of its structure to record its position during the simulation as precisely as possible.

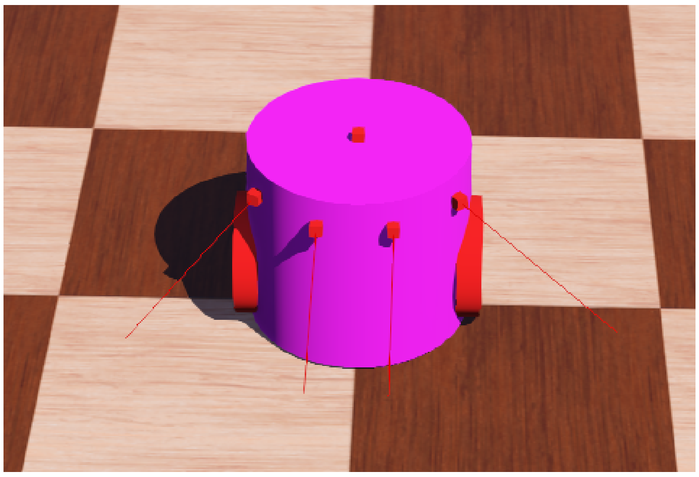


Figure 3: Custom Webots robot.

Finally, the environment is a square shape map. Throughout the whole map were placed different objects that could be identified by the robot sensors. The map had walls to keep robot in the area and test the algorithm with the same conditions in every iteration.

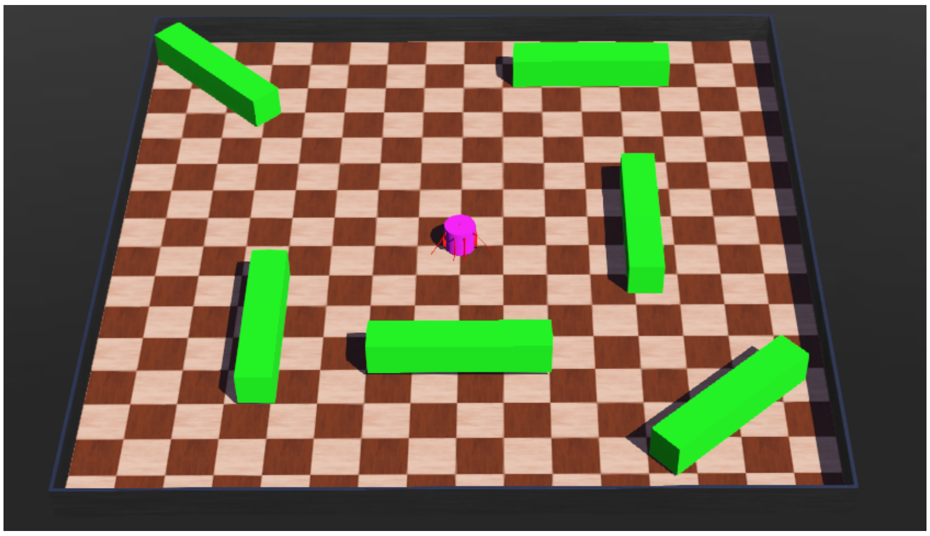


Figure 4: Webots environment.

2.3 Neural Network

The experimentation involved the implementation of a robot controller based on a feed-forward neural network (FNN), optimized using a GA. The FNN served as the core decision-making component, leveraging its ability to model complex relations between motors and sensors mappings and hierarchical representations of the environment. By incorporating multiple hidden layers, the FNN extracted meaningful features from sensor inputs, enabling the robot to navigate and explore effectively. The use of a GA for parameter optimization enhanced the FNN's performance by iteratively evolving the neural network's weights and biases, thus improving obstacle avoidance and exploration capabilities.

The related work experimental results have demonstrated the efficacy of the proposed approach in real-world scenarios, showcasing the robot's ability to avoid obstacles and explore unknown areas efficiently [10] [11]. The FNN robustness and generalization capabilities are features that may enable the robot to navigate complex environments with minimal human intervention.

2.4 Genetic algorithm

The genetic algorithm is based on Charles Darwin's natural selection theory. Darwin says *"if variations useful to any organic being do occur, assuredly individuals thus characterized will have the best chance of being preserved in the struggle for life; and from the strong principle of inheritance, they will tend to produce offspring similarly characterized. This principle of preservation, I have called, for the sake of brevity, Natural Selection"* [12].

The fundamentals of a genetic algorithm according to [13] are **population**, **individual**, **fitness** and **selection**.

- **Population:** Collection of candidate solutions that are considered during the course of the algorithm. Over the generations of the algorithm, new members are *"born"* into the population, while others *"die"* out of the population.

- **Individual:** Single solution in the population.
- **Fitness:** Measure of how *"good"* the solution is represented by the individual. Usually, the higher the fitness, the better - Depending on the problem to be solved.
- **Selection:** The survival of the fittest individual. Individuals that are selected for *"breeding"* – That is where the **crossover** takes place, based on the parent's fitness values where a high fitness makes more likely an individual to reproduce. Moreover, during each generation, there is a small probability for each individual to **mutate**, which changes the individual in some small way.

The genetic algorithm pseudo-code implemented based on [13] and [14] is shown below:

1. Create a **population** of random candidate solutions, number of genes, and number of generations.
2. Calculate **Fitness** for the initial population.
3. Until the algorithm termination conditions are met, do the following (each iteration is a generation):
   (a) **Select** parents proportional to **fintess**.
      i. Parents selected by rank selection.
   (b) **Crossover** parents to create children.
   (c) **Mutate** children.
   (d) Calculate fitness of the children.
   (e) **Add** children to new population.
4. Replace original population with new population.
5. Select the individual from new population with the highest fitness as the solution to the problem.

2.5 Fitness function

The fitness function used was the count of squares that the robot has visited. The grid of the environment was created, and the robot trajectory stored. The grid of the map is a list of lists where the inner list is made of instances of a class called Square. Then, for each coordinate in the robot trajectory a function is evaluated. The function changes an attribute of the square to true if the robot has been at any point in that square area. Finally, the parameters set with the highest count of explored squares is the best solution equivalent to be the solution with best fitness.

3 Results

3.1 Fixed and not fixed bias comparison - Mutation rate 0.1 - Generations 15

The initial results of the experiment evidenced a slight superiority of the algorithm that mutated all the parameters. As the one with fixed bias explored in average four less squares. Tables 1 and 2 show the top three parameters set and its fitness.

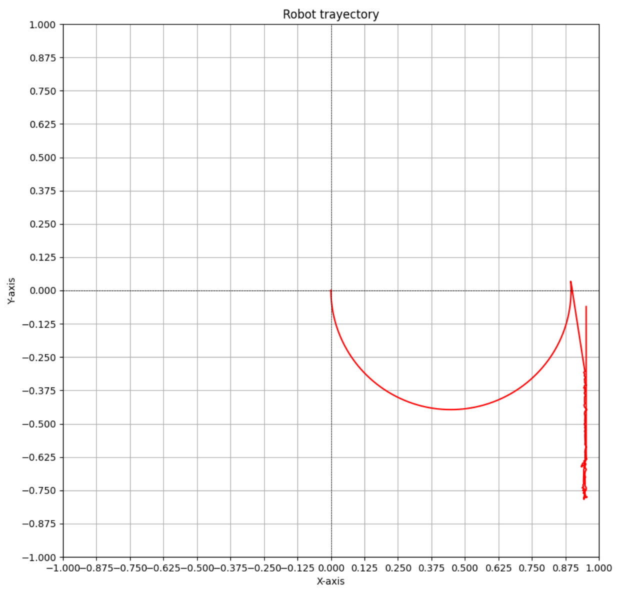
Top 3	Best Fitness	Avg Fitness
1	31	22
2	30	17
3	29	18

Table 1: Best three fitness values for no fixed bias, mutation rate 0,1 and 15 generations.

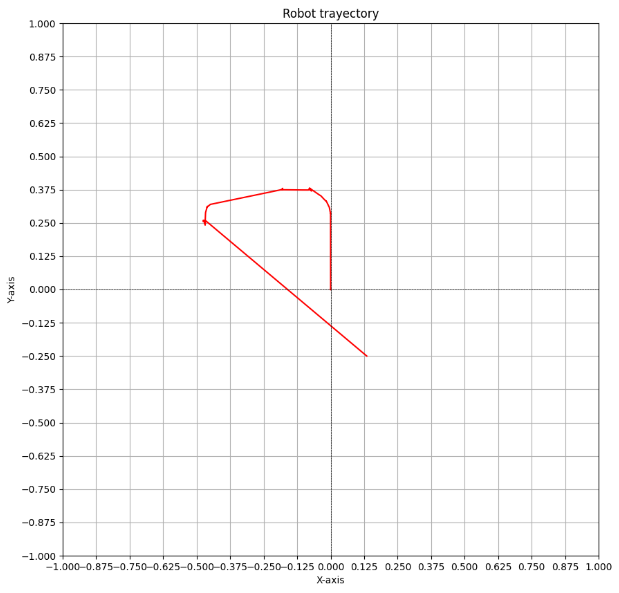
Top 3	Best Fitness	Avg Fitness
1	25	19,5
2	24	16,5
3	23	14

Table 2: Best three fitness values for fixed bias 2, mutation rate 0,1 and 15 generations.

The robot trajectory for the best solutions is shown in figure 5. In this case, not using a fixed bias is the best option to explore more of the map with a difference of 6 more explored squares than the solution with a fixed bias. However, as the previous results are using just 15 generations is necessary to test the methodology robustness with more generations and varying the mutation rate.



(a) Robot trajectory for the best solution with no fixed bias, mutation rate 0,1 and 15 generations.



(b) Robot trajectory for the best solution with fixed bias, mutation rate 0,1 and 15 generations.

Figure 5: Robot trajectory comparison for fixed and not fixed bias - mutation rate 0,1 - 15 generations.

3.2 Fixed and not fixed bias comparison - Mutation rate 0.1 - Generations 100

The experimentation revealed that is still a better option to evolve all parameters rather fixed the bias. Even if the best value found by the optimization algorithm is almost the same, is important to check the average fitness. Meaning that the first option found more optimal parameters set in general than the fixed bias option.

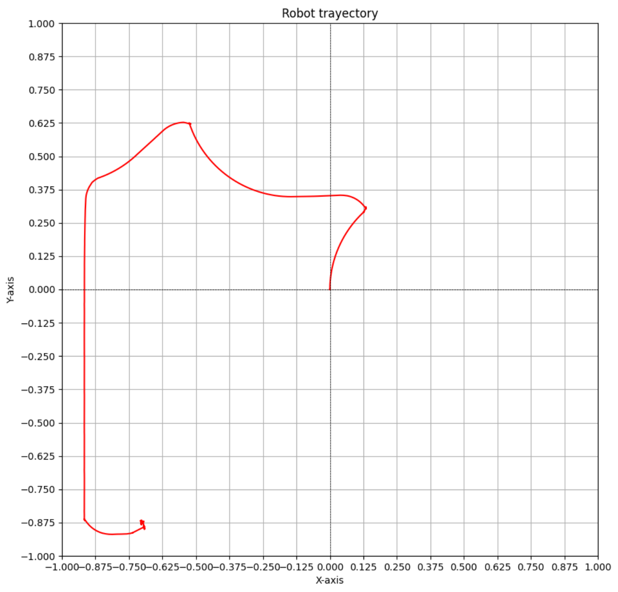
Top 3	Best Fitness	Avg Fitness
1	34	28.5
2	32	25.5
3	31	27

Table 3: Best three fitness values for no fixed bias, mutation rate 0,1 and 100 generations.

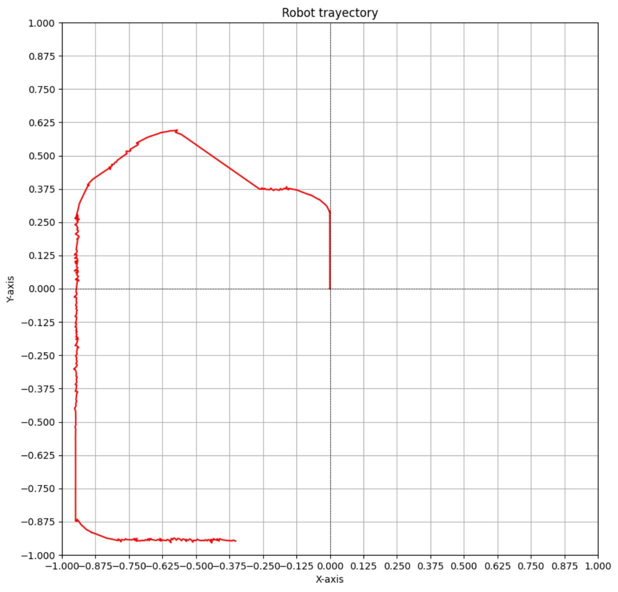
Top 3	Best Fitness	Avg Fitness
1	33	14
2	19	16
3	18	16

Table 4: Best three fitness values for fixed bias 2, mutation rate 0,1 and 100 generations.

As done before, the robot trajectory for the best solutions is shown in figure 6. The trajectories are similar. Nonetheless, as mentioned before the no fixed bias strategy display better results. Moreover, the first trajectory reveal that the robot explored the area without much interference. However, the second trajectory indicate that the robot could collide with the wall or stayed too close to the object in its exploration. Therefore, in real life, this situation could be dangerous for the robot and the task.



(a) Robot trajectory for the best solution with no fixed bias, mutation rate 0,1 and 100 generations.



(b) Robot trajectory for the best solution with fixed bias, mutation rate 0,1 and 100 generations.

Figure 6: Robot trajectory comparison for fixed and not fixed bias - mutation rate 0,1 - 100 generations.

3.3 Fixed and not fixed bias comparison - Mutation rate 0.2 - Generations 100

Finally, as proposed in the methodology a slightly higher mutation rate is tested. Providing more possibility to explore the solution space better, without varying the population abundantly and keep the positive characteristics acquired throughout generations.

The behaviour is similar to the previous results. Not using fixed bias seems to be the best option to evolve the NN parameters. The difference in the best solution fitness and the average fitness is considerable. Over 9 more explored squares in best fitness comparison and 6 more explored squares in average. Table 5 and 6 show the top three best solutions and average fitness throughout generations.

Top 3	Best Fitness	Avg Fitness
1	39	28
2	38	28.5
3	35	26.5

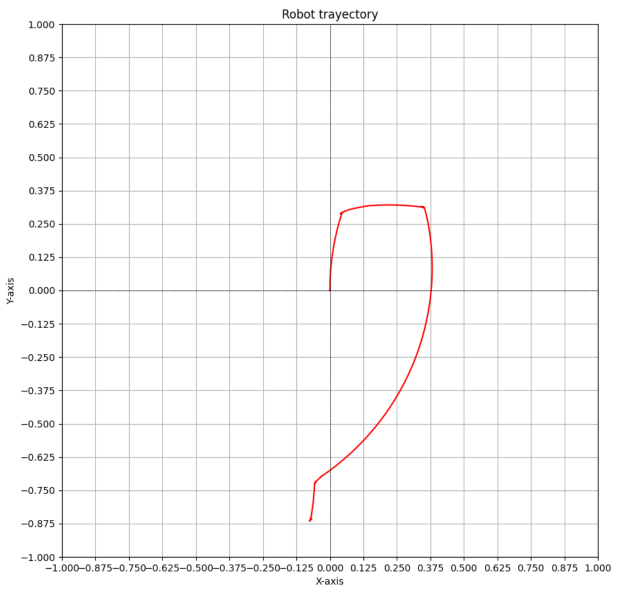
Table 5: Best three fitness values for no fixed bias, mutation rate 0,2 and 100 generations.

Top 3	Best Fitness	Avg Fitness
1	30	16.5
2	28	22.5
3	27	23

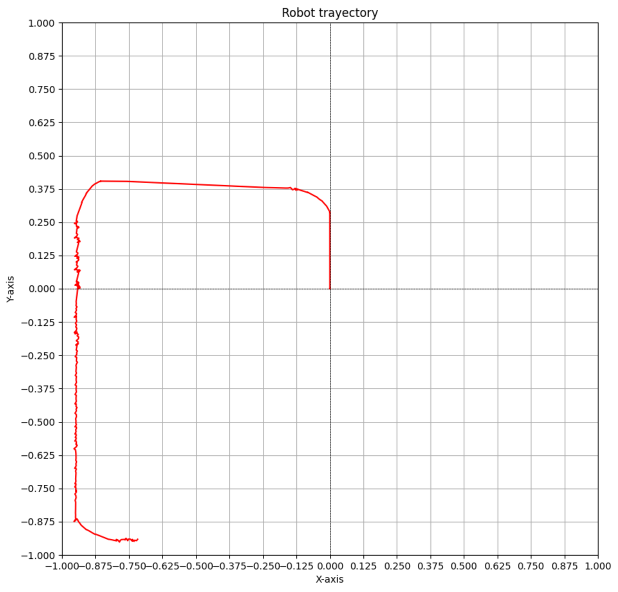
Table 6: Best three fitness values for fixed bias 2, mutation rate 0,2 and 100 generations.

The robot trajectory for the best solutions is shown in figure 7. Once again, the first option is superior. The move of the robot is even and straightforward avoiding the obstacles. On the other hand, the second option with fixed bias evidence a more labored trajectory.

Furthermore, use a higher mutation probability was helpfully to increase the performance of both options. Over 4 more maximum explored squares and around the same average fitness over the generations.



(a) Robot trajectory for the best solution with no fixed bias, mutation rate 0,2 and 100 generations.



(b) Robot trajectory for the best solution with fixed bias, mutation rate 0,2 and 100 generations.

Figure 7: Robot trajectory comparison for fixed and not fixed bias - mutation rate 0,2 - 100 generations.

4 Discussion

Autonomous exploration in unmanned vehicles is a complex task. The high number of variables in the robot, complex and changeable environment conditions, possible failure of any asset of the robot, and more. Are key factors that prove the complexity of any autonomous and intelligent task of a robot or unmanned vehicles. That is why the objective of this experiment was to test a feed forward neural network evolution that works as a controller of a simple robot with four front sensors. The approach studied was to evolve the neural network parameters using a genetic algorithm. Also, varying the generation number and the mutation rate to allow a deeper understanding of the solution space.

The results of the first experiment with just 15 generations showed from the beginning that the best option to explore the environment the most is to evolve all the neural network parameters. The difference at this point is not massive. Furthermore, the genetic algorithm evolves the population keeping the individuals with the highest fitness, hence, the characteristics that lead to better exploration in the map are kept and the solutions are better with each generation. However, if the initial randomly generated population does not have at least one individual that stands out with high fitness, is necessary to add more generations and increase the mutation rate.

When the generations were increased to 100 the fitness increased. There is not a big change in fitness. However, the results confirm that is the right path to keep improving the model. Moreover, the average fitness over time is also higher. Meaning that the algorithm is finding the best individuals and creating new ones that are effective in the task. Also, increasing the mutation rate is helpful, as using these characteristics in the genetic algorithm led to explore almost 40 squares, that was the highest number of squares explored in all experiment. Furthermore, is important to highlight that evolving all the parameters seems to be the best option. The robot trajectory evidence that the robot avoids obstacles better and moves safely using this technique rather than fixing bias.

Likewise, is key to underline that in some cases, the robot moved backwards when all the parameters were evolved. In this case, that is not a sensible option as the robot has sensor in the front and move backward may lead to collisions and damages to the robot so adding more sensors to the robot may help to avoid obstacles better and improve exploration. Finally, due to the complexity of the environment, the limitations of the feed forward neural network and the robot as is equipped with just 4 distance sensors, some of the best solutions were when the robot kept doing circles around an object what is not the best output when the objective is to explore the whole map. Therefore, implementing activation functions and backward propagation to the neural network is recommended. Also, add more sensors to the robot, at least two rear sensor to avoid collisions when going backwards. Additionally, add a function that identifies loops and finds a way to avoid them.

5 Future work

Moving backwards in this case is not what is expected as an optimal solution, that is why fixing bias strategy was carried out. Hence, is recommended to add more sensors to the robot, at least one in the back may help to avoid obstacles better and explore more of the map. Also, the neural network used has some limitations, therefore, adding an activation function and backward propagation is necessary to improve the exploration in complex environments. Moreover, using more generations in the genetic algorithm and a maximum of 25% mutation rate may help to obtain better results.

References

[1] S. Shoval, I. Ulrich, and J. Borenstein, “Robotics-based obstacle avoidance systems fot the blind visually impared,” *IEEE Robotics and Automation Magazine*, 2000. [Online]. Available: `https://www.researchgate.net/publication/2854750_Invited_article_for_the_IEEE_Robotics_and_Automation_Magazine_Special_Issue_on_Robotics_in_Bio-Engineering_Last_revised_August_27_2000`.

[2] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, “A depth space approach to human-robot collision avoidance,” in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 338–345. DOI: `10.1109/ICRA.2012.6225245`.

[3] T. Petrič, A. Gams, N. Likar, and L. Žlajpah, “Obstacle avoidance with industrial robots,” in *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, G. Carbone and F. Gomez-Bravo, Eds. Cham: Springer International Publishing, 2015, pp. 113–145, ISBN: 978-3-319-14705-5. DOI: `10.1007/978-3-319-14705-5_5`. [Online]. Available: `https://doi.org/10.1007/978-3-319-14705-5_5`.

[4] H. Rezaee and F. Abdollahi, “A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots,” *IEEE Transactions on Industrial Electronics*, vol. 61, no. 1, pp. 347–354, 2014. DOI: `10.1109/TIE.2013.2245612`.

[5] D. Chen, “Fuzzy obstacle avoidance optimization of soccer robot based on an improved genetic algorithm,” *J Ambient Intell Human Comput*, vol. 11, pp. 6187–6198, 2020. DOI: `https://doi.org/10.1007/s12652-019-01636-0`. [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/8897013`.

[6] F. Leung, H. Lam, S. Ling, and P. Tam, “Tuning of the structure and parameters of a neural network using an improved genetic algorithm,” *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 79–88, 2003. DOI: `10.1109/TNN.2002.804317`.

[7] S.-S. Han and G. May, “Optimization of neural network structure and learning parameters using genetic algorithms,” in *Proceedings Eighth IEEE International Conference on Tools with Artificial Intelligence*, 1996, pp. 200–206. DOI: `10.1109/TAI.1996.560452`.

[8] M. Lin, J. Xiaoming, and Q. Fei, “A robot obstacle avoidance method based on improved genetic algorithm,” in *2018 11th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, 2018, pp. 327–331. DOI: `10.1109/ICICTA.2018.00081`.

[9] C. Ltd. “Cyberbotics.” (2024), [Online]. Available: `https://cyberbotics.com/#webots`.

[10] J. A. Fernandez-Leon, G. G. Acosta, and M. A. Mayosky, “Behavioral control through evolutionary neurocontrollers for autonomous mobile robot navigation,” *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 411–419, 2009, ISSN: 0921-8890. DOI: `https://doi.org/10.1016/j.robot.2008.06.012`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0921889008000936`.

[11] J. A. F. León1, O. E. Goñi, G. G. Acosta, and M. A. Mayosky, “Neuro-controllers, scalability and adaptation,” pp. 1279–1288, 2004. [Online]. Available: `https://d1wqtxts1xzle7.cloudfront.net/70869659/Documento_completo-libre.pdf?1635668135=&response-content-disposition=inline%3B+filename%3DNeuro_Controllers_scalability_and_adapta.pdf&Expires=1712891479&Signature=P7kEgmND1qyOik2q9ENWu38RY3UfGv-9v1czEEg8f-49F~KQxLf50QB23be8FY5o6Sm7sMchkE6E4wmrEBbjpgvP4~l6S~q5jKJrqQm4FHdnrCmT72sfPkSDs4-eBYoid6O2WJrnikGwIPPGEBSUP1geMK4TrVvw8s1BdD2KlmjMh7hhbJEqnaC7~jRdZDNXgulNEK7NtRxNu5H7QyJHagJ9I pJQBmTymjuTbEBb0aHF4iSze44-i3VbgVPdX-BayciI0CcdKDjkHASzq7eYUAw9VEEheBd0qjEa1srj0Cr0b1KVMdXQeY _&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA`.

[12] C. Darwin, *On the origin of species in from so simple a beginning: the four great books of Charles Darwin. edited, with introductions, by Edward O. Wilson. W. W. Norton and Company.* 2006.

[13] S. M. Thede, “An introduction to genetic algorithms,” *DePauw University*, 2004. [Online]. Available: `https://www.researchgate.net/profile/Scott-Thede/publication/228609251_An_introduction_to_genetic_algorithms/links/00b7d52cdb71f1e8bf000000/An-introduction-to-genetic-algorithms.pdf`.

[14] C. Johnson, “Evolution as algorithm,” *University of Sussex*, 2023.