Acquired Intelligence and Adoptive Behaviour report

Title: Performance Comparison of Car Robots Trained with Spatial and Microbial Genetic Algorithms in Real-World and Simulated Environments across Varying Generational Settings

Student name: Keisuke Tsujie

Candidate number: 263065

Table of contents

0 Abstract 3

1 Introduction 3

1.1 Genetic algorithms (GA) 3

1.2 Spatial GA and microbial GA 3

1.3 The agent and the environment applied in the practice 4

1.4 Hypothesis 4

2 Methodology explanation 4

2.1 Overview 4

2.2 Simulation environment 5

2.2.1 Brief of the experiment in simulation 5

2.2.2 Environment setup. 5

2.2.2 Setup Genetic Algorithms 6

2.2.3 Genotype-phenotype encoding 7

2.2.4 Run the experiment 7

2.3 Real-world experiment method explanation 7

2.3.1Equipment 7

2.3.2 Hardware specifications of the robot 8

2.3.3 Software specifications 8

2.3.4 Environment conditions 8

2.3.5 Setting up the development environment 9

2.3.6 Run experiment 9

3 Result evaluation 9

3.1 Simulation environment 9

3.1.1 Quantitative comparisons 9

3.1.2 Qualitative comparisons 11

3.2 Real-World environment 12

3.2.1 Quantitative comparisons 12

3.2.2 Qualitative comparisons 12

4 Conclusion and outlook 13

4.1 Conclusion 13

4.2 Outlook 13

5 appendix 14

0 Abstract

This research involves performance comparison of a vehicle agent trained in two different genetic algorithm approaches: spatial genetic algorithm and microbial genetic algorithm. The desired action of the agent is to detect and follow light. The comparison is done in simulation and real-world. The investigation also monitors the change in performance across the different genetic algorithms. Spatial genetic algorithm is expected to provide more robust and higher performing model in the later generation. Contrary, microbial genetic algorithm is expected to output reasonably well performing model in the early generation. In the simulation, the result depicts the attributes of both genetic algorithms as it was expected. On the other hand, the real-world examination ends up with failure that the most model simply does circular motion in the clockwise direction. In the document, it refers to the detailed methodology explanation and the result evaluation and the background concept.

1 Introduction

This paper refers to an investigation about comparison of performance of car robot which is trained in two genetic algorithm (GA) approaches: Spatial GA and Microbial GA. The comparison is done in a real-world environment and a corresponding simulation environment. The experiment also compares the performance of each method in different generation settings for each approach.

1.1 Genetic algorithms (GA)

Genetic algorithms are search methods aimed at finding the most optimized solutions for various task [Manoj Kumar, 2010]. It imitates the evolutional process the world of nature. Single or multiple agent(s) are optimized to a given task throughout the process of parameter – so called genotype – inheritance, mutation, selection and cross over between each generation. The criterion of the selection is so called fitness which is calculated from different aspects to analyse the performance of agents. Therefore, creating the individuals which has high fitness score tends to be a principle of training in the GA training. In this report, we focus on two GA approaches which are spatial GA and microbial GA.

1.2 Spatial GA and microbial GA

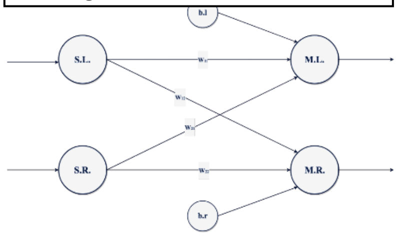
The notable point of spatial GA is that the individuals are mapped onto a spatial structure such as grid. Using the structure, it makes a limitation to interactions between individuals so that they can only interact with their spatial neighbourhoods. This makes it possible to implement parallelly and maintain the diversity for their search. It contributes to generate a robust genotype in the long term [Andrzej Broda, 1996]. On the other hand, microbial GA is inspired from bacterial conjugation and microbial evolution including tournament selection [Kagioulis, 2021]. Each individual is required to compete against one of the other individuals. The genotype of the individual which lost in the competition is overwritten by that of the winner with some information mutation. This tournament like style of genotype improvement achieves simplicity on implementation and outputting genotypes which reasonably behave well in early generations [Umer, 2015]. To sum up with, spatial GA has strength on producing robust genotypes in more generation, and microbial GA has advantages in rapid adaptation. Therefore, it would be meaningful to take a track of their performance throughout various generations settings and various environments.

1.3 The agent and the environment applied in the practice

Figure: the agent

An abstract simulation environment and the real-world environment is applied in practice. Both environments handle the same agent which is a car robot. It consists of two wheels on each side of the vehicle, a ball caster placed under the body, and two light detecting sensors set at the front of the vehicle [School of Engineering and Informatics, 2024]. As a parameter to control robots, it contains a neural network consists with four weights administrate the implication of each sensor to each wheel, and two biases which influence the final wheel movement. For example, the motion of the left wheel is determined by following formula: $lm = x \times w_{ll} + y \times w_{rl} + b$. "$x$" and "$y$" are the light intensity while "w_{rl}" and "w_{ll}" are the weight to manage the implication of the left sensor to the left motor and the implication of the right sensor to the left motor. The bias "b" is added to the result of calculation. The goal of the investigation is to train the parameters as a genotype and optimize the motion of the agent to follow the light source. The agents are trained within the simulation and the result of training is applied to the real-world. Predetermined number of agents are trained separately and are compared in terms of their fitness in each generation. The further specification about the agent and the environment are stated in the methodology explanation section.

Figure: The neural network



1.4 Hypothesis

In terms of the growth speed of the agent in the training process, microbial GA would give the better result. As microbial GA has advantage with its simplicity and rapid improvement of agent`s performance, this method is expected to provide individuals which has reasonably high performance in the early generation. On the other hand, because spatial GA gives limitations to the comparison and selection of agents, it could be possible for it to produce an individual that has relatively low performance in the early stages of the training process. However, as the number of generations allowed increases, the performance of the spatial GA will improve, and possibly overtake the agent trained by microbial GA in the later generation. The robust ness of the genotype provided from spatial GA training would positively affect the behavior of the robot in the real world.

2 Methodology explanation

2.1 Overview

There are two steps in this experiment. The initial step is to train the genotype for automobile vehicle motion control and performance monitoring under simulation. Various GA techniques are used to train the genotypes. The first step is Setting up the simulation environment in which the experiment is to be conducted is. Next, set up each GA technique. Lastly, use the environment and the developed strategy to train the agent. Comparisons of the agent's performance in different generation circumstances are another feature of the training. As a result of the experiment, the trajectory of the auto robot with the best genotypes and the trend of change in fitness function during training in each example are monitored. The genotypes learned in the simulation are applied to the second stage of the experiment. The environment in which the robot operates mimics the scene from the simulation. The distance that remains between the robot and the light source after the runtime has been established is noted as the outcome. The experiment is conducted three times for each scenario, and the mean of the resulting distances is calculated. The resultant distance between the robot and the light source and the trajectory of the vehicle passed are measured as independent variables. For the dependent variables, the generation allowed for training and the genetic algorithms methods are changed to see the difference in the result of independent variables.

2.2 Simulation environment

Configuring the simulation environment supports the main goal in two ways. Making a comparison between the simulation and actual testing is the first step. Given that comparing how well each GA technique performs in diverse circumstances and settings is the experiment's main goal. It is valuable to compare simulation with in-person testing. This is due to the fact that the simulation allows for complete parameter control, creating the ideal environment. This characteristic creates a noticeable contrast with the noisy real environment. Its effectiveness in training agents is the second feature. Its effectiveness in training agents is the second feature. Before moving on to real-world experiments, which need more resources, they provide efficient initial training and testing as well as iterative development since they enable rapid experimentation under completely controlled conditions. All code for simulation is run on Google Collaboratory.

2.2.1 Brief of the experiment in simulation

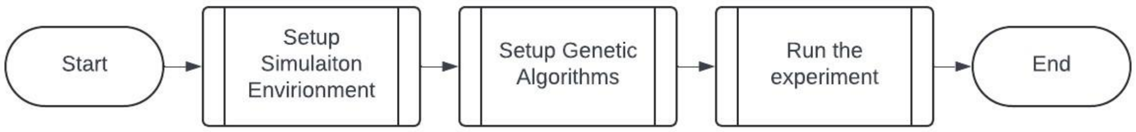


Figure: The flowchart about the process run in the simulation environment

Here is a flowchart shows overview of the procedure of the experiment in the simulation environment. It begins with setting up the simulation environment. Then, each GA method is initialized, and the experiment is run as the last step.

2.2.2 Environment setup.

The implementation for setting up the simulation consists with agent initialization and environment configuration. This is implementation is an application of the Lab 3 resource from the module. The setup of agent is done by the class “Braitenberg” This involves the following attributes which holds the genotype of the agent and the initial bearing and the initial position of the agent in the experiment. It contains a method to return the genotype to pass the information to the environment and evolutionary algorithms.

A class “Environment” oversees configuring the simulation. The simulation is segmented by time step of 0.05 seconds to refresh the simulation. The total running time is 5 seconds for one set, so the simulation is segmented in 10000 steps. The radius of the physical size of the machine is unchanged from the setting in the lab resource which is 0.05. To mimic the real-world simulation, gaussian noise is added on the sensor and the position information. The level of noise will change by the input value of standard deviation. Since the circumstance in the real world is not noisy, they keep the low value that the positional noise has 0.05 and the sensor noise for 0.1. It also determines the location of the light and the robot at the starting point [Jakobi, Husbands and Harvey, 1995].

This class also contains just one method “run()” which runs the experiment involves simulating the movement of the agent. This simulation process includes computing sensor inputs, updating positions of the car, and applying noise to the final position of the car and the sensor input. The light intensity received by each sensor is calculated by the gain of the sensor divided by the distance to the source. The movement of the motor is calculated by the formula mentioned in the past. Finally, the change in position of the robot is applied by summation between the previous location and the new motion happened in the time step. The motion is calculated by:$(speed\ of\ the\ agent) \times (time\ step) \times (\cos(x)\ or\ \sin(x))$. X is the machine`s bearing. The calculation for vertical move, sine is applied, and cosine applied for the horizontal move.

The simulation iterates loops over the simulation time segmented by the time step. Throughout each iteration, the position and the bearing of the vehicle are updated based on sensor input and motor output. Let us break down the sequence of the iteration.

2.2.2 Setup Genetic Algorithms

The GA algorithm setup is significant for evolutional training of the agent under simulation. This process involves three classes: EvolutionalAlgorithm, SpatialGA, FullMicrobialGA. EvolutionalAlgorithm class contains the common functionality of GA. This could be utilized by various GA approaches. It contributes to ease of adding a new approach by using the utilities from this class. This assume that the scope of the investigation is extended to see the performance of other GA techniques. The class SpatialGA and FullMicrobialGA focus on implementing each gene selection. They inherit EvolutionalAlgorithm class.

To begin with, let us look at the attributes of the class. This class in charge of, initializing the population ang genotype, mutation, fitness calculation management, running the genetic algorithms. The generation is initialized in a list. The population is initialized in an integer. Increasing these parameters causes trade-off between the performance of agents and the time consumption [Tsoy, 2003].. The genotype is set as 50, 150, 300, and the population as 100. These parameters are set in the maximum value allowed within the time limitation to adjust other hyperparameters. It takes seven and half minutes in average. The genotypes are determined randomly within the range of predetermined maximum value and minimum value which is 5 and 0 in the practice. By limiting the range of values in the genotype, stable behaviour is expected from the agents. The mutation is added to preserve the diversity of genotype. It is implemented by adding gaussian noise to genotypes. The mean as 0 and setting the standard deviation as 0.1 to be used in the gaussian noise. The value of mutation is not set as too high value as it could causes changes in well performing genotypes [Vose, m., 1994]. The fitness is calculated by $\frac{Change\ in\ distance\ between\ the\ agent\ and\ the\ light\ source}{the\ original\ distance\ between\ the\ agent\ and\ the\ light\ source}$. It could be better in analysing the performance of the agents with the length of the trajectory and the average bearing they were looking at. However, those functionalities are not implemented due to the time limitation. The fitness values of each agent in a generation is sorted in list form to make an evaluation in the later stage. The stated functionalities are implemented in a loop which manages the genotypes and fitness and apply the evolutional strategies from the information across generations.

FullMicrobialGA class simply implement the evolutional strategy of microbial GA. It firstly applies the elitism. This means that the predetermined number of well performing genotypes are remained for the next generation without mutations. 10 elite – the top 10% of the population – genotypes are passed to the next generation in this experiment. This value takes a balance of achieving diversity of the genotype and protection of the strong genotypes

[Bhatia and Basu, 2006.]. After the process is done, a while loop starts to determine the remaining individuals. Two individuals are selected randomly from the remaining genotypes, and their fitness is compared. The individual with fitter genotype is mutated by the method mutation function inherited from EvolutionalAlgorithm and added into the new population list. This loop continues until the new population list becomes full. As a result, a new population is created.

SpatialGA class is developed to put the spatial GA's changing strategy into practice. It firstly allocates x and y coordinates to each genotype within a grid-based population structured framework. This spatial structure is aimed to have low complexity to run each trial fast so that it could spend shorter time for each trial and find the best hyperparameter settings for the experiment. Then the three-by-three grid cantered on the current individual from the initial grid is then segmented using a for loop. Next, every element in the divided grid is added to a list known as "neighbors." Once the iterations are finished, it chooses each individual’s best neighbor and passes it to the new population list. If the newly added individual has lower fitness than a criterion of fitness, it is mutated before being added. Finally, the list of new population variable is returned.

2.2.3 Genotype-phenotype encoding

Direct encoding method is used for genotype-phenotype encoding. This simply pass the value of genotype to the phenotype [Chow, 2024]. This is due to the dependence between each motor and each sensor is high when it comes to determining the overall motion, the simplicity in encoding is required. For example, the genotype segmentation is done with the sensors and the motors as encoding. It cuts the coordination between the motors and the sensor which gives significant problem on the motion of the vehicle after all.

2.2.4 Run the experiment

This section is made to run the experiment and to collect and visualize the result. Throughout a for loop to run training with various generation settings, it runs the entire process of the experiment such as simulation of the agents, the training calculation involves the fitness calculation and both genetic algorithms application, recording the genotypes and fitness. Once the experiment is done, it plots the trend of fitness and the trajectory of the outputted agent`s motion under the experimental environment. It also saves the best genotype in the final generation with the best fitness value in txt format.

2.3 Real-world experiment method explanation

2.3.1Equipment

Figure: Equipment used in the experiment

Some equipment is used in the experiment. The robot as an agent in the real world. A measure and timer to check the any physical distance and the time. A role of tape to segment the area for experiment and to mark the resultant location of the robot. An A flash light (iPhone SE 3 flash light) is utilized as the target for the robot. A set of genotypes are needed to be applied to the robot throughout the experiment. Finally, a laptop and a cable are required to communicate with the robot and to apply the code.

2.3.2 Hardware specifications of the robot

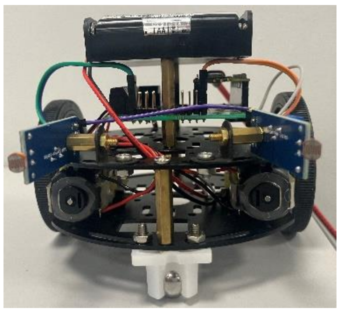


Figure: picture of the robot

The robot is a wheel bot which equipped with two DC motors and a ball caster at the from [shepai, 2024]. It has a light-sensitive sensor which read analogue input. The input is interpreted as digital 0-1through a pin on the front edge of the sensor. The machine has two board: a Raspberry Pi Pico for applying the code which is run by MicroPython and Cytron motor controller for motor control administrated by CircuitPython which is located under Raspberry Pi Pico Board. On the Raspberry Pi board, a reset button is located to flush the state of the machine. The on/off switch is located on the side of the Cytron motor controller.

2.3.3 Software specifications

To apply codes in the machine, a custom robotics library is used. It is compatible with both CircuitPython and MicroPython. They are edited and run by IDE called Thonny. Two files code.py and EduBot_CP.py is given to interact with the robot. The main control script is written in code.py. The motor sensitivity and the sensor gain could be modified in the script to determine the basic behaviour of the robot. The motor sensitivity is set as 0.18 which is the least value that the robot can stably accelerate and decelerate. The sensor gain is set as 1 since the light source has low intensity. Note that at the end of the calculation stage, the output speed is multiplied by 0.85 for the right motor. These settings are done to make the robot to run straight with a genotype which has 2.5 – mean of the maximum value and the minimum value – for all value. EduBot_CP.py module provides the wheelBot class and methods for controlling the motors and sensors of robot and some utilities such as halting all motor activity and software reset operation. It also contains the script to configures the robot`s hardware based on the robot type.

2.3.4 Environment conditions

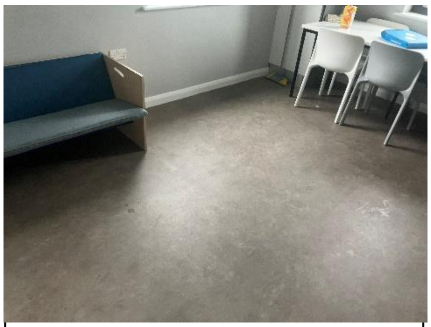


Figure: room used for experiment

The Dimension of the area used for the trial is the first factor as the robot requires reasonably wide area to move freely. This time, the area is set with 0.55x0.55 (m) square and 0.5m margin on each side. The surface condition and the existence of obstacles are also important since they could influence the motion of the vehicle. The flooring of the room is flat without any tilting and having plenty friction for the robot to move. There is no obstacles or bumps in the area. In addition, the darkness of the testing station is a significant element in order to guarantee stable performance of the light sensors. The experiment is done in the nighttime which provides constant darkness that the sensor did not detect any light without light source. The light source and the robot is initially located on polar opposite side of the area with confirmation of the robot detects the light with the setting. The robot initially facing towards the light source as the intensity of the flashlight is low.

2.3.5 Setting up the development environment

In order to setup the machine, the first step to be taken is to flush the board by pressing the reset button. Then, search up a utility "Adafruit-circuitpython-raspberry_pi_pico-ja-9.0.4.uf2" on searching engines and downloading it. After getting it, connect the robot with the PC and add select "Adafruit_motor" file and "Simpleio.mpy" file from the utility to add them into "lib" folder of the machine file. Finally, add files "code.py" and "EduBot_CP" into the machine file. Once completing all the sequence above, the PC would be interactable with the machine through out an IDE called Thonny. To apply the code to the machine, the first step is to confirm the machine is switched off. Then, connect it with the PC. Once it`s connected, configure the interpreter "CircuitPython(genetic)・CircuitPython CDC control @... " from the bottom right banner in Thonny UI. After the interpreter gets ready, open "code.py" and "EduBot_CP" file in the IDE and apply the settings such as genotypes. Finally, the move to the "code.py" tab and run the code from the "Run current script". This command is included in the run section located in the top left banner in the IDE.

2.3.6 Run experiment

There are three trials given for the trial. In each run, the timer starts to count runtime 5 seconds which is same setting as the experiment in the simulation environment. After the runtime is over, the robot is captured, and the final location is marked. Then, the distance between the location and the light source is measured. Once the third set of the trial is done, the genotype setting is changed.

3 Result evaluation

The result evaluation involves qualitative and quantitative comparisons of the result from the investigations in the simulation and the real-world

3.1 Simulation environment

3.1.1 Quantitative comparisons

One of the elements to be pointed out at this point is the fitness score in each situation. The two aspect, the peak and the trend of the score are going to be analyzed.

	50 (generation)	150	300
Spatial GA	0.403	0.897	0.7614
Microbial GA	0.837	0.8715	0.902

Table: The best fitness score in each case is the first component to evaluate.

The fitness score is derived from $\frac{Change\ in\ distance\ between\ the\ agent\ and\ the\ light\ source}{the\ original\ distance\ between\ the\ agent\ and\ the\ light\ source}$.

A fitness score of 0.5, for instance, indicates that half of the initial distance between the agent and the light source is lost. When the generation limit was 50, the spatial GA result showed a noticeably lower score. The agent with 150 generations of training has a score that is more than twice as high as the agent with 50 generations of training. Although the agent trained for 300 generations performed worse than the agent that evolved over 150 generations, it was still better than the agent that evolved over 50 generations. However, in the 50 generations of training, the microbial GA method received a score of 0.837. Compared to the 300-generation case in the spatial GA approach, the value is even better. After the agent underwent 300 generations of evolution, the fitness finally reached 0.902, with the score progressively improving as the number of generations grew. It could be pointed out that the simplicity of the structure of the neural network of the car control variable helps microbial GA to produce high performance.

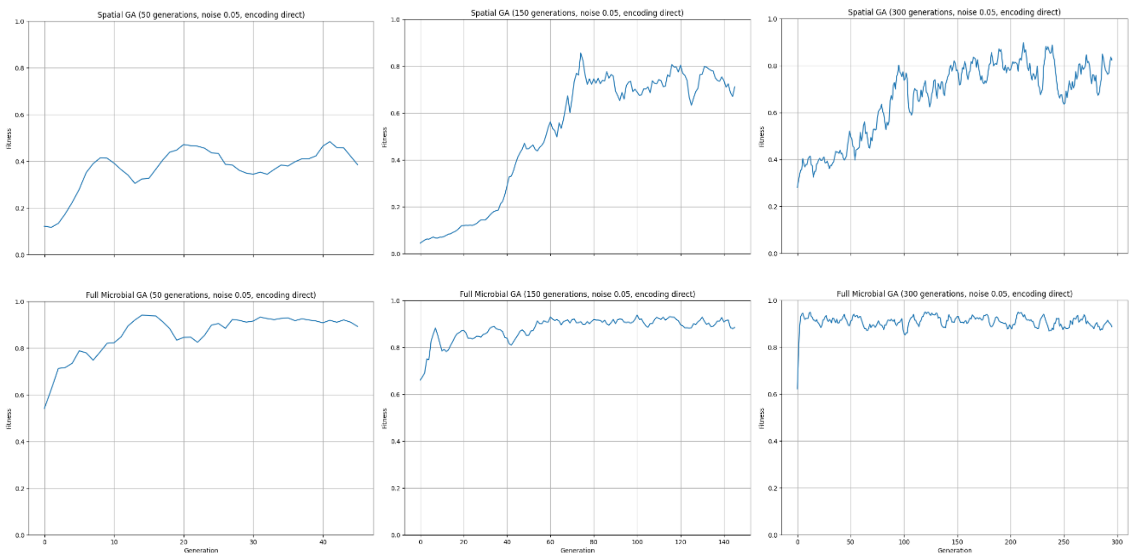


Figure: Fitness trend over generation. From the left to right: 50, 150, 300 generations

The spatial GA's fitness trend is shown in the graphs' upper section. Throughout the experiment, the fitness score increased steadily and progressively in each case. The fitness was initially less than 0.3 in each graph. When it comes to the 50 generation, the graph peaks at generations 10, 20, and 40, with values of about 0.4. In the case of 150 generations, the score increased until it reached 0.8 for 100 generations. Following that instant, the trial's progress toward improving the score stalled. Up until generation 200, the practice with 300 generations saw steady improvement. Be aware that to determine the true site of convergence, it might be beneficial to conduct spatial GA training with a bigger generation. Microbial GA method experiences convergence at about 10 generations which is relatively early. After reaching the peak value range of 0.8 to 1, it ceases to expand.

The Microbial GA method converges at an early stage after producing remarkably high performance at an early point. In contrast, the spatial GA technique demonstrates long-term steady, moderate growth. The outcome unequivocally demonstrates that part of the convergence speed theory is true. It is unexpected that spatial GA was unable to surpass microbiological GA at any point in the fitness score. This could happen because the spatial GA-trained agent was not yet fully optimized within the generation range.

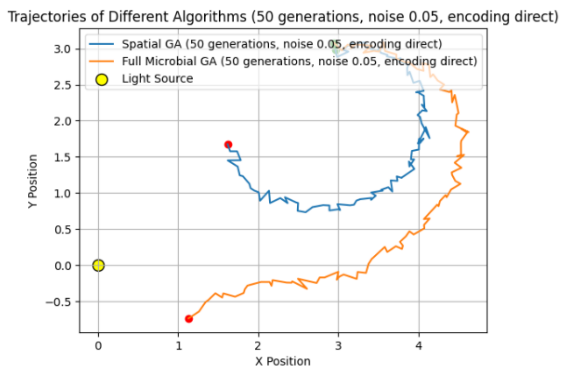
The resultant genotype is also an important element to be referred to.

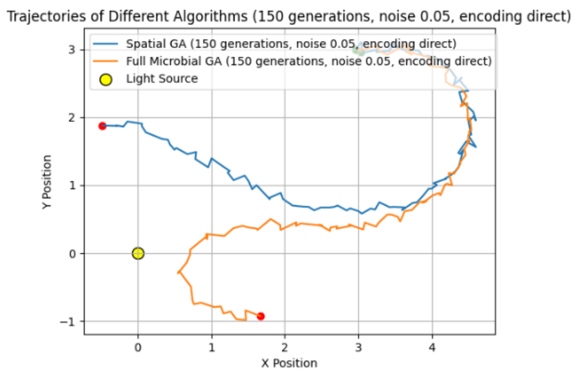
				Value for each segment			
	Generation	W_ll	W_lr	W_rl	W_rr	bl	br
	50	4.579	4.985	2.797	2.087	1.214	1.048
Spatial	150	3.012	0.038	0.464	4.930	2.788	2.398
	300	0.725	3.065	4.563	3.836	2.534	2.121
	50	0.484	3.530	3.137	2.289	2.287	1.862
Microbial	150	1.449	4.514	1.056	0.130	2.954	2.480
	300	1.452	1.840	3.288	3.777	2.341	1.975

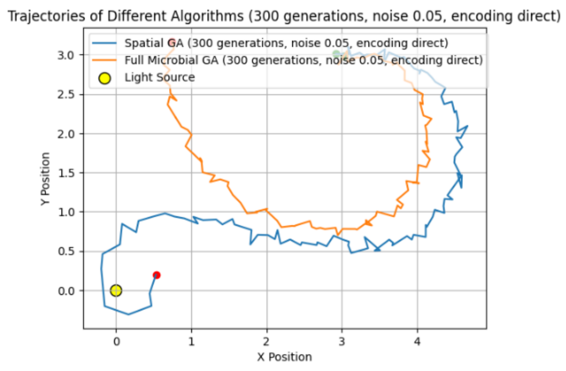
Table: the value of each component in the outputted genotypes

One of the inclinations of the result is that the implication of the right sensor and the left wheel becomes higher as the training generation increases. Presumably, this occurs due to the starting bearing of the robot. Since the starting bearing of the vehicle is 10, the vehicle needs to rotate itself to the right for the first move. As a result, the right sensor and the left motor becomes more sensitive. It might cause some problems for the machine to run straight or turning to the right.

3.1.2 Qualitative comparisons







Let us see the record of trajectory in each cases as a qualitative comparison. The red dots on the graph shows the resultant position of the agent. In every practice, agent firstly turns in clockwise to face to the direction of the light source from its initial bearing to gain more light intensity. The behavior after this motion differs in trained for

trained for 300 generations, it move around the light source to keep the short distance once it get extremely close to the light source. The notable phenomenon in the traials for microbial GA training, the agents try to get away from the light source when they get closer to it in the generation 50 and 150. One possibility is that the genotype adjusted to turn right affect the agent`s motion. Another possibility is that the agent failed with an attepmt to keep the close distance by going around the light so do the agents trained by spatial GA for 300 generations.

3.2 Real-World environment

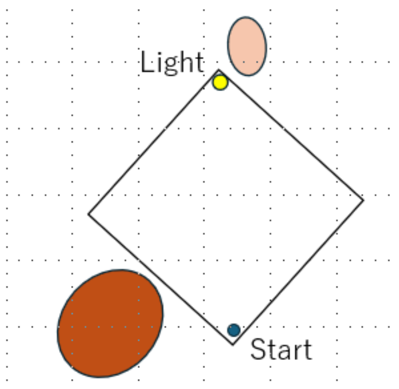
3.2.1 Quantitative comparisons

Microbial GA		performance		left		
Generation	1	2	3	mean	StdDev	Variance
50	96	97	94	95.66667	1.527525	2.333333
150	68	84	86	79.33333	9.865766	97.33333
300	70	77	83	76.66667	6.506407	42.33333
Spatial GA		performance		left		
generation	1	2	3	mean	StdDev	Variance
50	21	24	13	19.33333	5.686241	32.33333
150	127	130	128	128.3333	1.527525	2.333333
300	90	92	104	95.33333	7.571878	57.33333

Table: The distance between the robot and the light source at the end of each trial

Considering that the distance at the initial point is 77cm, the robot becomes more distant from the light source in the resultant location in the most cases. However, the robot with the genotype trained for 50 generations in Spatial GA provides the desired result. Also, the standard deviation for some sections is notably large. It points out the problem in the distance measurements in the experiment.

3.2.2 Qualitative comparisons



For qualitative analysis of the real-world experiment, let us look at the heat-map of the resultant location. The oval area colored with light brown is the location where the agent which trained with Spatial GA for 50 generations reaches. The robot moves towards the light with having deceleration when it approaches to the light. The area colored with dark brown is the one which the robot reaches in all the other case. As soon as they start moving, they made a clockwise circular motion, and end up with reaching to the area.

Figure: heatmap of the final location of the robot

As it mentioned before, the imbalance in the sensor and motor might causes the result. The fact that there is less imbalance in motors for the Spatial GA agent with 50 generation training support the possibility. This also implies that the agent trained by the spatial GA takes longer term to optimize its genotype as the spatial GA is not biased from the condition of the training process.

4 Conclusion and outlook

4.1 Conclusion

The experiment in the simulation was generally succeed. Looking at the quantitative comparisons and quantitative comparison individually, they dipicts the attributes of spatial GA and microbial GA well. Microbial GA produced high performance from quite early stage, and spatial GA begins with relatively lower performance, but the having gradual GA. However, comparing the fitness and the actual trajectory of the agent, the separation could be seen between those elements. We could not find out the certain reasons to explain this phenomenon.

The experiment in the real-world environment could not get the desired movement. All genotypes which had high fitness in the training process takes clockwise circular motion. This shows that the initial bearing of the robot in the training process negatively affects to the real-world experiment. It is difficult to make an answer to the hypothesis about the robustness of the genotypes trained in different approaches from the result.

4.2 Outlook

In order to earn the better results, there are some points that need to be changed in the training process and the real-world experiment. The result shows that the genotypes produce by the training process is not robust enough to well behave in the reality. To solve this issue without changing the approach of GA to use, the first thing that can be done is to make a comparison of genotypes in different situations. For example, the result in the real-world experiment points out the genotypes are overfitted to the bearing setting of the training environment. By taking the average fitness of agents in different initial bearing settings in the training, the model could be robust to the different bearing settings at the beginning in the experiment. It would also be effective to change the hyperparameter settings. For example, increasing the noise to the sensor input would help the agents to experience the training with more reality. Additionally, the fitness function was not sufficient to correctly evaluate the performance of agents. In the practice, the change in distance was the only element to be considered. However, the best movement for the agent is that it takes shortest distance and shortest time possible to the location of the light source, and halt just in front of the light source. Considering this, the average angle difference between the direction that the robot facing and the direction towards the light source could be one of the factors to consider. Also, it might be a strong idea to measure not only how well they did, but also how bad they did in the training. In the experiment, some agents trained in microbial GA made a movement to get away from the light. These cases could be prevented by calculating how far did an agent get away from the light at the end of a trial compared to the moment when it was closest to the light. In the real-world experiment, there was some problems on setting up the environment. The first issue was lacking equipment. This makes it difficult to reproduce the simulation environment and to measure the result accurately. Especially for the light source, the flashlight of the phone was inappropriate since it does not have enough intensity, and it only illuminates one direction. This made it difficult for the vehicle to detect the light in the longer distance or in the case when they are not facing towards the light.

5 Reference

Kumar, M. et al. (2010) ‘Genetic Algorithm: Review and Application’. Rochester, NY. Available at: https://doi.org/10.2139/ssrn.3529843.

Umer, S. (2015) ‘Investigation into mutation operators for Microbial Genetic Algorithm’, in 2015 7th International Joint Conference on Computational Intelligence (IJCCI). 2015 7th International Joint Conference on Computational Intelligence (IJCCI), pp. 299–305. Available at: https://ieeexplore.ieee.org/abstract/document/7529337 (Accessed: 28 May 2024).

Broda, A. and Dzwinel, W. (1996) ‘Spatial genetic algorithm and its parallel implementation’, in J. Waśniewski et al. (eds) Applied Parallel Computing Industrial Computation and Optimization. Berlin, Heidelberg: Springer, pp. 97–106. Available at: https://doi.org/10.1007/3-540-62095-8_11.

Tsoy, Y.R. (2003) ‘The influence of population size and search time limit on genetic algorithm’, in 7th Korea-Russia International Symposium on Science and Technology, Proceedings KORUS 2003. (IEEE Cat. No.03EX737). 7th Korea-Russia International Symposium on Science and Technology, Proceedings KORUS 2003. (IEEE Cat. No.03EX737), pp. 181–187 vol.3. Available at: https://ieeexplore.ieee.org/document/1222860 (Accessed: 28 May 2024).

‘Acquired Intelligence & Adaptive Behaviour Lecture 7: Microbial GA’ (2021). Falmer, Brighton. Available at: https://canvas.sussex.ac.uk/courses/28147/files/4664499?module_item_id=1358368 (Accessed: 20 April 2024).

School of Engineering and Informatics (2024) AIAB Lab3: Reality gap, Google Collaboratory. Available at: https://colab.research.google.com/drive/1CFbqaumd5r7Y0T1vPEbAxaaCU9bqLLnp?hl=ja (Accessed: 2 May 2024).

Chow, R. (2004) ‘Evolving Genotype to Phenotype Mappings with a Multiple-Chromosome Genetic Algorithm’, in K. Deb (ed.) Genetic and Evolutionary Computation – GECCO 2004. Berlin, Heidelberg: Springer, pp. 1006–1017. Available at: https://doi.org/10.1007/978-3-540-24854-5_100.

Chow, R. (2004) ‘Evolving Genotype to Phenotype Mappings with a Multiple-Chromosome Genetic Algorithm’, in K. Deb (ed.) Genetic and Evolutionary Computation – GECCO 2004. Berlin, Heidelberg: Springer, pp. 1006–1017. Available at: https://doi.org/10.1007/978-3-540-24854-5_100.Vose, M., 1994. A closer look at mutation in genetic algorithms. Annals of Mathematics and Artificial Intelligence, 10, pp. 423-434. https://doi.org/10.1007/BF01531279.

Bhatia, A.K. and Basu, S.K. (2006) ‘Implicit Elitism in Genetic Search’, in I. King et al. (eds) Neural Information Processing. Berlin, Heidelberg: Springer, pp. 781–788. Available at: https://doi.org/10.1007/11893295_86.

Jakobi, N., Husbands, P. and Harvey, I. (1995) ‘Noise and the reality gap: The use of simulation in evolutionary robotics’, in F. Morán et al. (eds) Advances in Artificial Life. Berlin, Heidelberg: Springer, pp. 704–720. Available at: https://doi.org/10.1007/3-540-59496-5_337.