

|  |
| --- |
| School of Computing  Faculty of Engineering |

Control System (AI) for Wrestling Robot

Fanhui Meng

Submitted in accordance with the requirements for the degree of  
MSc Advanced Computer Science (AI)

**Session 2019/2020**

The candidate confirms that the following have been submitted*:*

*<As an example>*

|  |  |  |
| --- | --- | --- |
| **Items** | **Format** | **Recipient(s) and Date** |
| *Deliverables 1* | *Report* | *SSO (xx/xx/xx)* |
| *Deliverables 2* | *Code and URL* | *SSO (xx/xx/xx)* |
| *Deliverable 3* | *Youtube video URL* | *Supervisor, assessor (xx/xx/xx)* |
| *Deliverable 5* | *User manuals* | *Client, supervisor (xx/xx/xx)* |

Type of Project: Exploratory Software

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)

© 2020 The University of Leeds and Fanhui Meng

# Summary

The artificial intelligence technology has developed for many decades and used in many fields. This project is focusing on using AI technology, design a high-level control system for Zumo robot in the Sumo robot league.

The Sumo robot league is a popular robot competition, and the main rule of this competition is two vehicle-like robots without mechanical arm trying to push each other out of the ring. The Zumo robot is the one that is going to use in this project. It's an off the shelf Arduino-based robot. Thus, no structure design or hardware design (e.g. circuit design) or low-level design (e.g. PWM motor speed control) in this project.

The high-level control system means the 'brain' of the robot. 'The brain' should have strategies to cope with different situations. There are many machine learning methods to develop strategies for the robot. Mainly, there are two kinds of machine learning can be applied in this project, which is the supervised learning and unsupervised learning. One is to develop the control system by telling the robot, which is the right thing to do and what is wrong. Another is to develop the control system by not telling what the robot should do, but only reward or punishment the robot according to the rules and it's behaviours.

This project aims to implement and compare two different control systems, one is supervised, and another is unsupervised.

Furthermore, this project will do more research about how the same control systems or the same ideas can be applied in other robot competitions or the robot in daily life (e.g. sweeping robot).

# Acknowledgements

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

*<This page should contain any acknowledgements to those who have assisted with your work. Where you have worked as part of a team, you should, where appropriate, reference to any contribution made by others to the project.*

*Note that it is not acceptable to solicit assistance on ‘proof reading’ which is defined as “the systematic checking and identification of errors in spelling, punctuation, grammar and sentence construction, formatting and layout in the text”; see* [*http://www.leeds.ac.uk/qat/documents/policy/Proof-reading-policy.pdf*](http://www.leeds.ac.uk/qat/documents/policy/Proof-reading-policy.pdf)*. >*

# Table of Contents

Summary iii

Acknowledgements iv

Table of Contents v

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Overview

1.2 Aim and Objectives

Chapter 2 Tables and Figures 3

2.1 Tables using the ‘table caption’ and ‘table description’ Styles 3

2.2 Figures using the ‘figure caption’ and ‘figure description’ Styles

List of References 31

Appendix A External Materials 33

A.1 Level 2 Heading with ‘heading 2’ Style Applied by Pressing Ctrl Shift 2 33

A.1.1 Level 3 Heading with ‘heading 3’ Style Applied by Pressing Ctrl Shift 3 33

A.1.1.1 Level 4 Heading with ‘heading 4’ Style Applied by Pressing Ctrl Shift 4 33

Appendix B Ethical Issues Addressed 34

B.1 Level 2 Heading 34

# List of Figures

Figure 1.1 Main features of the Zumo 32U4 robot

Figure 2.1 Decision tree example for surviver on Titanic (Wikipedia Contributors, 2019)

Figure 2.2 General process for GA

Figure 2.3 General procedure for reinforcement learning

Figure 3.1 Decision tree based control system general architecture

Figure 3.2 The general architecture of GA control system

Figure 4.1 Example code for display sensor reading

Figure 4.2 Experiment reading and distance in real world

Figure 4.3 Part of the dataset for decision tree

# List of Tables

Table 2.1: Fuzzy rules for sumo robot (Erlan et al., 2013)

Table 2.2: Genes for robot attributes (Lehner et al., 2019)

Table 2.3 Occurrence of top 10 fittest genetic combinations (Lehner et al., 2019)

Table 2.4: Bayesian example for sumo robot

Table 3.1 Function requirement for decision tree based control system

Table 3.2 Function requirement for GA based control system

Table 4.1 Libraries and purposes in decision tree based control system

Table 4.2 Libraries and purposes in decision tree based control system

Table 4.3 Reading and distance comparison tabl

# List of Abbreviations

BEAST Bio-inspired Evolutionary Agent Simulation Toolkit

GA Genetic algorithm

FLC Fuzzy Logic Control

# Chapter 1 Introduction

## Project Aim

This project aims to design an excellent robot high-level control system, which is the 'brain' of the non-arm Zumo robot. The control system would have its own strategy and drive the robot's movement during the competition. As for the strategy, it can be assigned with a specific action in different cases, or the robot can develop its own strategy, which may be related to the evolutionary algorithm. And this will be discussed later in the report.

The main goal of this project is to make the Zumo robot be competitive and perform well in the Sumo robot competition. Besides, this project will find out if the idea of a control system can be applied to a broader range of different robots, such as other robot competition or the robot service in daily life.

This project is going to use the Zumo 32U4 robot, which is a complete, versatile robot controlled by an Arduino-compatible Atmega32U4 microcontroller. Therefore, extra hardware structure and improvement is not considered in the project. The Zumo robot has two motors, one Atmega32U4 chip as the brain and a variety of sensors, including two front proximity sensors, two side proximity sensors, five line sensors (three of them are the receiver, two of them are emission) and accelerometer. So the Zumo robot can detect the opponent and run towards or away from it, which satisfy every requirements of Sumo league.



Figure 1.1: Main features of the Zumo 32U4 robot

Sumo robot league is a prevalent international robot wrestling competition, which is two robots attempt to push each other out of the ring. Thus, the wrestling robot must be capable of autonomously locating the opposing robot and pushing it out of the ring (Wilson et al., 2016). The last stand robot in the ring is the winner. Also, there are other rules in the Sumo league, which the control system should be designed according to these rules. And these rules will be explained in details in the next chapter.

### Objectives:

* To get familiar with the hardware functions of Zumo robot.
* Conduct a theory study to compare different control system in the wrestling case.
* Implement two or more different control system and compare it’s advantages and disadvantages.
* Create a wrestling simulation environment. (Due to the lockdown policy, it’s hard to find the opponent in the real world)
* Evaluate the results with its performance in different simulation cases.

#### Deliverables

1. A software that can simulate the Sumo robot wrestling. Built using BEAST.
2. A Github repository that contains the source code of the system.
3. A developer documentation that provides:

* An overview of the simulation, algorithm used, programming languages and style.
* Instructions for setting up the project in a local development environment. (Provide VM for Mac or Windows user)

1. The MSc project report.

#### Ethical, legal, and social issues

This project is trying to avoid all ethical issue. All the data used in developing the software are from the Zumo robot sensor data and the BEAST, no personal data or classify data contains in this project. All Arduino library and BEAST external code that used in developing the software are open sources. And these codes were not credited as own work. These codes will be downloaded and put in a separate folder in the repositories to avoid plagiarisms.

And this project’s code and ideas should not be used in the military field or any field that could cause violence action.

# Chapter 2 Background Research

## 2.1 Literature Survey

The sumo robot contest feature two self-controlled robots trying to push each other out of a ring. (Anon, n.d.) The basic rules are the first robot that touches outsides of the ring loses the round, and the last stand robot wins the round. The match is the best of three sets. And the robot wins the most matches are winning the contest.

Several structure requirements for the participant robot (e.g. robot size, mass, etc.) are clearly defined in the sumo robot rules. But these requirements are not being considered in this report, since this project's experiment and simulation are based on ready-made Zumo robot, which already satisfies all the hardware and structure aspect requirements.

Regarding the contest environment, it’s a large, flat ring. The ring made of smooth, rigid wood. The top surface is dull black, and all of these black areas are in bounds.

A robot usually started by pressing a button or other ways, such as hand-clapping, a whistle and so forth. After start command, the contestants immediately leave the exterior area around the ring. And each robot must not move at all for five seconds.

5 seconds after pressing the button, the robot should move and start the competition. However, if the robot does not move for a long time after the start command. It will be punished or judged for losing the round.

During the round, all people and object must be kept out of the ring to avoid distracting the robots or altering it’s outcome.

A robot loses the round when any portion (including touch sensors, whiskers, scoops, or skirts) of the robot touches outside the ring. It doesn’t matter if the robot falls out on its own or is pushed out. Or the components of the robot is fall out of the ring is also considered as lose the round.

If the match satisfiy any of the conditions below, the reference may choose to restart a round.

* Three minutes have expired
* No progress has been made in some period of time
* The robots fail to touch each other for some period of time
* The robots are hopelessly entangled or otherwise deadlocked
* Both robots fail to start or both contestants signal stoppage

This project design is based on the sumo rules above. The robot’s action and strategy, and the way that evolve the robot’s strategy should be impacted by the rules as well. For example, the robot’s should not stay still too long, even this is part of the strategy, because the rule will punish the robot that doesn’t move for an extended period.

There are pretty much studies in Sumo robot control and even more research on the general intelligence control system. And there are also different aspects of this project. Some are fucus on the low-level development, and some aim to design the hardware or the construction of the Sumo robot, others are focus on electronics aspect and so on. And this project is going to focus on the high-level design.

One study uses single fuzzy logic control (FLC) as the microcontroller for detection and tracking of an opponent in the competition ring. Three sharp infrared sensors for target detection. Then the fuzzy controller fuses the sensor data and provides the control signal to the motor for driving the robot toward the opponent (Erdem, 2007).

Then the author develops the fuzzy rules, which are intuitive rules that can be driven by all possible scenarios with input sensor values. For example, one of the fuzzy rules is that, if S1 and S2 detect the weak signal and S3 detects a medium signal, the logical action of the robot is to control the motor to take a small turn to the right. The optimized rules for detection and tracking of the target are shown in below.



Table 2.1: Fuzzy rules for sumo robot (Erdem, 2007)

For every sensor, there are three fuzzy sets, including Low, Medium, High. The five singleton membership functions are Full Left, Small right, etc. Based on the output of the fuzzy controller, the motor will make the robot turn left or right.

The experiments result concluded as two aspects. First is the sensors perception area, when sumo robot is staying still, three proximity sensors detect moving object nearly for 0.02 second. The second experiment is when the head of the robot is against the object, and sensors cannot sense the opponent. Robot can starts in search mode and turn its head directed to the object, then speed up until hitting the object and push the opponent out of the ring. And the result shows, in this condition, which is in searching mode and rotational motion, the robot can response and rotate, detect an opponent in 0.2 seconds as it’s within the sensor range and hit the opponents every time.

In summary, compared with the traditional sumo robot control algorithm, the designed system takes minimum time for execution of control algorithms. And it also has advantages in fast sensing, localization and manoeuvring in time.

Similarly, another project also uses fuzzy logic as the main idea of sumo robot control. It uses the Neuro-Fuzzy (NF) hybrid system as the control system (Erdem, 2011). In other words, it uses two systems, which is ANN and Fuzzy Inference System (FIS). FIS is for detecting and tracking the opponent, which relates sensor output (IR sensor) to motor control pulses. ANN is used for rule extraction and tuning the FIS parameters. Furthermore, NEFCLASS model is introduced, which can be evaluated as a hybrid ANN-based system. NEFCLASS has an ANN like structure, which uses fuzzy parameters, or it can be interpreted a FIS implemented in a parallel structure. The primary function of NEFCLASS is to using a learning algorithm to create the structure (rule base) and tuning the parameters (fuzzy sets) of a fuzzy classifier from the system’s data. Or, in other words, NEFCLASS optimises the FIS parameters by adding and deleting the rules.

In this way, with FIS and NEFCLASS, the number of original fuzzy rules can be pruning down to less. With the optimized number of fuzzy rules, the robot’s control can respond faster than before.

And the result shows that this NF control system can improve robot performance in the wrestling league. Compared to the traditional way (single fuzzy control), application of learning algorithm has provided 30% rule reduction, which improves the robot response speed in the competition. And the robot can attack the opponent with minimal manoeuvres. The performance can be improved with adding extra sensor for sensing the opponent, which can be seen as the direction of future study.

This is a good thought to develop a good control system. It's pretty much the low-level development. It's just like human work out to improve physical strength, but I would focus on brain development. However, the wrestling environment is uncertain, and the data is non-linear. Thus, the method that using ANN to improve fuzzy control is a good thought for high-level control as well.

Besides, a few studies use genetic algorithm (GA) to optimize the sumo robot. One study uses Java with Eclipse and the *dyn4j physics engine* for simulating the sumo robot fight (Lehner et al., 2019). Each robot has six attributes (e.g. the speed, search range), which are allocated to gene positions and can be controlled by the GA. The intensity of each genetically controlled capability is the characteristics, which is expressed with a value ranging from 1 to 9.

The genes for robot attributes and it’s controlled capability is shown below.



Table 2.2: Genes for robot attributes (Lehner et al., 2019)

The process for each robot is, at the start of the fight, each robot receives a randomly selected mix of values, which is the combination characteristics for the robot. Then run the battle to determine the highest fitness. The fitness is defined as how many movements are required until it's defeated or stands last. The next step is the crossover and selection for the next generation. This project uses roulette wheel selection, which distributes the probability for the selection of each robot based on their relative likelihood of the fitness. And 50% mutation rate per gene is also applied to increase the probability of changing the characteristics randomly. Finally, create a new generation and transfers the new generation in a new loop. After thousands of generations, the last robot stands in the ring is the final winning genes of the GA optimization.

The experiment concludes the top fittest genetic combinations and has generated 630 different sets of genes from 1200 generations. The result shows that there is no winning genes appeared in different variants. Therefore, the dominance of winning genes is random because no winning genetic combination appears in more than one variant. Isolated analysis of single capability levels does not provide the winning tactic for a robot. When comparing the individual with the combined genes, no pattern seems to exist because, in the winning genes, the dominant attributes do not possess the highest characteristic value. Probably because the random starting positions of each robot for each new fight, and the researchers say this need further study.



Table 2.3 Occurrence of top 10 fittest genetic combinations (Lehner et al., 2019)

Another result shows that different selection methods do influence the success of the robot. A roulette wheel selection distributes the gene proportionally, which led in this experiment to a low dominance of the generated genes compared to the results when using the randomness.

## 2.2 Methods and Techniques

2.2.1 Fuzzy Logical Control System

Fuzzy control is an intelligent control method based on fuzzy set theory and fuzzy logic inference. It's an intelligent control method that imitates a human's fuzzy inference and decision-making process from the behaviour. The technique first compiles the experience of the operator or expert into fuzzy rules, then fuzzifies the real-time signal from the sensor. Use the fuzzed signal as the input of the fuzzy rule, completes the fuzzy inference and output the inference.

Several things are essential for defining a fuzzy controller. First is the fuzzy interface, which is for performing a conversion from a specific real value of the input into a fuzzy set. For example, in this project, it may transfer the proximity sensor output, such as a number value, to a fuzzy set, like low, medium or high. Secondly, the knowledge base, which is consist of data base and rule base. The data base stores all the input and the fuzzy set. At the same time, the rule base is the fuzzy rules usually based on expert knowledge or personal experiment. A series of relational words often connect fuzzy rules, and the most commonly used relational words are if-then, else, etc. Finally, is the inference and defuzzy-interface. The result of the inference indicates that the rule inference function of fuzzy control has completed. However, the result obtained still cannot be used directly as a control variable. A conversion must be made to have a precise control variable output, which is the defuzzification. In this project, this could be used for determination of real value for directing of motors or other controls.

Fuzzy logic control doesn't require an accurate mathematical model. However, the factors such as the integrity of fuzzy control rules, the definition of fuzzy subsets and the fuzzy inference mechanism will have an impact on the performance of the fuzzy controller. Thus, most of the factors depend on the experience of experts. For this project, the primary basis of control based on a large amount of personal experience, knowledge and strategy in the sumo league.

2.2.2 Decision tree based Control System

Similarly to FLC, the performance of the decision tree based control system also depends more on personal experience and strategy in a wrestling robot. As one kind of supervised learning, the decision tree can be well applied in the classification as well as the regression problem.

The decision tree algorithm uses a tree structure and uses layers of reasoning to achieve the final classification. It’s consist of one root node, leaf node and internal node. Root node contains a full set of samples; the internal node corresponding characteristic attribute and the internal node represented the final result or decision.

For example, in Figure 2.1 below. The root node is the ‘gender’ node, which must be the purest feature. ‘age’, ‘sibsp’ are the internal node. ‘survived’, ‘died’ are the predicted result, which is the leaf node.



Figure 2.1: Decision tree example for the survivor on Titanic (Wikipedia Contributors, 2019)

Basically, there are three-step to build the decision tree. First is the feature selection, and it's determining which features are going to make judgements. In the training data set, there may be many attributes of each sample, and the effects of different attributes are different. Therefore, the function of feature selection is to screen out the features that are more relevant to the classification results. In short, it's intended to find out the features with strong classification ability. After the features are selected, the next step is to generate the decision tree. It's triggered from the root node, calculated purity for the node. The feature with the most purity is selected as the node feature, and the child nodes are established according to different values of the feature. Each child node is generated in the same way until purity is the lowest, or there are no features to choose from.

The key things here is the word ‘purity’. It’s deciding which is the optimal partition feature. And it means if most of the sample that contained in one tree node belongs to one category, then we say this tree node is pretty pure. There are mainly two ways of representing purity, which is the information entropy and Gini index. According to this, three decision tree algorithm is introduced, including ID3, C4.5 and CRAT. ID3 based on information gain to choose features. C4.5 is quite similar to ID3, which use information gain ratio. And CRAT uses the Gini index to select features, which can be used in both classification and regression problem.

After generating the decision tree, a final step is an option, which is pruning. The primary purpose of pruning is to prevent overfitting by removing unnecessary branches.

Furthermore, random forest is another possible solution for the control system, which made of many independent decision trees. When performing a classification task, new input samples are entered, and each decision tree in the forest is judging and classified separately. Each decision tree will get its own classification result. And the conclusion that most of the decision tree classified is the final result for the random forest.

2.2.3 Artificial neural network

An artificial neural network (ANN) is a computing model, which is composed of a large number of artificial neurons connected to each other. Each node represents a specific output function. Each connection between two nodes represents a weighted value for a signal passing through the connection, which is called the weight. The network itself is usually an approximation of a specific algorithm or function or an expression of a logic strategy.

Feedforward neural network (FNN) is one kind of ANN. It's the most straightforward and widely used type of ANN. In this network, the information moves in only by forward. There are no cycles or loops in the network.

For this project, the sensor value can be the input of the network, and output is the generated strategy. Moreover, the ANN’s parameter (e.g. weight, basis) can be adjusted or optimized or evolved through training. Thus, a different way of optimization can be applied, such as Neuro-fuzzy or GA with FNN.

2.2.4 Naïve Bayesian Model

Bayesian is a kind of classification method. For example, suppose a robot’s right proximity sensor detects obstacle while the left IR detect nothing. In that case, this situation can be classified as an obstacle in the right-hand side of the robot. The content of the classification algorithm is to require a given feature to let us get the category, which is also the key to all classification problems. And how to specify the characteristics to get our final category is the key thing to be considered.

One example of sumo robot bayesian model shows as below. For proximity sensor, 0 means the distance to the obstacle is close, one means the distance is far, two means too far. For linesensor, 0 means out of the ring, one expects in the ring.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Proximity Sensor  Front Right (S1) | Proximity Sensor  Front Left (S2) | Line Sensor  Right (S3) | Line Sensor  Left (S4) | Label |
| 0 | 0 | 1 | 1 | Enemy Front |
| 0 | 1 | 1 | 1 | Enemy Front Right |
| 0 | 2 | 1 | 1 | Enemy Front Right |
| 1 | 0 | 1 | 1 | Enemy Front Left |
| 1 | 1 | 1 | 1 | Enemy Front |
| 1 | 2 | 1 | 1 | Enemy Front Left |
| … | … | … | … | … |

Table 2.4 Bayesian example for sumo robot

The bayesian formula is . A means the features, and B means the label or category. And each feature should be independent.

So in the sumo case example, if it require to calculate the possibility if the enemy is in the front or left or right, and the sensor values are S1, S2, S3, S4 respectively. Then use bayesian formula to calculate the possibility of front or left or right respectively.

First, calculate the possiblity of enemy in the front. It can be represented as below.

Because of each feature is independent, .

And

After getting everything we need, can be calculate.

Then do the same to and . The posibliity with the largest value is the final predicted result.

In general, the good thing about bayesian is that it’s easy to implement and supposed to have less error rate.

2.2.5 Genetic Algorithm

Genetic Algorithm (GA) is based on Darwin’s theory of evolution, simulating natural selection and the survival of the fittest. The primary purpose of GA is to keep the best gene so that it will evolve better and more fit gene or individuals.

GA usually uses a fixed-length linear binary representation for its genotype. It can be designed as containing the control strategy of the sumo robot. And it’s a heuristic algorithm, which is suitable for Non-deterministic polynomial (NP) problem. This will be explained in the next section.

The general steps of GA is shown below.



Figure2.2: General process for GA

At the first generation, the population will be the group of randomly generated solution. Then use the fitness function to calculate the fitness of each individual separately.

Fitness is the most crucial thing about GA. It’s playing the role of ‘God’ in GA. Fitness score measures the pros and cons of individual and decides who is going to stay or be eliminated. For example, in sumo cases, the fitness can be if the robot wins the most matches, then it’ll get more fitness score, while the one loses the most games are tend to get a lower score. GA will perform N generations, and each generation will generate several individuals, which is population. The fitness function will give a score to all the individual to judge the fitness for each individual. Only the individual with higher fitness are retained, so the quality of the population will become better and better after several iterations.

After calculating the fitness score, the next step will be the selection, which is also quite crucial for GA. There are several ways of selection. For example, the main idea of roulette wheel selection is that individuals with better fitness values are more likely to be selected. The tournament selection takes a certain number of individuals from the population each time and selects the best one to enter the offspring population. Repeat this operation until the new population size reaches the original population size. Ranked based selection, which only selects the top individual according to fitness score. There are also many other selection methods, like stochastic tournament, excepted value selection, truncation selection, and so on.

After selection, GA will finally go to the evolution process. Do the crossover first, then mutation. The crossover needs to find two chromosomes from the selected individuals of the previous generation. Then cut a specific position of the two chromosomes and splice them together to generate a new chromosome. This new chromosome contains a certain number of the two individuals’ genes from the last generations. And there are many different ways of design the crossover proportion. For example, 70% of the new chromosome comes from the highest fitness score, 30% from the second. Mutation means to change the small part of the genes randomly. Introduce mutation can help the solution escape from local optimal and is helpful for the algorithm to find the global optimal solution.

Then calculate these new generations’ fitness score, and do the selection, crossover, mutation again until the result is good enough or reach the number of iterations limits.

2.2.6 Reinforcement learning

Reinforcement learning is also unsupervised learning. And it’s usually described by Markov Decision Process (MDP). Then the overall concept is for reinforcement learning is that a machine is in an environment, and it can only interact with the environment through actions. And the state is the machine’s perception of the current environment. When the machine performs an action, the environment will feedback to the machine a reward based on the potential reward function. Also, it will make the environment transfer to another state according to a certain probability. In summary, reinforcement learning mainly includes four elements, which are state, action, transition probability and reward function.

The general procedure for reinforcement learning shown as below.



Figure 2.3 General procedure for reinforcement learning

As the agent performs a certain task, it first interacts with the environment to generate a new state, and the environment gives a reward at the same time. Then continue the iteration, the agent and the environment continue to interact to generate more new data, and modify its own action strategy according to these new data. After several iterations, the agent will learn the action strategy needed to complete the task.

In this project, the agent would be the Zumo robot. And state including the environment state, which is the sumo ring, agent state, which is the input data for agent and information state, which include useful information needed for future prediction. The environment is the partially observable environment, which means the agent will discover the environment by with its sensor.

There are several algorithms for reinforcement learning, including Q learning, Sara, Deep Q network and so on. For example, the famous Q learning based on Q chart like the chart below.

|  |  |  |
| --- | --- | --- |
| Q | A1 | A2 |
| S1 | -3 | 3 |
| S2 | -4 | 4 |

Table 2.4 Example for Q chart

S means the state, and A means action in Table 2.4. In this Table 2.4, , which means A2 rewards more than A1. Thus, it would be reasonable to choose A2 as an action. Then comes to S2, repeat the previous steps, choose the most rewards action and get to the next state.

However, the Q chart includes the estimated value, so this chart needs to be updated. For example, , so multiply the max value by an attenuation value gamma (e.g. 0.8) and add the reward when reaching S2. And this value is seen as the actual value. Now we got the actual and estimated value, and the Q chart is able to be updated. According to the difference between estimation and actual, multiply this difference by a learning rate and add the old estimation value , then got the new updated value.

## 2.3 Choice of methods

There are many AI methods out there for designing the control system for the Zumo robot. I always do believe the same that the best way to evaluate one method or idea is to experiment with it. However, there is no time for me to implement and test all the AI technologies in the Zumo robot. So I have to choose a suitable method according to theoretical research.

In my opinion, two different kinds of methods are necessary to be implemented to the control system in this project. One is supervised learning, and another is unsupervised learning.

Supervised learning means that with input variable and output variable, and use a certain algorithm to learn the function from input to output. By this method, the human strategy can be implemented into the Zumo robot, and the control system can design as what people want the behaviours of the robot. As for the decision tree, which is one kind of supervised learning, it’s easy to understand and explain, can be visualized and analyzed. And it’s an excellent way to get a start and familiar with this project. It’s able to handle both data type and regular type attributes. Also, when dealing with the data set, the program run time can be very fast, which means the time complexity is low. The speed of running code is quite important because the Zumo robot is off the shelf robot, the processor and memory on the Zumo robot are fixed. And it can’t compile too many codes and unable to run the code with much time complexity. The sensor output of the robot can be outdated soon, so the robot should react timely according to the current sensor output. And these conditions emphasize that the code should not be too complicated, and it should be as easy as possible. For the decision tree, the code can be less complex and easy to compile and run, and it will perform the strategy correctly as expected.

As for unsupervised learning, which is going to be considered as the majority part of this project. Because in the sumo league, it’s hard to say what is the right thing to do and what is wrong. And no one knows the absolutely undefeated, unrivalled strategy. It seems wrong that people are trying to train the robot with the best strategy while people themselves don’t even know what is best. However, this is just subjective thinking, which kind of learning method have better performance has to be determined by experiment.

Wrestling robot fight can be defined as an NP problem. Don’t know if a correct solution or strategy exists for sumo league. However, for any possible solution or strategy can be verified in polynomial time. For NP problem as well as sumo league, there are no perfect solution or global optimal. In this case, the heuristic algorithm is suitable for this kind of problem, and it will provide a feasible solution within an acceptable time.

GA is a kind of unsupervised learning as well as a heuristic algorithm, which is suitable for this project. GA was introduced in COMP5400M Bio-inspired computing module last semester, which gives me a general understanding of GA, including the concept of heredity, variation, selection, etc. Also knowing that GA has many applications, including designing a LEGO bridge using modified CAD software, optimise the shape of airfoils and so on. A GA simulation toolkit called BEAST is also introduced in this module. BEAST can be used to simulate many cases using GA, like the mouse with cheese, predators and prey simulation.

GA also has large search space, which reduces the risk of falling into a local optimal. The GA does not use deterministic rules, but uses probabilistic transition rules to guide his search direction. The fitness funciton can be customize, which can have more possibilities. And these characteristics of GA makes GA get more ‘fitness score’ in this project.

With BEAST, an Animat class with a built-in feed-forward network and GA compatibility is already provided. This class returns the neural net’s configuration vector as its genotype. It initialises the net with one input per sensor and one output per control (e.g. the right and left wheel in the basic Animat). The built-in FNN’s input, output, number of the hidden layer are able to set. All in all, the control method would be: feed the input from the sensor into the FNN and then sets the motors using the network’s output. And the way to optimise the parameter of FNN and configure the network would be using GA.

One reason why I prefer the simulation in this project is that GA is usually going through hundreds or even thousands of generations. There are tons of experiments in GA, and it would be impossible to experiment with the whole evolution progress in the real world. Another reason is that this year is a special year due to Covid-19, and the closedown of the school, the lockdown of the country make it’s hard to find another real sumo robot as opponents. Many competitions are cancelled, including the School of Computing Zumo Wrestling league. Thus, simulation robot wrestling would fit this project best, and this project is also going to emphasis more on simulation.

# Chapter 3 Software Requirements and System Design

## 3.1 Software Requirements

The project requirements focus on the design of a well-performed high-level control system. And there are two ways of designing the control system in this project, including decision tree and GA based control system. Thus, two different systems with different functional requirements (FR) are going to be listed in this chapter.

Table 3.1 state the decision tree based control system. Table 3.2 state the GA based control system.

|  |  |
| --- | --- |
| FR 3.1 | Requirements |
| FR 3.1.1 | System shall generate decision tree accoding to the data of sensor reading and label. |
| FR 3.1.2 | System shall provide three different algorithm option to generate decision.  (ID3, C4.5 and CRAT) |
| FR 3.1.3 | System shall allow detail changes for the content of action model (label).  (e.g. adjust speed or rotate speed) |
| FR 3.1.4 | System shall allow adding more kind of sensor data |
| FR 3.1.5 | System shall visualize the generated decision tree |
| FR 3.1.6 | System shall show each node’s detail of the generated decision tree.  (feature, gini/ information gain ratio, number of sample, etc.) |

Table 3.1 Function requirement for decision tree based control system

|  |  |
| --- | --- |
| FR 3.2 | Requirements |
| FR 3.2.1 | System shall have two independent agent class represent two Zumo robot. |
| FR 3.2.2 | System shall create the simulation wrestling environment.  (Two robots and one ring with the same proportional when compared to the real world) |
| FR 3.2.3 | System shall have the same kind of sensor or the same function of sensor that Zumo robot has in real world.  (Proximity sensor, line sensor) |
| FR 3.2.4 | System shall have three selections algorithm options.  (Roulette wheel selection, ranking selection, tournament selection) |
| FR 3.2.5 | System shall allow the changes of parameter of GA.  (number of population, crossover proportion, mutation pability) |
| FR 3.2.6 | System shall allow the changes of FNN.  (The number of input, output and hidden layer) |
| FR 3.2.7 | System shall present the average fitness and the best fitness score in each generation. |
| FR 3.2.8 | System shall present the time of each generation. |
| FR 3.2.9 | System shall include the sumo robot league rules.  (If one of the robot run out of the ring, restart the game and reset both robot’s location, orientation, etc.) |
| FR 3.2.10 | System shall draw the plot for the final results after evolution.  (Two agents’ average and best fitness score in each generation) |

Table 3.2 Function requirement for GA based control system

Both two systems’ requirements will be evaluated and addressed in Chapter 4 and 5, which are implementation and software testing chapter. Moreover, the README file will be provided.

For the record, because hardware and structure are not considered in this project. The sumo robot league rules like *‘if any piece of the robot touches or fall outside the ring, the robot is considered out’* are not part of this project’s requirement.

## 3.2 System Design

As the description above, this project will design two control system for Zumo robot, one is unsupervised, and another is supervised learning. In this design section, two control systems’ architecture and technology will be discussed respectively.

3.2.1 Decision tree based control system design

The Zumo robot is controlled by an Arduino-compatible Atmega32U4 microcontroller, which is a fine chip, but it’s memory and calculates ability are limited. So the process of generating the decision tree is going to execute in another environment. Then transfer the generated decision tree into the Zumo robot program.

In the Zumo robot program, input the sensor reading to the decision tree, then the output would be the label. The label is then corresponding to a control mode. The control mode can be written as a function, and the content of this kind of function would be the motor control. For example, if one label or control mode comes to ‘Charge’, then the content of this function would be set both motors at maximum speed.

And the dataset that used to generate decision tree would be written by hand, but also comes from the real experiment result. First, writing a simple program that the LCD screen would display the sensors real-time data. Then use a cup as the robot’s opponent put the cup at different position of the robot and observe the sensor readings. Assign a different label or action mode to different sensor reading. Then come up with the data; each line of the data has the reading from a separate sensor and a label. And the data set would be saved in a txt file. The code should read the file and extract the data and label from it.

The dataset would be small. One reason is the microcontroller wouldn’t process too much data. Another reason is that the explanation in section 2.3 choice of method that the decision tree is not going to be the major part of this project. Thus, a feasible but not very detailed dataset would fit this project.

As for the algorithm for generating the decision tree, there will be slightly different between different algorithm because the dataset is quite small. This project would use the CRAT algorithm, which means to use the Gini index minimization criterion to select features and divide. However, the system would still provide three options for choosing an algorithm and use CRAT as implementation. And the comparison of these algorithms wouldn’t be involved in this report.

The general architecture from a logical point of view shown below in Figure 3.1.



Figure 3.1: Decision tree based control system general architecture

3.3.2 GA based control system

The GA based control system is going to have an experiment in a simulated environment, including the simulated agent, sumo environment, etc.

There are two agents class in the environment. And these two agents class are not sharing the same GA process, which means their evolutions are independent, but also can be seen as co-evolution or competition relationship. This relationship just like predator and prey, they are two different species, but their evolution progress can affect each other. The common things of these two agents are that they have the same framework, which is both controlled by FNN, and GA is used to adjust the parameter of FNN. Also, the physical structure of agents is the same, including the same sensors, same size and same motors, which means the same maximum speed.

The assessment means the wrestling robot fight in within time. It can be called as test time or battle time. Consider that even the agent with better behaviour could lose one or two battles, the assessment won’t stop just because one agent is out of the ring. Only when the time is up, the assessment ends. And suppose one agent is out of the ring while the assessment is not over. In that case, the system will reset everything in the environment, including the agents’ position, orientation and so on. Also, there are two counters, which record how many times each agent has run out of the ring within one single assessment. And the counters can be seen as one major factor to calculate the fitness score.

Then general architecture of GA control system are shown below.



Figure 3.2: The general architecture of GA control system

# Chapter 4 Software Implementation

This project has designed two control system, as mentioned above. Therefore, this chapter is going to discuss the two system implementation, respectively.

* 1. Decision tree based control system implementation

Follow the discussion in Chapter 3. This system is going to implement in two different environments, one is for generating the decision tree; another is to guide the robot’s behaviour according to the decision tree output. The next few sections are going to discuss the implementation in detail.

* + 1. Technologies employed in this system

Because this project is about hight-level control system design, so it’s reasonable to use object-oriented programming language.

The decision tree generating progress would use Python, and several libraries, which will be listed in Table 4.1. The file type of this code would be .ipynb, which means this code is able to run in Colab or Jupyter notebook environment. For some of the library may not be pre-installed in the user’s local machine, run the code in Colab would be my recommendation.

|  |  |  |
| --- | --- | --- |
| Table 4.1 | Library | Purpose |
| T 4.1.1 | from sklearn.preprocessing import LabelEncoder | For data preprocess, convert string to incremental value for serialization. |
| T 4.1.2 | from sklearn.externals.six import StringIO | Mainly used to read and write data in the memory buffer. |
| T 4.1.3 | from sklearn import tree | Main library for generating the decision. Provide decision tree model for solving classification and regression problems. |
| T 4.1.4 | import pandas | In order to serialize string type data, pandas data was needed, which is convenient for the serialization work. |
| T 4.1.5 | import pydotplus | Used for the visualization of decision tree. |
| T 4.1.6 | import JSON | Used for transfer the generated decision tree to the Zumo robot code. |

Table 4.1 Libraries and purposes in decision tree based control system

As for the code that is going to compile and run in Zumo robot, it’s written in C++, and the software for compile and run is Arduino IDE. The type of the file is .ino, which are able to be upload to Zumo robot microcontroller. And the libraries and purpose are shown in Table 4.2 below.

|  |  |  |
| --- | --- | --- |
| Table 4.2 | Library | Purpose |
| T 4.2.1 | ArduinoJson | A simple and efficient JSON library for embedded C++. (Blanchon, 2020) |
| T 4.2.3 | Zumo32U4 | Zumo 32U4 robot Arduino library for the Arduino IDE that helps interface with the on-board hardware on the Pololu Zumo 32U4 robot. (e.g. line sensor, proximity sensor, motor, led, lcd, button, accelerometer, etc.) |

Table 4.2 Libraries and purposes in decision tree based control system

* + 1. Create dataset

As mentioned in section 3.2, the idea of creating the dataset to generate the decision tree is to write by hand. Then write and upload a simple code to Zumo robot, which is to display the real-time sensor reading in the LCD screen. Part of the code shown in Figure 4.1.



Figure 4.1 Example code for display sensor reading

Experiment and find out the sensor reading corresponding to the distance by using a cup as the opponent. About the proximity, I only experiment the front left one, because the proximity sensor in the Zumo robot is the same. And the result is shown below. (Note: the distance is an approximate value, measured by using the *'Measure'* app in iPad. There is a deviation of two or three centimetres).

|  |  |  |
| --- | --- | --- |
| Proximity sensor reading | Distance (Near end) | Distance (Remote end) |
| 0 | N/A  (Too close to measure) | More than 60 cm |
| 1 | N/A  (Too close to measure) | 37cm – 60 cm |
| 2 | 1cm – 5 cm | 20 cm – 37 cm |
| 3 | 6cm – 19cm | 6 cm – 19 cm |

Table 4.3 Reading and distance comparison table



Figure 4.2 Experiment reading and distance in real world

And do the same for the line sensor, which is used to detect the black ground. After the experiment, the threshold is approximate 1100. If any line sensor’s reading is large than 1100, then it’s out of the ring or touching the white border. Else, it’s still inside the ring.

After knowing the reading and distance, the next work would be assigned the label. For example, if both proximity sensor reading is 2 or 3, which means opponents detected, and the label would be object seeing.

If the reading is 0 or 1, there are two possibilities. One is the robot has hit, or extremely close with the opponents, Another situation is the robot is extremely far from the opponents or no opponents detected. Thus, only two proximity sensor reading could be misleading, so the accelerometer is introduced in this project. One represents detecting speed change, which means hit the opponents. 0 means everything is normal. Therefore, with an accelerometer, it’ll be more transparent for the generated decision tree to make a decision.

Part of the dataset is shown in Figure 4.3. The whole dataset would be avialiable in my repository.



Figure 4.3 Part of the dataset for decision tree

* + 1. Generate decision tree

Step one: Fetch the data.

Save the data file in Google drive, then use Colab fetch the data.



Figure 4.4 Code of fetch data in Colab

Step two: Create a dictionary called *mode\_dict* to store the data.

Load the data file, and process. Create the feature label array as *modeLabels*. In *mode\_dict* the key is the feature label, value is the the rest data in each line, except the label.





Figure 4.5 Code of generate dictionary to store data

Step three: Use *mode\_dict* to generate DataFarme type of data. And the we may check it by print it.



Figure 4.6 Generated DataFrame data and print the result

Step four (optional): Serialization.

Create LabelEncoder object helps to do the serialization, which means transfer the non-numeric data into number data, in case of the data is not number. However, in this case, all the data is number. Thus, this step is not necessary, but it’s feasible, even if some of the data need to be changed to non-numeric data.



Figure 4.7 Code of data serialization

Step four: Generate decision tree by using sklearn.

Sklearn.tree provide decision model, which makes it’s easy to generate, only by selecting the criterion and put the data into the decision tree. Once created, it’s easy to test the predicted result as well.



Figure 4.8 Generate decision tree



Figure 4.9 Detail of generated decision tree



Figure 4.9 Test one predicted result

Step five: Visualize and save the decision tree.

By using sklearn, one line of code can do the visualization. And by using pydotplus, save the visualized decision into a PDF file.



Figure 4.10 Visualization and save the decision tree

The visualizaiton decision are shown below.



Figure 4.11 Visualized decision tree

Step six: Transfer the decision tree into JSON format.

Also by one line of code, it’s easy to transfer the decision tree into JSON format. And then copy this JSON string, paste it into the Arduino code.



Figure 4.11 Transfer the decision tree inot JSON format



Fugure 4.13 Part of the JSON decision tree

* + 1. Code in Zumo robot

First, define some useful sensors, including line sensors, proximity sensors and accelerometer. For the accelerometer, it’s controlled by LSM303 chip. Thus, initialize the LSM303 is equivalent to the accelerometer.



Figure 4.14 Initialize essential sensors

And the waitForButtonAndCountDown function is written to satisfy the sumo rule. When the reference starts the competition, press button A. And it’ll go for this function. Zumo robot then delays for 5 seconds, while the yellow led blink every second. After 5 seconds, the program goes for the rest of the program.

Then Allocate variables with different sensor values. Put the JSON format decision tree into the Arduino code. And input sensor values into the decision tree, get the output. The code is shown below.



Figure 4.15

According to the decision tree output value, assign its control mode. For example, if the label is ‘No object seeing’, then execute search() mode. If ‘Object detected’, then execute other modes. The code for object detected in Figure 4.16.



Figure 4.16 ‘Object detected’ corresponding control mode

And one of the control modes, like the turn() function, are shown below. It gets the value of direction (e.g. RIGHT or LEFT), and randomize. If it’s true, then the speed of the motor will remain uncertain when turning, which increased randomness of the robot’s movement.



Figure 4.17 *turn()* function in control mode

And the primary condition is to keep the robot remain inside the ring. So at the beginning of the loop function, it will first consider the value of the line sensor, then control the robot behaviour if necessary.



Figure 4.18 Code for not running out of the ring

* 1. GA based control system implementation

4.2.1 Technologies employed in this system

Same reason as the decision tree based control system, this is going to use object-oriented programming language. The whole GA based control system is going to implemented in the simulation environment, which is in the BEAST. And the BEAST is using C++, so this project in GA is going to use C++ as well.

The libraries are going to use in GA are in the table below. The libraries are quite different from the previous control system. Because it’s based on simulation, most of the libraries are the built-in head file in the simulation tool.

|  |  |  |
| --- | --- | --- |
| Table 4.4 | Library | Purpose |
| T 4.4.1 | C++ STL (Standard Template Library) | Basic function and type of data in C++ |
| T 4.4.2 | animat.h (BEAST built-in library) | Including Animat class and associated constants, which the agent shall extend this class |
| T 4.4.3 | sensor.h, sensorbased.h (BEAST built-in library) | Include some useful functions which set up the most common types of sensor |
| T 4.4.4 | beast.h (BEAST built-in library) | The main include file for the simulation environment |
| T 4.4.5 | simulation.h (BEAST built-in library) | The framework providing the facilities for implementing a range of different types of simulation |
| T 4.4.6 | geneticalgorithm.h (BEAST built-in library) | Contains GA process function, including selection, crossover, setting population group, etc. |
| T 4.4.7 | worldobject.h (BEAST built-in library) | Provide many overridable methods which ensure any kind of thing can be represented in the simulation environment, including the ring in sumo league |
| T 4.4.8 | import numpy | Store fitness score data temporary. Used for data processing |
| T 4.4.9 | Import matplotlib | Used for printouts of the plots of the fitness score. |

Table 4.4 Libraries and purposes in GA based control system

4.2.2 Create simulated environment and Zumo robot.

The sumo league environment is made of a big black ring and two wrestling robot. First all, create three class, represent the ring, one wrestling robot and another wrestling robot, respectively.

The ring class would extend the *WorldObject* class in *worldobject.h* file. Then set it’s size, location, and color.



Figure 4.19 Code for the ring object.

Then two agents class, *ZumoKing* and *ZumoQueen*, both initalize only with it’s basic attributes, including it’s sensor, location, orientation, size, etc. And these two agents class extend the EvoFFNAnimat class, which is in the animat.h file.



Figure 4.20 Code for the agent

Then, Figure 4.21 shows to implement these classes in the simulation class.



Figure 4.21 Code for implement classes into simulation class

After having these classes. The ring and robot in simulated environment must have the same size proportion as the proportion in real world. In real world, the diameter of the ring is approximate 72 cm, and the robot length and width are approximate 9 cm, the size proportion is 1:8. Therefore, set the radius of the ring is 200 in simulation environment, the radius of the robot is 25 then. (In the simulation, the agent is a round shape, while it’s square shape in real world. The area of the robot will be considered, but this shape difference will not consider in this project).

The real world and simulation environment comparison shown as below.



Figure 4.21 Comparison of real world and simulation

Then define the same sensor function as the sensor on the Zumo robot. Bascially, there are two kind of sensor, proximity sensor and line sensor.

The proximity sensor is a built-in function. All the thing need to be done is to add the sensor in the robot, define It’s scope, range and orientation. And these three inputs should be defined according to the real world as well. With the experiment in the real world, the maximum range is about 60 cm, the orientation is facing forward. As the measures in Figure 4.22, knowing the length of the isosceles triangle, it’s easy to calculate the angle with trigonometric function and inverse trigonometric function. And the result is approximately 45 degree, which is .

The line sensor is not a built-in function. However, some changes can make the built-in proximity sensor has the same function as the line sensor. First, create a small object in the center of the ring. It’s a dot with no collision volume. Then add one proximity sensor that detect nothing but this little dot. Keep returning the distance with the little dot, if the distance is exceed the radius of the ring, then this agent is out of the ring.

From Figure 4.22, it’s quite clear that a white dot in the middle of the ring, which is useful for the line sensor. The white transparent thing in the simulation is the detect range of all the sensors. From the code in Figure 4.21, there are one line sensor and two proximity sensors added in the agent, which simulated the Zumo robot. And the ring is also well simulated in BEAST.

The agent is controlled by FNN as metioned in Chapter 3. And both agent class is extend from EvoFFNAnimat class, so the control function is built-in. From Figure 4.20, it’s also initialised the FeedForwardNet (FFN) with five hidden nodes. And the number of inputs and outputs is decided by the number of sensors and the number of motors. Thus, the output of FFN controls the two motor and the movement of the agents. As for the control function, it’s not necessary to override it.



Figure 4.22 The sensor range comparison in real world and simulation

4.2.3 During the assessment

In each generation, the numer of assessment and the time for each assessment can be set. Figure 4.23 shows that the time of each assessment will set as 20 seconds, and five assessments in each generation. The population for each agents are five. Each assessment will be the 1 verus 1 competition, and in the next assessment will be another 2 agents play the game. The simulation class will make sure that the next two agents (one from ZumoKing, another from ZumoQueen) are picked.



Figure 4.23 Code for population and assesssment set

During the assessment, if one robot is fall out of the ring, then the code in Figure 4.23 will reset everything, including the position, speed, etc.

For the record, rule of *‘Upon pressing the start buttons, each robot must not move at all for five seconds’* are not considered in this GA based control system implementation. Which means, the robot wouldn’t wait for five seconds at the beginning of each assessment. It will save much time in each assessment.



Figure 4.23 Code for reset.

As for the *inRangeA* or *inRangeB*, these two variable are boolean type. And *counterA* and *counterB* are integer for count how many times each agent has run out of the ring separately. If *ZumoQueen* is out of the ring, then *inRangeB* is set to false, then *counterB* plus 1, then reset *inRangeB* to true. And do the same for *ZumoKing*.

By the end of one assessment comes to the GA.

4.2.3 GA process

Here comes the fitness score, selection, mutation and crossover. Most of the function has already been implemented in the *geneticalgorithm.h* file.

As for fitness score, this built-in function need to be override. The *ZumoKing* and *ZumoQueen* in this project have different fitness function. As for *ZumoKing*, the code of fitness is in Figure 4.24. The score is related to counterA and counterB, which are the number of times it run out of the ring and the number of times it’s opponenets run out of the ring. For ZumoKing, it’s reasonable that fitness score has positive correlation with counterA, while has negative correlation with counterB.

According to the rules, *‘However, if the robot does not move for a long time after the start command. It will be punished or judged for losing the round’* and *‘The robots fail to touch each other for some period of time, the reference may choose to restart a round’*. The fitness would encourage robot’s movement rather than stand still by introduce *DistanceTravelled*. Thus, *DistanceTravelled* would also has positive correlation with fitness score, but wouldn’t affect the score as much as *counterA* and *counterB.*

Because there are three selection choices, including roulette wheel selection, which is the probability selection according to fitness score. Thus, the fitness score should be positive rather than negative number.

In Figure 4.24, the ZumoKing would reward 5 times score for each time the opponent out of the ring. And it will be punished less for out of bounds itself. Because the DistanceTravelled is a large number and we don’t want it affect too much to the score, it’s reasonable to assign it with a small weight.

In some special cases, like if the score less than zero, then it’s seen as zero. And if both agents are not run out of the ring, which means no progress in the mathch. Then we seen it as passive behaviour in the game according to the rules. Thus, both agents will get zero for their fitness score, regradless their travelled distance score.



Figure 4.24 Fitness code for ZumoKing

The same fitness rules for ZumoQueen, but has slightly difference with ZumoKing. In Figure 4.25, the agent would reward for it’s opponenets out of bounds, while it will be punished 5 times more for out of the bounds itself. And the travel distance is the same as ZumoKing.



Figure 4.25 Fitness code for ZumoQueen

Also, the fitness scores of two agents will be stored in files seperately, which may provide data for drawing the plot.

Although, only slight difference are made for these two agents’ fitness, the expected behaviours can be very different. ZumoKing are expect to be more aggressive as well as offensive, and it will care more about attack and care less about keep it self inside the ring. In the opposite, ZumoQueen are expect to be more defensive, it’ll avoid run out of the ring and wait for the opponent to make mistakes. These two strategies will lead these two agents into different evolutionary process and different behaviours.

This is only one way of defining the fitness function, more way of calculating fitness score (with the same variables) will be examined in the next chapter.

For selection function, it’s a built-in function, it’s only need one line of code to implement it. There are three selection methods, including roulette wheel, rank, tournament selection. The selection code in Figure 4.26 is based on rank proportional selection. Other two selections are simular and will be experiment in the next chapter as well.



Figure 4.26 Code for selection

For crossover and mutation function, it’s also a built-in funtion. In Figure 4.27, it means the crossover probability is 0.7 and the mutation probability is 0.05.



Figure 4.27 Code for crossover and mutation

4.2.4 Plot the fitness score

As the code in Figure 4.24 and Figure 4.25, the fitness scores for both agents are stored in two files seperatly. I write a Python code for plotting it for better observe and analysis.

First, read the data stored file, then calculate the best and average fitness score in each generation. Then store the best and average score in the separate list, called *best* and *ave*. Plot these two lists. And set detail of the plot, like the linestyle, color, title, etc. Finally save the plot image.

This is not the main code, it’s more like the supporting role of the project. Thus, I won’t write details about this section, but only describe it concisely. And the codes are aviabile in the repositery as well.

# Chapter 5 Software Testing and Evaluation

5.1 Decision tree based control system testing

The decision tree generating progress is using the CRAT algorithm. It will calculate the Gini index of each feature. Smaller of Gini index, more purity of the feature.

As the visulalized decision tree in Figure 4.11. It’s pointed out the details of the result. In each root node, the Gini index are 0, which means the sample has been well devided according to features. Every root node is completed purity.

After generated and transfer the decision tree into the Zumo robot program. Compile and upload the program. Figure 5.1 shows that the code only takes 53% of the program storage space, and global variables use 25% of dynamic memory. Even with limited storge and memory, the program still leave much space, which leave the space for future work, like improve the control system by a more complexity decision tree, or the control mode can be more agile and hard to predicte it’s next move.



Figure 5.1 Storage and memory occupation for decision tree control system

As for it’s behaviours or control system judgement, it’s hard to be objective. The only fair way of judging the robot’s control system that I can think of, is according to the number of winning matches and total matches. If one robot winning the most of matches (assuming all of the participant robot has the same hardware and structure conditions) in the league, then it defineitly has the best control system. However, as it mentioned earlier, it’s hard to find the opponents or attend the sumo league during this COVID-19 global epidemic period. Thus, the best way that I can think of is using a cup controlled by hand as it’s imaginary enemy.

The testing experiments are from simple to difficult. First test, put the cup inside the Zumo robot’s sight, and the cup remain still (no hand control). This experiment shows that the Zumo robot can easily locate the cup and trying to push it out of the ring. Due to no sufficient contact surface, the Zumo robot fail to push it out of the ring even with the maximum horsepower. But it’ll work if the opponent is the same structured Zumo robot rather than a cup.

The second and third test, is to put the none controlled cup in the left and right hand side of the robot, respectively. And both experiment is to make sure that the robot wouldn’t see the cup in the first place. The robot will do the search mode first, then it will hit the boundary of the ring. Turn back, and continue search the cup. Once detected, trying to push it out of the ring. Both experiments results are fine, the Zumo robot will quickly find the cup and push it, while make sure it wouldn’t run out of the ring itself.

The forth test, using the hand controlled cup to compete with the Zumo robot. Because the Zumo motor is far too weak compare to human strength, I can defeat it with one finger. But to be fair, I have to keep my strength, trying to move the cup as the same speed as the robot. The robot doing quite well, it keep collide with the cup. Since the Zumo robot are trying it’s best to push the cup out of the ring, I let it win.

As the saying goes, talk is cheap, show me the code. Thus, not only the code are provided in the reposity, but also a video for decision tree based control system experiment are aviabile. This video including the four experiments, which are mentioned above. The link for the video are here. (URL)

5.2 GA based control system testing

In the simulation environment, the agents behaviour and fitness score can be observed. This section will discuss according to these.

And there are 4 experiments are done in this project, including three different selection methods and one special cases. All of these simulation experiments are recorded and edited in the video, the agents behaviours changes and the fitness score changes are aviabile in the video as well. (URL)

5.2.1 Tournament selection

The robot’s movement in the first few generations are random and make no sense. The robot just move round with different radius, and it will hit the opponent if it’s lucky. It’s just a initial start, so excute ‘High Speed Mode’ in BEAST to pass the few meanless generations.

Generation 18, the agents on the right hand side, which is the ZumoKing. One of the agent in ZumoKing population seems is able to locate the opponent and trying to push it. In these generations, ZumoKing clearly takes more advantages, while ZumoQueen is the underdog one.

However, the next many generations shows that ZumoKing population eliminated the agent with better fitness score and better behaviour. It’s fail to locate the opponents, and keep running out of the ring itself. And ZumoQueen gain more score because of it. The ZumoQueen just need to make sure that it won’t run out of ring itself, it’ll still get the high score.

As for the observed behavior, this experiment shows the counterexample of co-evolution. If one agent getting worse or fail to evolve, then it’s rivals won’t evolve as well.

As for the fitness score, the plot in Figure 5.2 and Figure 5.3 demonstrate the same thing. In the first 50 generation, ZumoKing win the most of the matches and get higher score. After 50 generation, it’s going downhill, and ZumoQueen begins to take advantages.

A few things that need to declare here. About the plot, you may see some bule lines in every plot. That’s just meanless lines, it doesn’t affect the observation and the result, so please ignore it. About the value on the y axis of the plot. The fitness is from 0 to 80 in ZumoKing, while it’s from 0 to around 8 in ZumoQueen. This difference is because of different fitness score calculation methods, so it’s meanless to compare these two values. The most important things is the trend of best and average score (the average score explain the trend better for the most of time) and the generations in each plot.



Figure 5.2 Tournament selection fitness score for ZumoKing



Figure 5.3 Tournament selection fitness score for ZumoQueen

5.2.2 Ranking selection

The first few generation are the same, the robot just move around randomly.

From 70th generation, the ZumoQueen begin to have some intelligence behaviour, which is locate the opponents, run towards it slowly then trying to push it out of the ring. Then ZumoQueen has domined the game until around 350th generation.

Around 420th generation, the ZumoKing gradually learned how to avoid the ZumoQueens attack, and let it run out of the ring itself. Thus, the ZumoKing begin to winning more matches.

Around 500th generation, the ZumoQueens are winning the game again. It’s behaviour become more straight forward. The most important thing is this strategy works well when it engage with it’s opponents current strategy.

From the plot in Figure 5.4 and Figure 5.5, it shows the same thing as the behaviour obversation. It’s also shows more information while we skip some of the generations. The ZumoQueen begin to rule the game until around 320th generation. Then the ZumoKing raises until around 540th generations. Then the ZumoQueen rule it again.



Figure 5.4 Rank selection fitness score for ZumoKing



Figure 5.4 Rank selection fitness score for ZumoQueen

5.2.3 Roulette wheel selection

The first few generation are the same.

When it comes to 57th generation, ZumoKing begin to win more matches, and it also evolved behaviour that can locate opponenet and trying to push it.

From 358th generation, ZumoQueen has developed strategy of avoiding ZumoKing’s attack, and it seems work. ZumoQueen begin to become the one with better behaviour. Especial in generation of around 440th, ZumoQueen barely lose a match, it’s main strategy is move to the back of the ZumoKing, then push it out of the ring.

Figure 5.5 and Figure 5.6 shows the plot of fitness changes of two agents. From the plot, it’s clear that ZumoKing are dominant the game from 50 to 300. Then it’s average fitness score are going down. From 400th generations to 550th generations, it’s going to the lowest point. After that, the fitness score for ZumoKing begin to raise again. As for ZumoQueen, it’s opposite to ZumoKing, it’s fitness line remian low when ZumoKing is high, and it’s going up when Zumo is going down.



Figure 5.5 Roulette wheel selection fitness score for ZumoKing



Figure 5.6 Roulette wheel selection fitness score for ZumoQueen

5.2.4 GA agent vs. traditionally controlled agent

This experiment is a little different from the experiment above. It would be interesting to find out if the GA agent start with randomly behaviour will beat an already implemented strategy agent without GA. The strategy for the agent without GA is to go round when no opponent detected, and speed up when opponent inside the sensor range.

As the experiments above with different selection. Roulette wheel selection has better performance. It’s speed of evolving are faster and the behaviour looks better compare with other two selection after same generations. Thus, in this experiment, the selection will go for roulette wheel selection.

In the first many generations, the GA agents are defeated most of the time, just as expected. Then in the 500th generations and later, it develop well strategy to cope with it’s enemy. And it’s defeat it’s opponents many times.

Figure 5.7, which is the plot for the fitness score, shows the same thing. The average score line goes up after iterations. And the best score are high and stable as well.



Figure 5.7 GA agent fight with traditionally controlled enemy

# List of References

Anon n.d. Autonomous Sumo Robot Rules - Robot Room. *www.robotroom.com*. [Online]. Available from: <https://www.robotroom.com/SumoRules.html#:~:text=Mini%2Dclass%20Sumo%20robots%20may%20be%2010%20centimeters%20(3.93%20inches>

Blanchon, B. 2020. bblanchon/ArduinoJson. *GitHub*. [Online]. [Accessed 16 August 2020]. Available from: https://github.com/bblanchon/ArduinoJson.

Erdem, H. 2007. A Practical Fuzzy Logic Controller for Sumo Robot Competition. *Springer-Verlag Berlin Heidelberg 2007*. (LNCS 4815), pp.217–225.

Erdem, H. 2011. Application of Neuro-Fuzzy Controller for Sumo Robot control. *Expert Systems with Applications*. **38**(8), pp.9752–9760.

‌ Erlan, C., Lima, O., Almeida De Araújo, F., Bibiano Da, M., Júnior, S., Edson, A., Filho, R., De Andrade, R., Rabêlo, L., Allisson, T., Da Silva, R., Jose, A. and Alves, O. 2013. *An Enhancement in Conventional Potential Field Using a Fuzzy System for Navigation of a Sumo Robot*.

Lehner, J., Dornberger, R., Simić, R. and Hanne, T. 2019. *Optimization of Multi-Robot Sumo Fight Simulation by a Genetic Algorithm to Identify Dominant Robot Capabilities*.

Wikipedia Contributors 2019. Decision tree learning. *Wikipedia*. [Online]. Available from: https://en.wikipedia.org/wiki/Decision\_tree\_learning.

Wilson, B., Author, S., Germann, T. and Al-Olimat, K. 2016. *Sumo Robot Competition* [Online]. [Accessed 4 August 2020]. Available from: http://people.cst.cmich.edu/yelam1k/asee/proceedings/2016/student\_regular\_papers/2016\_asee\_ncs\_paper\_58.pdf.

# Appendix A External Materials

<Level 1 Heading with ‘heading 1’ Style Applied by Pressing Ctrl Shift 1> Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading.

‘High Speed Mode’ is just like the ‘fast-forward’ key. The robot’s movement and details in each generations won’t be shown, only the result are displayed, including average fitness and best fitness in each generation for each agent. As Figure 5.2 shows, BEAST is in the ‘High Speed Mode’ and the result is in the ‘Log Window’ is on the right hand side.

Khaleesi seems back to Westeros with her dragons, (video)

All of the experiments shows the co-evolution of these two agents.

In the aspect of behaviour, if one agents develop it’s intelligence behaviour first, then it would winning more game. As for it’s opponenets, it will evolve it’s strategy according to the winning one, and after hundred of generations, it’s losing situation will change. However, if one agent keep making mistake, or it’s cant out of local optimal. It’s opponent will not going to evolve better strategy as well. As the Chinese confucian philosopher Mencius said, one prospers in worries and hardships and perishes in ease and comfort. In this case, both agents wouldn’t develop good strategy.

In the aspect of fitness score.

(conclusion)

## A.1 Level 2 Heading with ‘heading 2’ Style Applied by Pressing Ctrl Shift 2

Text under level 2 heading. Text under level 2 heading. Text under level 2 heading. Text under level 2 heading.

### A.1.1 Level 3 Heading with ‘heading 3’ Style Applied by Pressing Ctrl Shift 3

Text under level 3 heading. Text under level 3 heading. Text under level 3 heading. Text under level 3 heading.

#### A.1.1.1 Level 4 Heading with ‘heading 4’ Style Applied by Pressing Ctrl Shift 4

Text under level 4 heading. Text under level 4 heading. Text under level 4 heading. Text under level 4 heading.

# Appendix B Ethical Issues Addressed

Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading. Text under appendix heading.

## B.1 Level 2 Heading

Text under level 2 heading. Text under level 2 heading. Text under level 2 heading. Text under level 2 heading.