Evolving Fault Tolerant Robotic Controllers

Yuyuan Zhang

PhD

University of York

Electronic Engineering

April 2018

Abstract

Fault tolerant control and evolutionary algorithms are two different research areas. However with the development of artificial intelligence, evolutionary algorithms have demonstrated competitive performance compared to traditional approaches for the optimisation task. For this reason, the combination of fault tolerant control and evolutionary algorithms has become a new research topic with the evolving of controllers so as to achieve different fault tolerant control schemes.

However most of the controller evolution tasks are based on the optimisation of controller parameters so as to achieve the fault tolerant control, so structure optimisation based evolutionary algorithm approaches have not been investigated as the same level as parameter optimisation approaches. For this reason, this thesis investigates whether structure optimisation based evolutionary algorithm approaches could be implemented into a robot sensor fault tolerant control scheme based on the phototaxis task in addition to just parameter optimisation, and explores whether controller structure optimisation could demonstrate potential benefit in a greater degree than just controller parameter optimisation.

This thesis presents a new multi-objective optimisation algorithm in the structure optimisation level called Multi-objective Cartesian Genetic Programming, which is created based on Cartesian Genetic Programming and Non-dominated Sorting Genetic Algorithm 2, in terms of NeuroEvolution based robotic controller optimisation. In order to solve two main problems during the algorithm development, this thesis investigates the benefit of genetic redundancy as well as preserving neutral genetic drift in order to solve the random neighbour pick problem during crowding fill for survival selection and investigates how hyper-volume indicator is employed to measure the multi-objective optimisation algorithm performance in order to assess the convergence for Multi-objective Cartesian Genetic Programming.

Furthermore, this thesis compares Multi-objective Cartesian Genetic Programming with Non-dominated Sorting Genetic Algorithm 2 for their evolution performance and investigates how Multi-objective Cartesian Genetic Programming could be performing for a more difficult fault tolerant control scenario besides the basic one, which further demonstrates the benefit of utilising structure optimisation based evolutionary algorithm approach for robotic fault tolerant control.

List of contents

List of tables

List of figures

Acknowledgements

Firstly, I would like to thank my parents Mr Chaoyi Zhang and Mrs Xiaodong Zhang for their support to let me finish my PhD course in the University of York. Especially, I would like to thank my wife Xiaotong Huang for her company with me during our life in UK.

Secondly, I would like to thank my supervisors Prof Jon Timmis and Dr Andy Pomfret as well as my internal examiner Prof Steve Smith for their help during my PhD study. In addition, I would like to thank our department graduate administrator Camilla Danese who helped me a lot whenever I got a problem during my work in our department.

And then, I would like to thank my colleagues Guangsha Xu, Danesh Tarapore, Richard Redpath, Alan Millard and other people, who helped me a lot when I got stuck in my research.

Finally, I would like to thank all my friends I have ever made during these four years' time in UK.

Thank you all.

Declaration

I declare that this thesis is a presentation of original work, which I undertook at the University of York during 2014 - 2018 and I am the sole author. This work has not previously been presented for an award at this, or any other, University.

Chapter 1 Introduction

1.1 Motivation

Fault tolerant control and evolutionary algorithms (EA) are two different research areas, yet have a natural synergy. With the development of artificial intelligence, EA has demonstrated competitive capability compared to traditional approaches for optimisation problems. In this case, the combination of fault tolerant control and EA shows great potential, with the ability to evolve new solutions that have the ability to adapt over time, and have greater potential for robustness to failure.

Typically, fault tolerant control based approaches employ EA to optimise the controller parameters for a given set of scenarios. However, the controller's structure usually remains fixed when parameters are being optimised. Although the parameter optimisation based EA approaches have demonstrated effective performance for fault tolerant control, work in this thesis considers optimising the controller structure, in addition to the parameter space, with a view to observing a greater degree of fault tolerance.

1.2 Thesis contributions

The research question that this thesis aims to investigate is: "how can structure optimisation based EA approaches be utilised to evolve, at a structural level, fault tolerant robotic controllers?" In order to answer the research question, some main contributions are made in the thesis, which are:

- The review of literatures in fault tolerant control with structure optimisation based EA approaches and Cartesian Genetic Programming of Artificial Neural Networks is identified as the best suited controller structure optimisation approach used for designing a robot fault tolerant control system
- The investigation of how Cartesian Genetic Programming of Artificial Neural Networks could be utilised to design a robust fault tolerant control system
- The review of survival selection along with population diversity and the investigation of how it could be utilised to improve the crowding fill strategy for Multi-objective Cartesian Genetic Programming of Artificial Neural Networks

- The review of how hyper-volume indicator is used for performance measure and the investigation of how it could be utilised to assess the convergence for Multi-objective Cartesian Genetic Programming of Artificial Neural Networks
- The development of a complete library of Multi-objective Cartesian Genetic Programming of Artificial Neural Networks based on a new crowding fill strategy and the investigation of how it could be utilised instead of single objective optimisation to obtain a Pareto set of controllers used for the design of a robust as well as switched fault tolerant control system
- The investigation of how Non-dominated Sorting Genetic Algorithm 2 could be utilised to design a robust as well as switched fault tolerant control system based on multi-objective controller parameter optimisation
- The comparison between Multi-objective Cartesian Genetic Programming of Artificial Neural Networks and Non-dominated Sorting Genetic Algorithm 2 for controller evolution in order to investigate the difference between controller structure optimisation and controller parameter optimisation
- The investigation of how Multi-objective Cartesian Genetic Programming of Artificial Neural Networks could be utilised to design a robust as well as switched fault tolerant control system based on multi-objective controller structure optimisation for a more difficult fault tolerance scenario

1.3 Thesis outline

This section gives an outline of each chapter for the remaining thesis summarised as below:

- Chapter 2 reviews fault tolerant control and different structure optimisation based evolutionary algorithms along with artificial neural networks in order to find out a suitable approach to design a fault tolerant control system. Moreover, different multi-objective optimisations are also reviewed and survival selection based on crowding measure is also mentioned along with population diversity. Finally, convergence criteria and statistics analysis are both introduced.
- Chapter 3 presents how Cartesian Genetic Programming of Artificial Neural Networks, which is the approach obtained in chapter 2, is utilised to achieve the robust fault tolerant control.

- Chapter 4 demonstrates how genetic redundancy and crowding measure along with hyper-volume indicator are utilised to develop the library of Multi-objective Cartesian Genetic Programming of Artificial Neural Networks and displays how it could be utilised to achieve both of robust and switched fault tolerant control.
- Chapter 5 shows how Non-dominated Sorting Genetic Algorithm 2 could be utilised to evolve feasible controllers so as to achieve both of robust and switched fault tolerant control and presents how it is compared with Multi-objective Cartesian Genetic Programming of Artificial Neural Networks for the evolution experiment performance.
- Chapter 6 presents how Multi-objective Cartesian Genetic Programming of Artificial Neural Networks is performed to achieve a more difficult fault tolerant control scenario for both of robust and switched fault tolerant control.
- Chapter 7 gives a summary about the thesis and the proposed future work.

Chapter 2 Literature review

2.1 Introduction

The aim of this thesis is to fill the research gap that controller structure optimisation based EA approach has not been investigated as the same level as controller parameter optimisation for fault tolerant control. In this case, the thesis will explore how controller structure optimisation could be utilised to design a fault tolerant control system. For this reason, this chapter will review the area of fault tolerant control firstly and then review how different structure optimisation based EA approaches have been performed in the controller structure optimisation tasks. This chapter will also estimate the respective benefit and drawback for different structure optimisation based EA approaches along with an investigation of artificial neural network for the controller type in order to find out the most suited approach to be utilised for the design of a fault tolerant control system.

2.2 Fault tolerant control

Faults in automated processes will usually cause undesired results especially the shut-down of controlled plants. These consequences could be harmful to the plant, to personnel or the environment. In this case, fault tolerant control was developed which is used to increase the plant availability and reduce the risk of safety hazards so as to avoid a simple fault becoming a serious failure [1].

Fault tolerant control can be classified into two aspects: passive or active [2]. Passive fault tolerant control uses a specific fixed controller to be robust against certain faults [3]. And active fault tolerant control redesigns the control system in order to maintain an acceptable performance after a fault occurs [4]. In active fault tolerant control, [2] indicates two necessary tasks: fault detection and isolation and fault accommodation or controller reconfiguration. Fault detection and isolation consist of a fault diagnosis scheme and fault accommodation or controller reconfiguration can be regarded as controller redesign [5]. Active fault tolerant control has more fault tolerant capabilities than passive fault tolerant control just equipped with a robust controller [6]. Because there will be more solutions to cover more classes of faults if the controller can be changed [7].

As for the controller redesign in active fault tolerant control, fault accommodation means that the dynamic structure and parameters of the controller will change to accommodate the fault, but the relationship between controller and plant still maintains fixed including the reference signal and control value. So the fault can be accommodated only if the controller has a solution to deal with the faulty system [4]. Although fault accommodation can be quick to find a suitable controller in order to realize some hard real time constraints [5], the controllers need to be pre-designed for all the possible types of faults. So the fault accommodation cannot work well if no solution is found among the controllers especially the relationship between controller and plant needs to be adjusted if a degraded performance has to be accepted in some cases. On the other hand, controller reconfiguration will establish a new control loop including a reconfigured controller with the introduction of alternative input and output signals between the controller and the plant [7]. In this sense, the controller can be reconfigured online to achieve the performance of different faulty systems including some degraded performance. However, the controller reconfiguration emphasizes the parameter reconfiguration based on some optimization techniques [2], so the research of controller structure reconfiguration is still in an early stage. Although the evolution of controller structure has been studied, this research field hasn't been put into the fault tolerant control scheme. Therefore the hypothesis of this work can be described that the fault tolerant control can work better if the controller structure evolution is associated with the controller reconfiguration.

2.2.1 Passive fault tolerant control

In the field of passive fault tolerant control, the robust control is the main approach [2]. It designs the controller with constant parameters as well as the structure to correct a specific fault so as to guarantee the required performance [7]. And the control objectives of robust control mainly include the following fields: stability, disturbance rejection and noise rejection [8]. Typically the most effective way of robust control is to cope with the faults which can be modelled as plant uncertainties [7]. For example, [9] designs a robust control system against the plant uncertainty. This work belongs to a kind of model following control which uses a correction mechanism to cope with the deviations between the real plant and the reference model to achieve the reference tracking task. The reference model reflects the expected performance of the plant and the correction mechanism is used to force the plant to follow the model. However due to the parameter variations or system disturbance, the uncertainty is always a problem occurred in the real plant. So the correction scheme is designed equivalently as a controller to control the plant in the worst case of uncertainty [9]. This work is a typical example to apply the robust control scheme to cope with the system uncertainty. So the effectiveness of the passive fault tolerant control emphasizes on the robustness of control system against certain faults as well as the disturbance and noise in the system with fixed controllers. However, this approach has limited fault tolerant capabilities with just robust controllers [6]. Therefore if the controller can be changed, there will be more solutions to cover more classes of faults compared to the passive approaches [7]. So that’s why the active fault tolerant control was developed.

2.2.2 Active fault tolerant control

In the research area of active fault tolerant control, [5]mentions two tasks: fault diagnosis and controller redesign. Fault diagnosis means an early detection, isolation and also identification of faults. And controller redesign needs to be performed after the fault is diagnosed to achieve fault tolerant control. Controller redesign contains two main approaches: fault accommodation and controller reconfiguration which are respectively shown in Figure 2.1 and Figure 2.2.

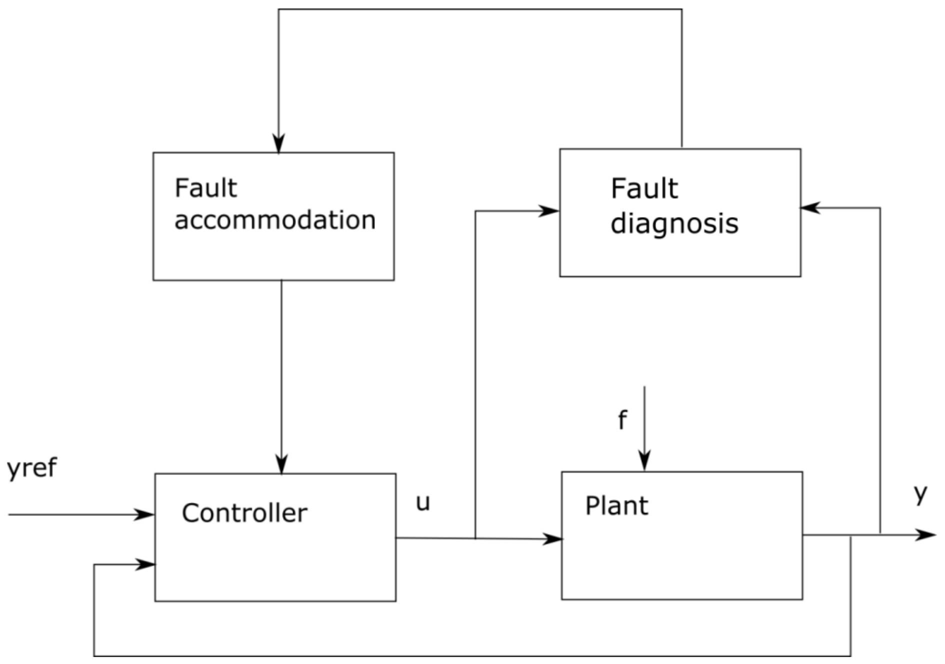

Figure 2.1: Fault accommodation (from [5]*)*

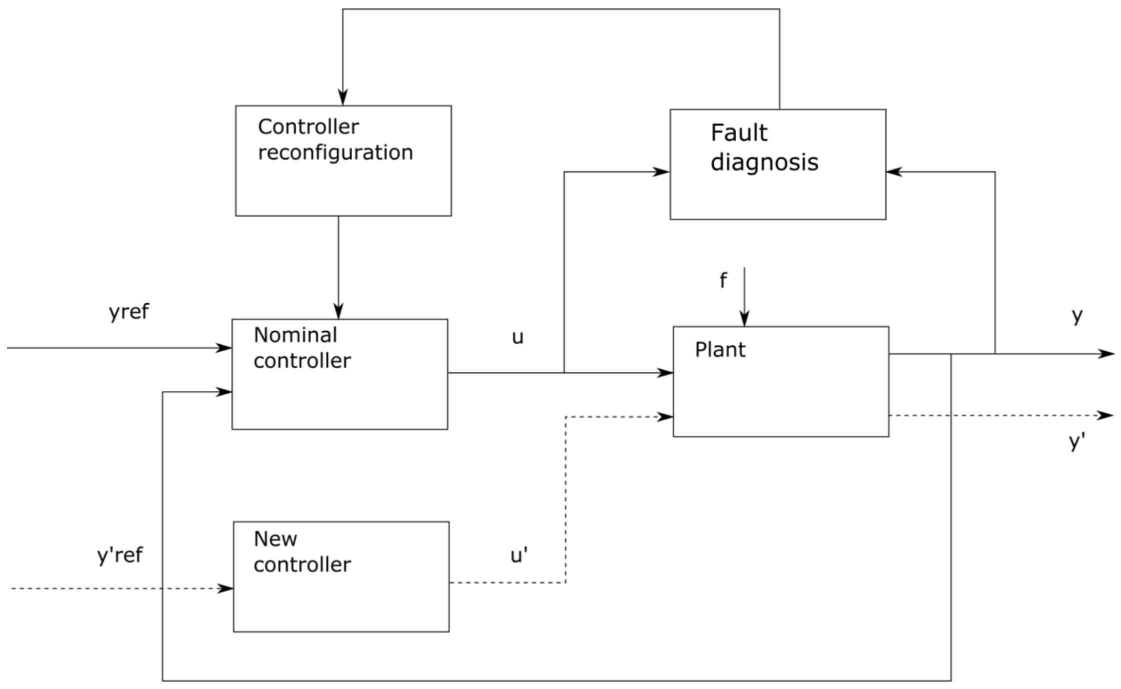

Figure 2.2: Controller reconfiguration (from [5]*)*

In these two figures, f is the fault, y_{ref} is the reference input, u is the control value, y is the system output, y'_{ref}, u' and y' are the corresponding new signals. These two approaches both need to change the parameters and structures of controllers to avoid the consequences of faults. However the difference is that controller reconfiguration needs to change the input and output signals between the controller and system so that a new control loop will be generated. But the fault accommodation maintains the same values for all the signals [5].

- Fault accommodation

In fault accommodation, one of the representative approaches is the switched control. It is based on the bank of controllers designed for the normal and different faulty systems [5]. The pre-designed controllers are generated offline to process different types of faults. So their internal structures may be different, but the I/O signals will remain the same to achieve accommodation [7]. Therefore it is a switching mechanism that a suitable controller needs to be selected in terms of the type of fault. The benefit of fault accommodation is that it can be quick to find a suitable controller so that some strong real time constraints could be realized [5]. However this approach needs to pre-design the controllers for all the possible types of faults. If none of the pre-designed controllers is available to deal with a typical fault, the required performance cannot be achieved.

- Controller reconfiguration

In controller reconfiguration, a new control loop is established with the introduction of alternative input and output signals between the controller and the system [7]. This approach could be applied when a fault is occurred in the system sensor or actuator. In this sense, a new control loop with a new controller and alternative signals needs to be established when alternative components are introduced [5]. This approach is able to process unplanned faults by changing the new control objectives and constraints, so a new control loop is also required. However designing a new control system based on a new control loop is definitely not an instant work, so the controller reconfiguration would be more suited to the tasks where sufficient time is allowed to designing a new control system during the system operation.

As can be seen from these two approaches, fault accommodation and controller reconfiguration have their own benefits and drawbacks. Actually fault accommodation refers to the offline designing controllers where the controllers need to be designed well before loaded to the real system. However controller reconfiguration always refers to the online designing controllers where the controllers are being designed during the system operation. In this case, the fault accommodation can guarantee that the controller will be working well since it was well designed offline. However controller reconfiguration cannot ensure when the controller design is finished before loaded to the system in order to avoid a crashed system. On the other hand, fault accommodation has to design all the possible types of controllers offline, if a unplanned fault occurs online, there is no way to tolerate this fault. However, controller reconfiguration is capable to deal with all the possible types of faults including unplanned one as long as the fault can be diagnosed. In conclusion, fault accommodation and controller reconfiguration both have benefits and drawbacks. Therefore, which one to be utilised for fault tolerant control is dependent on the difficulty of the given task including the passive fault tolerant control approaches.

2.3 Evolutionary algorithms in controller structure optimisation

2.3.1 Introduction of evolutionary algorithms

EA is a kind of optimization algorithms in the artificial intelligence area which was developed based on the inspiration of natural selection and survival of the fittest in Darwinian evolution [10] [11]. Generally speaking, there are several steps to constitute a complete evolution loop. Firstly, the initial population needs to be created randomly as the first generation. Secondly, this population needs to be evaluated for the given problem and their performance is recorded as fitness values where the given problem is normally called fitness function. After that, this population needs to be selected based on the fitness value and the selected parents will be utilised to create their children for the next generation based on genetic operator including crossover and mutation. And when the children are obtained, they also need to be evaluated based on the fitness function. Now it comes to the crucial step called survival selection. In the survival selection, one option is just utilising the children as the next generation, which is easy and straight forward for many EA applications. The other will compare the obtained children with their parents. If children's fitness is not better than the parent, the parents will be directly copied into the next generation without any change, which is also called elitism strategy. However whether elitism is required depends on the given task since elitism will not always be the suited idea to obtain the new blood for the next generation. Nevertheless, one significant benefit of elitism is that it always guarantees the next generation to be at least performing equivalent as the last generation, which is convenient for convergence observation and helps to achieve a better convergence especially for multi-objective EA (MOEA) [12]. The above is a whole evolution loop and EA will only stop when termination condition is met such as the target fitness value is obtained, the maximum generation number is reached or the convergence criteria is realised [10] [11].

In terms of fault tolerant control, genetic algorithm (GA) based approaches have been investigated extensively. GA is used as an optimization tool that the task is normally about how to optimize the parameters of a controller to deal with different types of faults [2]. For example, [13] designs a fault tolerant control system for an active magnetic bearing task using a multi-objective GA. In this work, the active magnetic bearing system is used to tolerate the faults occurred in a coil or an amplifier in a machine. To design an active magnetic bearing system, PID controller is applied with multi-objective GA to tune the parameter of the PID controller to achieve different configuration of this active magnetic bearing system.

This work shows a typical example of using GA as an effective approach to tune the parameters of controllers to achieve the fault tolerant control. However, GA is just one of the simplest EA which can be only used for the parameter reconfiguration so that the controller structure always maintains fixed. If the controller structure could also be changed, more solutions might be generated to deal with more types of faults. However controller structure optimisation hasn't been developed as the same level as controller parameter optimisation in the fault tolerant control area and that's why the combination of controller structure optimisation with fault tolerant control would be a new research topic. On the other hand, active fault tolerant control needs rigorous identification of all classes of faults so that the controller redesign could be carried out [14]. Therefore the controller structure configuration could also be a promising approach to deal with a wider range of faults as long as the fault could be diagnosed. For this reason, there are some other EA approaches which were developed to work for the optimisation of the structure as well as the parameters. Those structure optimization based EA approaches are reviewed in the following sections.

2.3.2 Genetic programming

Genetic programming (GP) is a kind of structure optimisation based evolutionary algorithms (EA) approach which is normally used to automatically create a computer program to solve a problem using program trees [15]. GP was firstly introduced in [16] based on the parse trees as the genome encoding in order to create programs. In this kind of tree based GP, the computer programs are created in tree structures where a tree node is an operator such as [+, -, *, /] and the terminal node is a variable such as [a, b, c, d]. Based on this tree structure, the programs will be evaluated for each generation and the evolution will be finally terminated when an acceptable program expression is found. In this case, Lisp became the first programming language applied to this tree based GP since Lisp is also expressed in a tree structure that matches the genotype of this tree based GP. In terms of the genetic operator, there are two different types applied for the mutation including the point mutation and sub-tree mutation. Point mutation randomly changes the functions or terminals of a proportion of the nodes within a parse tree and the number of nodes are determined by the mutation rate. Sub-tree mutation randomly changes the whole sub-tree to a new one with randomly selected functions and terminals. On the other hand, sub-tree crossover is the only type for crossover which creates two children with the swapped two sub-trees from the selected two parents [16]. An example of tree based GP genotype is shown in Figure 2.3.

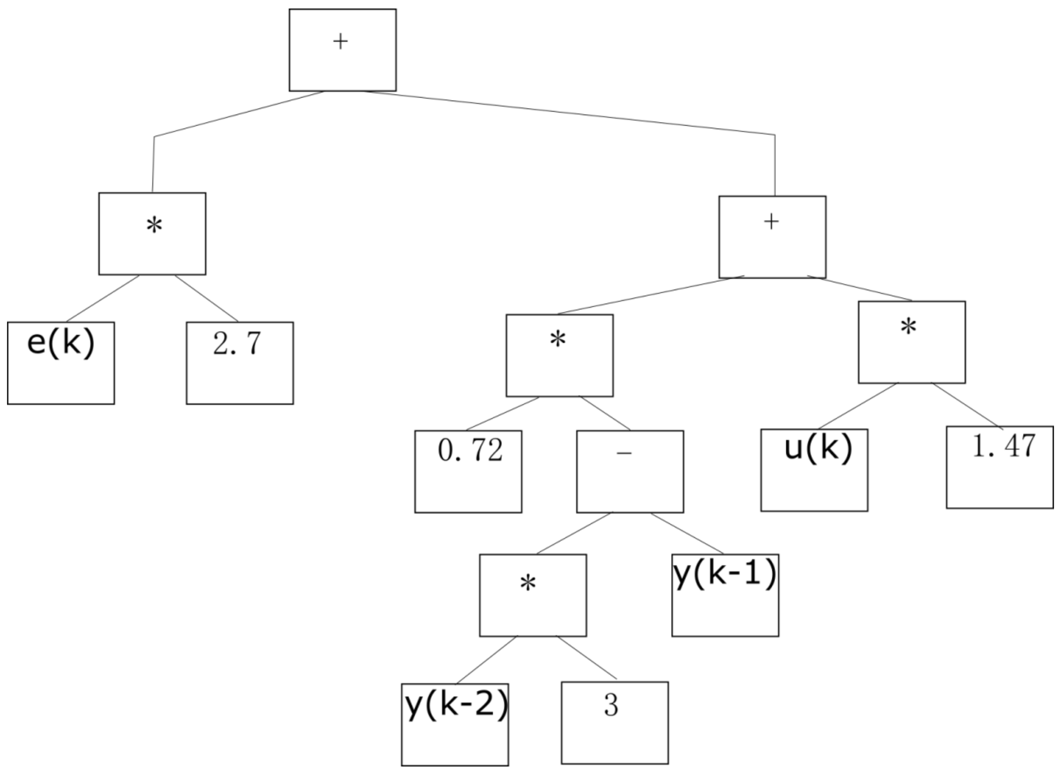

Figure 2.3: An example of GP genotype [17]

Besides the basic approach of GP to write a computer program, the program trees can be also interpreted to construct a complex structure, such as an electrical circuit [18]. Moreover, the program trees could be interpreted to represent the block diagram of a controller so as to achieve the controller evolution [15]. In this research field, several related works are reviewed as following including control system design and robotic controller design based on GP. Among these works, [17] presents a typical implement of how to use GP to evolve a controller so as to design a control system, so this work will be described in more details.

- GP for control system design

[17] considers a simple feedback control loop to be used for controller evolution which is shown in Figure 2.4. In this control loop, the process is a continuous time dynamic system, the controller is also a dynamic system with unknown structure and parameters, y is the controlled variable, r is the reference variable, u is the control variable, e is the control error.

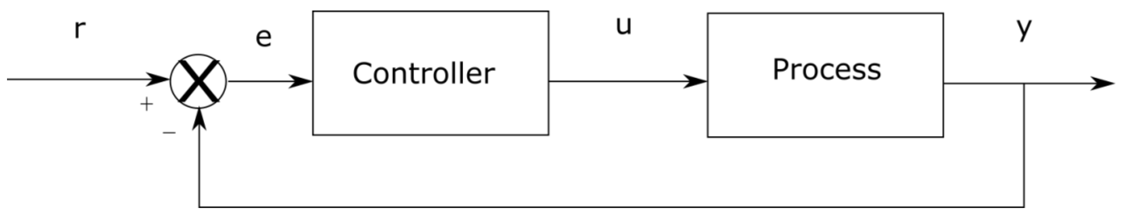

Figure 2.4: A simple feedback loop [17]

In this case, a simple integral performance index is chosen as the cost function which is defined in equation 2.1 where T is the simulation time and $\dot{y}$ is the controlled variable derivative.

$$J=\int_0^T |e(t)|dt+\alpha\int_0^T |\dot{y}|dt \tag{2.1}$$

The aim of controller design is actually an optimization task which searches for a controller so that the chosen performance index could be minimized [17]. This cost function consists of two parts. One is a basic integral absolute error (IAE) form which integrates the absolute error over time in order to minimize it. The other is described in a form of integral absolute output derivative multiplied by a coefficient. It could be used to minimize the output slope over time so that the output trajectory could become smoother with an appropriate choice of the coefficient.

To demonstrate the performance of GP, two different case studies are implemented in this work including a continuous time and a discrete time controllers design. The first test case uses a continuous time interconnected network to describe the control algorithm with a table based representation of individuals which is different from classical tree based representation in GP. The function blocks include integrator, derivative unit, amplifier (multiplication by a constant) and summation/multiplication unit. The objective is to find an optimal controller network with these function blocks which minimizes the above cost function. The crossover used here will exchange the corresponding parts of two random positions between two columns of the table. And mutation will change the type of block or delete and add a block or change the value of a constant [17]. In the second test case, a discrete time recurrent control algorithm is designed with a classical tree representation of genotype. The crossover exchanges the randomly selected sub- trees from two trees and the mutation replaces a randomly selected sub-tree by another one. To demonstrate the effectiveness of using GP for controller design, two linear dynamic systems are implemented, which are shown in transfer function 2.2 and 2.3.

$$G(s) = \frac{s^2 + 3s + 1}{4s^{5+}8s^4 + 10s^3 + 6s^2 + 3s + 1} \tag{2.2}$$

$$G(s) = \frac{7s + 1}{s^3(2s^3 + 14s^2 + 30s + 18)} \tag{2.3}$$

The optimization results and algorithm running time of obtained controllers of two linear dynamic systems are demonstrated in Table 2.1 compared with the results of a PID controller tuned by GA for the first system. GP1 means the table based continuous time controller, GP2 means the tree based discrete time controller and GA PID means GA based PID controller.

Table 2.1: The obtained results for two different linear dynamic systems [17].

Experiment 1	Cost function value	Time
GA PID	11268	3h55min
GP 1	3950	18h57min
GP 2	12265	5h53min
Experiment 2	Cost function value	Time
GP 1	6509	20h27min
GP 2	19646	5h24min

As can be seen from the Table 2.1, GA PID has the fastest running time but high cost function values. GP1 achieves the lowest cost function values but much slower running speed. And GP2 has similar results compared to GA PID and higher cost function values and faster running time compared to GP1. So the table form based GP method could be a promising approach due to its obvious benefit of lowest cost function values compared to GA tuned PID controller. However the running time of this approach is much longer than the other two approaches and this issue needs to be improved. According to the performance index, GP1 obtains the best dynamic performance which has the shortest rise time and settling time with no overshoot for the first system. In terms of the second system, GP1 is also better than GP2 with slightly faster rise time, shorter settling time and lower overshoot.

As can be seen from this work, GP is capable to find acceptable solutions for controller design based on the feedback closed loop, which outperforms the GA based PID controller. Furthermore, table based GP also produces better performance than classic tree based GP, which indicates that the tree based GP may not be a first choice depending on the given task in spite of a shorter running time. Finally, this work also notes that GP can be used to design the controller with complex systems, but the only limitation is the high requirement of computation time, which may be a common issue for GP based approaches.

- Improvement of GP based control system design

Besides a description of how to use GP to construct the controller [17], there are also some approaches to improve the performance of GP based controller design. According to [19], GP can be used to construct a discrete recursive feedback control law using the equation 2.4.

$$\mathrm{u(k) = u(k-1) + [output\ of\ individual\ tree]} \tag{2.4}$$

For a population size of *M* the output of the *i*th {*i*=1,2,3…M} controller at time *k* is equal to the output at time step *k-1* plus some correction term applied by the *i*th GP individual. The fitness function is shown in equation 2.5. This is calculated using P independent and randomly generated set point changes: ΔS_j{j=1,2,3…P}.

$$\mathrm{F_i} = \sum_{\mathrm{j=1}}^{\mathrm{p}} \frac{\sum_{\mathrm{k=0}}^{\mathrm{n}} \mathrm{k|e(k)| + r\Delta u^2(k)}}{\Delta \mathrm{s_j}} \tag{2.5}$$

In 2.5, n is the number of discrete time steps which is decided by trial and error before GP runs. k|e(k)| is the integral time absolute error (ITAE) term and $\mathrm{r}\Delta u^2(k)$ is a weighted penalty term for excessive control effort u(k) with a constant r determined by trial and error [19]. This fitness function minimizes two aspects of the controller performance which are the error and the controller output. Although the output slope is not minimized here which is mentioned in [17], the excessive control effort could be decreased in this case. To demonstrate the effectiveness of this approach, [19] uses two chemical processes for controller design including a constrained second order ARX(auto-regressive exogenous) process and a non-linear CSTR(continuous stirred tank reactor) process. The ARX process is defined in equation 2.6. And the non-linear dynamic model of CSTR is referred from [20].

$$\mathrm{y(k)} = 0.99\mathrm{y(k-1)} - 0.05\mathrm{y(k-2)} + 0.1\mathrm{u(k-1)} + 0.01\mathrm{u(k-2)} \tag{2.6}$$

As can be seen from the ARX process response comparison, the evolved controller has longer rise time than the PID controller but without any overshoot. While for the settling time, they have similar performance. And according to the CSTR process response comparison, they both perform similarly just the evolved controller has slightly larger overshoot. Therefore, GP is capable of producing dynamic recursive controllers which provide similar performance compared with PID controllers [19]. Hence the concept of recursive feedback control law used in GP could be applied to the controller design in the discrete time domain. Although its performance is similar to PID controller, it is still an encouraging idea to use GP for the discrete controller design.

[21] also improves the performance of GP based controller design by creating a controller with a free variable. The reason to introduce a free variable in the controller design is that the evolved controller could control an entire category of plants through modifying the value of the free variable instead of a particular plant with a fixed variable. The tree format is used to present the controller. A three-lag plant is used for the controller design and the controller contains a free variable representing the plant time constant τ. This free variable can be changed among 0.1, 0.3, 1.0, 3.0 and 10.0 which are defined in this work. In this sense, the evolved controller becomes a function of this free variable which corresponds to the plant time constant. The transfer function of this three-lag plant is defined in equation 2.7 where K is the plant's internal gain(tested by values of 1.0 and 2.0) [21].

$$\mathrm{G(s)} = \frac{\mathrm{K}}{(1+\tau\mathrm{s})^3} \tag{2.7}$$

The fitness is measured by means of 42 separate fitness measurements. Among these 42 fitness measurements, the first 40 are based on a modified integral of time-weighted absolute error (ITAE) which is shown in equation 2.8 where e(t) is the error; τ is the externally supplied value of the time constant; B is a constant; A is an additional weight value which varies depending on the error so that unacceptable overshoot could be avoided; and finally each integral value needs to be divided by τ^2 so as to equalize the influence of five different values of τ. The 41st One is in frequency domain which constrains the frequency of the control value to avoid extreme high frequencies applied into the plant. The last one is also in frequency domain measuring the effect of sensor noise.

$$F = \frac{\int_{t=0}^{10\tau} t|e(t)|A\big(e(t)\big)B dt}{\tau^2} \tag{2.8}$$

This obtained controller is compared with the Astrom and Hagglund controller which is a PID controller tuned with a new simple tuning rule by Astrom and Hagglund [22].

As can be seen from the result, [21] calculates that the controller created by genetic programming is better than 3.69 times as effective as the Astrom and Hagglund controller as measured by the integral of the time-weighted absolute error(ITAE), has only 57% by the rise time, and has only 55% by the settling time. Moreover, the genetically evolved controller is more robust to the disturbance than Astrom and Hagglund controller indicated from the disturbance sensitivity. The computation time to find the best of run evolved controller is 23.43 hours. The conclusion demonstrates that GP can be used to create a controller with a free variable which outperforms Astrom and Hagglund controller [21]. Therefore the evolution of robot controller could be referred to this approach using a free variable in the controller design. Although this approach has much better performance than the Astrom and Hagglund controller, its computation time of 23.43 hours is still high.

- GP based robust controller design

GP can be also used to construct a robust controller. [23] applies GP to construct a robust flight controller against the wind shear. The occurrence of strong downbursts could cause serious crashes of landing aircrafts. So the problem is how to construct a robust flight controller with GP to make the aircraft land along the reference trajectory: $\mathrm{h} = -0.0352\mathrm{x} + 225.0$ in the case of wind shear. The performance of the generated controller is illustrated using the aircraft trajectories in terms of different sizes of wind shear. The result shows that the GP based robust controller could achieve effective performance for aircraft to be landed safely in spite of different sizes of wind shear. Therefore this work describes another application of GP in the robust controller design, and the results show that GP is able to get effective solutions.

- GP for tuning controller parameters

[24] describes the application of GP to tune a controller parameters. In this work, GP is used to construct a self-evolved Model Reference Adaptive System (MRAS) which is designed for a second order system based on a pre-defined reference model. MRAS is one of the adaptive controllers, its performance is described through a reference model which gives the desired response to a reference signal [25]. The aim of this work is to evolve a suitable controller which is based on the desired model to control a process. Actually this work applies GP to automatically tune the controller to meet the desired performance. Because the structure of the controller is already given, so the work of GP is to provide the correct controller parameters [24]. The block diagram of this work is shown in Figure 2.5.

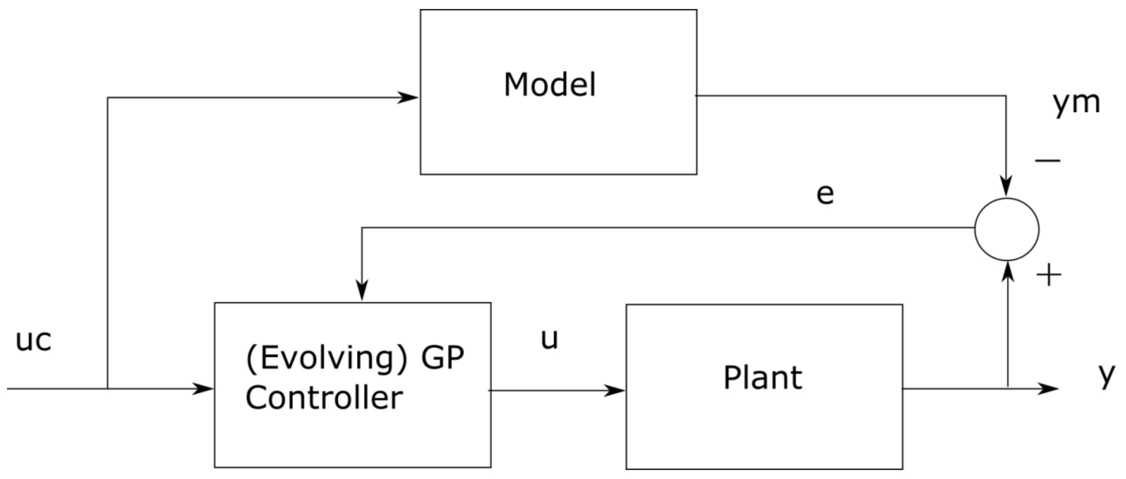

Figure 2.5: Block diagram of GP system design for MRAS controller [24]

In this diagram, uc is the controller input, u is the plant input, y is the plant output, ym is the model output and e is the error between the model output and the plant output. Although this work doesn't use GP to evolve the structure of a controller, it presents another approach of GP to generate the controller parameters. According to the conclusion of [24], GP is able to generate desired parameters of MRAS controller based on the model following without any prior knowledge about the system parameters [24]. Therefore this work presents another application field of GP for the controller parameter generation and GP is also able to find acceptable solutions.

- GP in the evolution of robotic controller

Apart from the controller design, GP can be also used for the evolution of robotic controllers [26] [27] [28]. [26] uses GP to achieve a robot reactive navigation task. The aim of GP is to evolve the best trajectory that the robot follows the environment without bumping into a wall. [27] uses GP to achieve the task of wall-following for a robot. In this work, different types of walls are tested for GP to evolve the acceptable solutions of robot behaviours without priori information about the environment. [28] also uses GP to evolve a robot behaviour controller. The aim of GP is evolving an appropriate relation between the sensor terminals and motor commands in order to manage the robot to achieve desired behaviours. Therefore two tasks are applied for GP to get acceptable controllers which are obstacle avoidance and box-pushing. Obstacle avoidance is to make the robot not bump any obstacle and box-pushing is to keep the robot pushing a box forward as straight as possible [28]. The results of these three works all show that GP can get good behaviours for a robot task based on the evolution of a robotic controller. Although they are not related to typical controller design problems in control theory area to realise dynamic performance index as well as the steady state error of static performance index, these works still present another application area of GP to achieve the robotic controller design. Moreover, GP is also suited to the multi-input multi-output (MIMO) controller design problems for robotics where the sensor readings can be used as the controller input values and the controller output values actually stand for different motor speeds, where a standard single input single output (SISO) controller is not able to achieve. In conclusion, the GP based robotic controller evolution is a promising way to achieve the robot behaviour management so as to achieve different robot tasks. In this sense, it would be interesting to investigate it into the robot fault tolerant control area and explore how it will be working.

2.3.3 Cartesian genetic programming

Cartesian genetic programming (CGP) is another type of GP which uses a two-dimensional grid of nodes to represent a program rather than the tree form used in GP [29]. In terms of the CGP genotype, each one is described with a directed acyclic graph of computational nodes. An example of CGP genotype is shown in Figure 2.6.

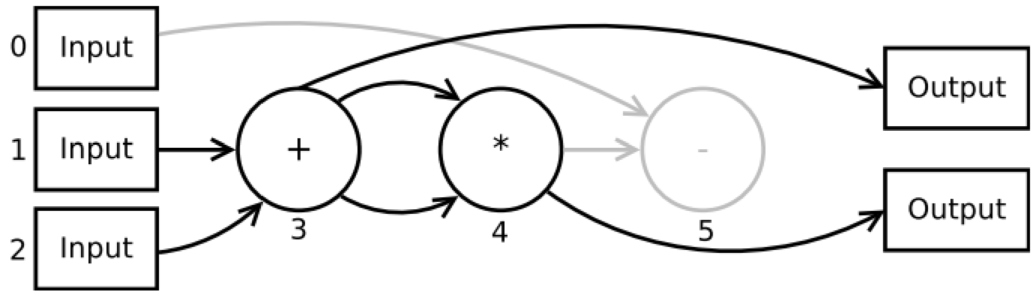

Figure 2.6: An example of CGP genotype [30]

The genotype of CGP consists of function genes, connection genes and output genes. One advantage of CGP over GP is that the node outputs can be reused more than once without recalculating the same required value, which can be seen in Figure 2.6. Another advantage is that CGP is quite suited to MIMO problems with the specified number of inputs and outputs. Moreover, CGP also does not suffer from program bloat problem and the details can be referred to section 2.4.4.2. Finally, CGP is also benefit from the inactive genes, where the details can be referred to section 2.4.4.5. Basically, CGP utilises (1+4) for the evolutionary strategy with point or probabilistic mutation. Point mutation changes the randomly selected genes with a fixed amount, which is determined by the total number of genes times the mutation rate. In terms of the probabilistic mutation, each gene will get a chance to be mutated based on a given mutation rate. Apart from the mutation, there is however no crossover utilised for CGP. A possible reason is that using crossover for CGP has not generally demonstrated any advantage for a wide range of task domains [30].

In terms of the CGP applications, three different fields are described which are related to CGP based controller design tasks. [31] and [32] are directly related to how to design a control system by CGP for two different nonlinear systems, so they will be discussed in more details. [33] is about how to evolve a robotic controller based on the relations between the input sensor signals and output motor speed of the robot. And [34] is about how to evolve the input signals of a motor controller to achieve sensor fault tolerant control.

- CGP for control system design

The work in [31] demonstrates that CGP can be also used for control system design besides the basic GP approach. [31] mentions that the computation time for GP based approach is extremely high so that an acceptable solution could take days of time to be evolved for a simple SISO controller design. So CGP is considered to be an alternative way with some limitations or simplifications in the task definition such as the orthogonal network for the individual representation. Since the interconnection of the nodes in this kind of network is not arbitrary as GP, so the solutions with much lower computation time could be obtained due to the reuse of nodes for the program description [31]. In this approach, CGP is used to design a controller for a nonlinear hydro-turbine system whose model can be referred to [35].

According to [31], each individual contains N interconnected building blocks where each block consists of three parts: the arithmetic operators (summation, subtraction, multiplication or division), the gain and the dynamic operators (integrator, derivative or unit gain). And the interconnection number between the controller inputs, building blocks and controller output is limited to M. N and M are priori determined based on the complexity of the system. So an appropriate selection of N and M by the designer will maximize the controller performance [31].

The fitness function is presented in the form of integral absolute error (IAE) which is defined in equation 2.9 where T is the simulation time.

$$\mathrm{I_{AE}} = \int_0^{\mathrm{T}} |\mathrm{e(t)}|\mathrm{dt} \tag{2.9}$$

This fitness function is just used to minimize the error between the reference signal and output signal. In this sense, unstable individual will be eliminated due to their high performance index. Moreover, a GA designed PID controller is also utilized as a comparison with CGP controller for the same problem.

As can be seen from the result, CGP designed controller achieves shorter rise time, shorter settling time and lower overshoot compared with GA tuned PID controller. And with the increase of generation number, CGP approach can get a lower cost function values compared with GA approach. The conclusion in [31] indicates that CGP is effective to obtain acceptable controller design result. And it uses additional limitations related to the controller structure and its size to reduce the computation effort compared to GP. In the future work, CGP can be used for controller design with complex MIMO and any type non-linear systems [31]. Therefore CGP based controller optimization could be a useful approach to design a control system. The only condition to apply this method is the existence of a suitable model of the controlled system [31]. As long as the system model is obtained and sufficient computation capacity is given, this approach is a promising method to obtain acceptable controllers.

[32] also uses CGP for the controller design of nonlinear system. This work uses a different system to demonstrate the ability of CGP to design acceptable controllers. [32] conducts an explicit comparison between CGP and GP. In terms of CGP, it has an exclusive limitation which defines the individual structures that the building blocks are normally organized in a fixed grid with a priori defined size and the task is to find the optimal types, parameters and interconnections among them. However GP generates the individuals with unlimited structures. So the limitation of GP is just the number of building blocks or the size of program tree or table [32].

The individual representation and fitness function of [32] is the same as [31]. The controlled system of [32] is a SISO system which is described by a differential equation in 2.10 where y is the system output value and u is the control value. Furthermore, a GA designed PID controller is also implemented as a comparison with CGP designed controller for the same system.

$$y'' + y' + y + 2y^3 - u = 0 \tag{2.10}$$

As can be seen from result, CGP approach can get controllers with acceptable performance while GA tuned PID controller has the problem of steady state error and even cannot reach the reference value when it changes. Therefore the conclusion of

[32] points out that PID controller doesn’t meet the requirements of all the different references for the time response due to its linear behaviour and insufficient robustness. However CGP controller is able to reach the reference value in an entire range. Although it is difficult to obtain the optimal controller because of the huge search space, this approach can still produce acceptable solutions [32]. In conclusion, [31] and [32] use two different systems to demonstrate the effectiveness of CGP to obtain acceptable controllers compared with GA tuned PID controller, which also indicates that CGP is capable to design a nonlinear control system based on the controller optimization.

- CGP in the evolution of robotic controller

Apart from control system design in the control theory field mentioned in the above two works, Cartesian genetic programming can be also used to generate controllers to manage robot behaviours [33] in addition to the GP based robot controller evolution [28]. In this work, the nodes from the first column of the evolved controller consist of two sensor inputs and two nodes from the last column stand for two motor speeds. The following functions can be selected for the nodes including Add, Subtract, Multiply, Divide, Compare, Min, Max, Fixed integer and Input node. The fitness functions are developed based on these factors such as time spent moving forward, total path length and Euclidian distance travelled [33]. Based on the utilizing of CGP, this work successfully creates controllers for two experiment tasks, which are escaping a room and solving a maze. As can be seen form this work, the approach could evolve a controller which constructs relations between the inputs of sensor values and outputs of motor speeds to complete the robot tasks such as obstacle avoidance and maze solving for robotic controller design mentioned in this work. What’s more, this kind of controller evolution based on CGP is quite suited to the MIMO controller design problems especially in robotic area since it could evolve a MIMO controller which utilizes the sensor readings as the controller inputs and creates controller outputs for each of the motor speeds respectively. In this case, this kind of MIMO controller will be working well to manage the robot behaviour rather than a typical SISO controller which is just designed on the utilize of the error as the unique controller input to generate an output value as the control signal to control the plant. In conclusion, this work indicates an interesting area of using CGP to evolve robotic controllers to manage robot behaviour to achieve different robot tasks. Based on this work, it would be interesting to consider evolving controllers to achieve robot fault tolerant control as long as the fault has been diagnosed.

- CGP in fault tolerance

[34] tries to use CGP to achieve sensor fault tolerant control. This work is related to controller design in the case of a sensor fault. However it is not about designing a controller, it is focused on how to generate the correct inputs to the controller using CGP with the remaining working sensors [34]. The controlled system is the Shaky Hand plate. The inputs of CGP are the plate sensor signals and the outputs will be the lateral and angle offset error voltages which are the inputs of controllers and used to drive motors to compensate for them. Therefore the aim of CGP is to generate the relation between the remaining working sensor signals and two offset error voltage values [34]. As can be seen from this work, CGP is still effective to search for reliable solutions for the sensor fault tolerant control. Although this work is not about the controller evolution, it indicates a new idea to evolve the inputs of controller which could also be helpful for fault tolerant control.

2.3.4 Grammatical evolution

Grammatical evolution (GE) is also another type of GP. It can evolve a program using arbitrary languages with a variable-length binary string. This binary genome determines which rule in the grammar is used to achieve the mapping from genotype to phenotype so that the program could be completed. Basically, Backus-Naur Form (BNF) is utilised as the original grammar rule employed for the mapping based on the building blocks in order to create the potential program. However, any language could be created based on this kind of simple binary string as long as an effective mapping process is available to implement [36].

In terms of GE applications, [37] presents a whole scheme about how to use GE to evolve a controller to design a control system and [38] talks about how to use GE to evolve a robotic controller for robot behaviour management.

- GE for control system design

According to [37], grammatical evolution can be used for controller design for arbitrary continuous time dynamic systems. The controller is represented in a continuous time functionF_c which includes the selected argumentsX_i, the mathematical relations and the parameters of the mathematical operations. The arguments of input variables are

$X_i \in \{e_i, ie_i, de_i, d^2e_i, y_i, dy_i, \text{other}\}$

where e is the control error, ie is integral of control error, de is derivative of e, r is the reference signal, y is the controlled value and other arbitrary variables. The individual can be represented in 4n genes:

$\text{individual} = \{f_1, f_2, \dots, f_n, x_1, x_2, \dots, x_n, p_1, p_2, \dots, p_n, b_1, b_2, \dots, b_n\}$

where f_i is the code of a mathematical operation, x_i is the argument of input variables, p_i is the parameter representing the coefficient of each x_i and b_i is the coefficient of the power operation. The grammar of the mathematical operation f_i is in the coding:

$0: \lambda \to \lambda, 1: \lambda \to (\lambda + \lambda), 2: \lambda \to (\lambda - \lambda), 3: \lambda \to (\lambda * \lambda), 4: \lambda \to \left(\frac{\lambda}{\lambda}\right), 5: \lambda \to (\lambda)^\wedge b$

where $\lambda_i = p_i x_i$

The fitness function is in a form of simple integral performance indices defined in equation 2.11 or 2.12 where T is the simulation time.

$$J = \int_0^T |e(t)| dt \ \to \ \min$$

(2.11)

$$J = \int_0^T |e(t)| dt + \alpha \int_0^T |\dot{y}| dt \ \to \min$$

(2.12)

A parallel evolutionary algorithm [39] is used in this work which is illustrated in Figure 2.7.

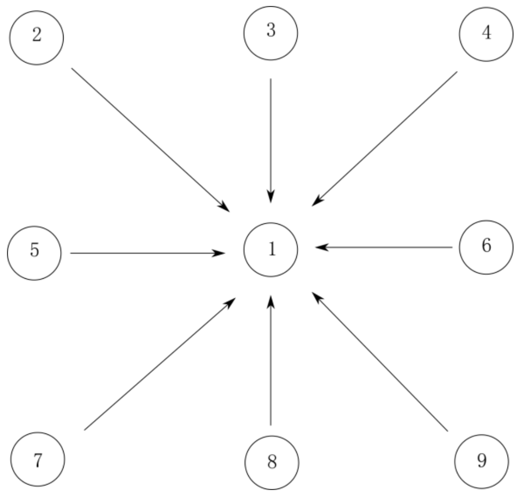

Figure 2.7: Migration topology of the used Parallel evolutionary algorithm [37]

In simple population of EA, this is always a conflict between the selective pressure and population diversity. Therefore by introducing multiple populations in parallel evolutionary algorithm, it is possible to simultaneously increase the selective pressure in some populations and improve the diversity of other populations [39]. In this kind of parallel EA shown in Figure 2.7, the individual representation is described using 9 islands in parallel architecture which are interconnected with migration connections and each island contains 50 individuals. It is a hierarchical structure that island 1 is the upper-level node while others are low-level nodes. Hence the difference between this kind of parallel evolutionary algorithm and the simple population evolutionary algorithm is the migration that in each generation, the best individual from island 2-9 will be selected and copied into the island 1 [37].

A non-linear stable controlled object is used for GE based controller design which is displayed in a differential equation 2.13. As a comparison, GA designed PID controller is also utilised for the same system.

$$\ddot{y} = 10u - 1\dot{y} - 2y - 4y^3 \tag{2.13}$$

As can be seen from the result, GE based controller has a faster rise time than GA based PID controller in terms of system output. And GA based PID controller also causes some oscillation when the reference signal drops to 0. Moreover, GA based PID controller generates much higher control value than GE based controller which means

PID controller needs more control effort to control the system. In conclusion, the result demonstrates that GE is an effective approach which could generate more effective controller than GA tuned PID controller.

On the other hand, a non-linear unstable system is also used for the design of GE based controller which is described in a differential equation 2.15.

$$\ddot{y} = 10u - 1\dot{y} - 2y + 4y^3 \tag{2.15}$$

According to the result, GA based PID controller leads much higher overshoot of system output than GE based controller. In addition, GA based controller generates much higher control value which means more control effort is required for GA based PID controller than GE based controller. In conclusion, the result also demonstrates that GE designed controller achieves better performance than GA tuned PID controller.

According to this work, the GE based controller has obvious advantages for the control of non-linear system due to its non-linear properties of the controller compared to GA based linear controller. The future research of this approach will design the controller for complex, non-linear and MIMO systems. On the other hand, this approach just uses five mathematical operations which are {+, -, *, /, ^}. In this case, more kinds of operations and functions can be considered to be added into the individual representation if they are needed. In summary, [37] demonstrates that GE is an effective approach to construct acceptable controllers to deal with nonlinear systems, which could be an effective approach for control system design.

- GE in the evolution of robotic controller

Similar to GP based robot controller evolution [28] and CGP based robot controller evolution [33], [38] also describes how to evolve a controller to achieve robot task but with GE. In this work, the task is navigating a robot toward a point light source and avoiding obstacles at the same time. The evolved controller by GE is a piece of computer program that generates C code in order to make robot achieve the task. The obtained program maps a relation between the sensor signals and the motor speeds in order to control the robot behaviour. The genotype is evolved using a steady state GA where only a small part of population is replaced each generation. The only difference is that the genomes in GE are represented in computer programs rather than binary or real values in GA for only evolving parameters. The fitness function is designed with two factors including a reward for finding a light and a penalty for collisions [38]. As can be seen from this work, it is quite similar to [28] and [33] where the evolved controllers are suited to solve MIMO controller design problems especially in robotics research area, which is also a potential way to achieve the robot fault tolerant control.

2.4 Evolutionary algorithms with artificial neural networks

2.4.1 Artificial neural networks

2.4.1.1 Introduction of artificial neural networks

Based on the reviewed literatures, EA is an effective optimization tool to design structurally evolvable controllers not only for SISO control problems but also for MIMO control scenarios. As can be seen from [28], [33] and [38], structurally evolvable EA approaches could also be promising to design controllers in terms of robot behaviour management. Although these approaches demonstrate benefits to design structurally evolvable controllers, those evolved controllers are created based on stochastic initial structures. That is to say, the output values of the controllers are actually arbitrary depending on which node functions are utilised and connected to the controller outputs. However the robot motor speed has the upper and lower limitations respectively. In this sense, the range of the controller output values has to be assessed and truncated before the output values can be utilised as the robot motor speed values [33], which is quite tricky with lots of extra work to do before initialising and evolving controllers.

Therefore, an alternative option is to use neuron transfer functions instead of basic mathematics functions as the controller node functions. The benefit is that the neuron transfer functions basically have their own output limitations such as [0, 1] or [-1, 1], which is well suited as the controller node functions in order to obtain output values in limited range as the robot motor speed without extra works to assess the controller output limits. In this sense, the work will become the evolution of neuron transfer function based controllers. In other words, artificial neural network (ANN) would be a suited choice as the basic controller type and EA could optimise a structurally evolvable controller based on it, which is called the training for the neural network. In addition, ANN has been investigated in the fault tolerance area extensively, which will be reviewed in section 2.4.1.2. Therefore it is also a promising idea to implement EA to realise a structurally evolvable ANN so as to achieve the fault tolerant control.

ANN is a significant research area in artificial intelligence and it has a wide application scope. Due to the nonlinear characteristic, ANN is capable to model a complex system where the accurate mathematical model is hard to obtain or just act as a nonlinear controller for a given task [2].

ANN is inspired by the animal brain's structure to mimic how the neurons transfer information in the real neural networks. Basically ANN describes a weighted directed acyclic/cyclic graph with a set of nodes implementing the neuron transfer functions so as to approximate the real biological neurons. In biological neural networks, the signals are transmitted as spikes between two connected neurons. In this case, the ANN which models the spiking behaviour for the information transmission is considered as Spiking Neural Networks. However most of ANNs utilise non-spiking neurons to constitute the network due to the less expensive computation effort with a wider application scope for the non-spiking ANNs.

Figure 2.8 shows a generalised neuron model for the non-spiking ANN. In this figure, x is the input from previous neurons; w is the connection weight which indicates the strength of the current connection; φ() stands for the neuron transfer function which processes the weighted sum of input signals to generate one output signal, and that output y in this figure represents the output from this neuron. In addition, there is an extra element for the ANN called the bias, which is b in this model. Bias is just used for any internal thresholds with respect to the neuron transfer function.

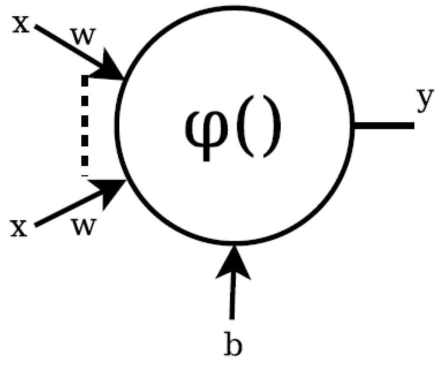

Figure 2.8: A generalised artificial neuron model [30]

There are many types of neuron transfer functions in the literature and most of them generate the output in the interval of [0, 1] or [-1, 1] [40]. Based on the implement of neuron transfer functions, ANN could be also used to describe graphs with different structures. For example, [41] and [42] demonstrate how feed-forward and recurrent ANN could be utilised respectively for universal system approximation based on a finite number of neurons, which indicate that both the feed-forward and recurrent ANN have wide application scope as long as they can be trained for the given task.

- ANN different structure types

To be more specific, several different ANN structures are reviewed as following. Feed-forward neural networks have the simplest structure. In this type of neural networks, each neuron outputs only the neurons of the next layer and there may be more than one hidden layer depending on the complexity of the system. Its architecture is presented in Figure 2.9 where each neuron outputs only to the neuron of the next layer [43].

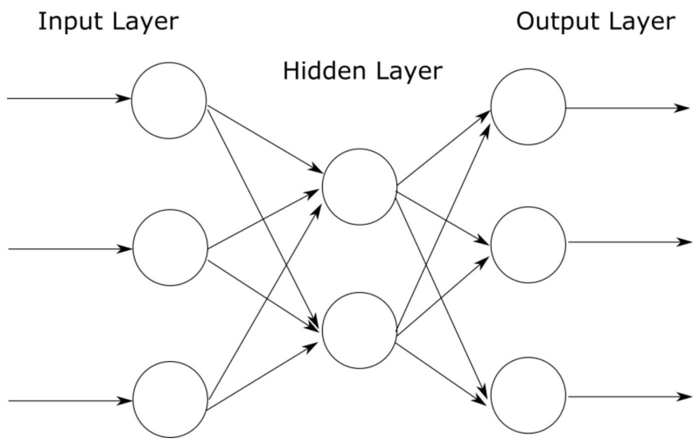

Figure 2.9: Feed-forward neural networks [44]

Recurrent neural networks have more complex structures than the simple feed-forward neural networks. This type of neural networks needs more computational power for training and implementation because of the reuse of past signals. Moreover, each input activity pattern passes through the network more than once before it generates an output [43]. Their structure is displayed in Figure 2.10.

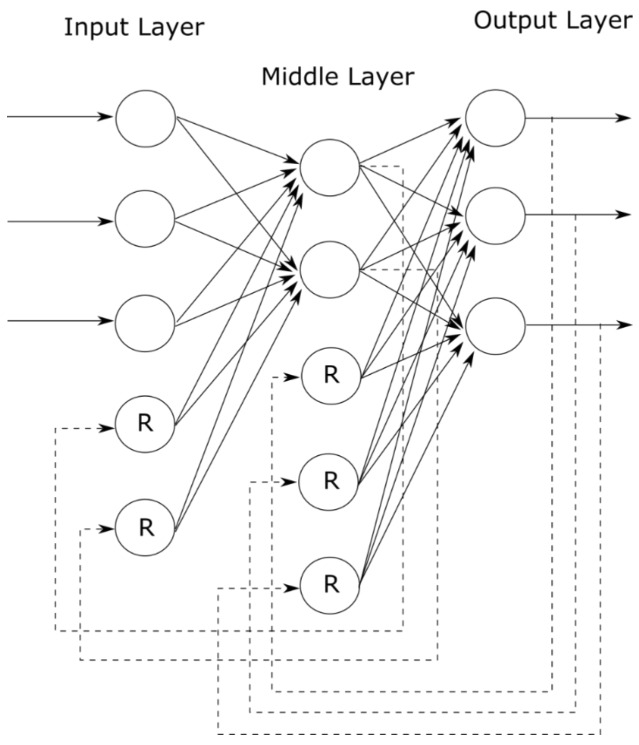

Figure 2.10: Recurrent networks [44]

Radial basis function neural networks are a kind of single hidden layer feed-forward networks which use radial basis function as activation functions. In this type of neural networks, the distance between the input vector and the vector of centres is calculated for each input which needs to be passed through the activation function [44]. Its structure is shown in Figure 2.11.

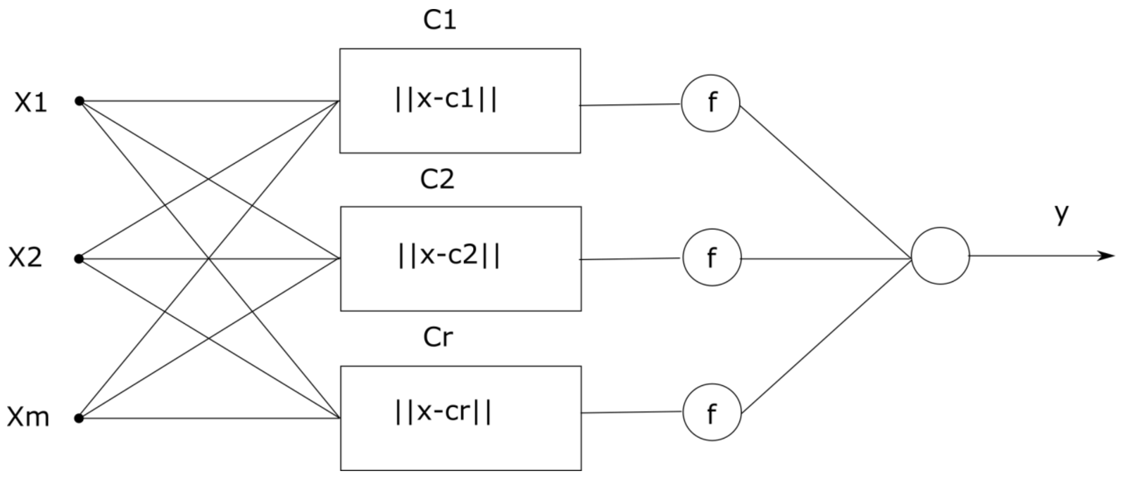

Figure 2.11: Radial basis function neural networks [44]

In this diagram, f is a radial basis function used for the activation function, y is the output, x is the input vector and c is the vector of centres. ||x-c|| is the distance between the input vector and the vector of centres.

Besides the above mentioned different kinds of neural networks, fuzzy logic can be also combined with neural networks using a fuzzifier to form the fuzzy neural networks [43]. Fuzzifier is used to convert the input data patterns to fuzzy categories which can be used as the inputs of neural networks. So this kind of neural networks is useful to deal with the system with imprecise information or noise with the aid of fuzzy logic [44]. Their structure is presented in Figure 2.12.

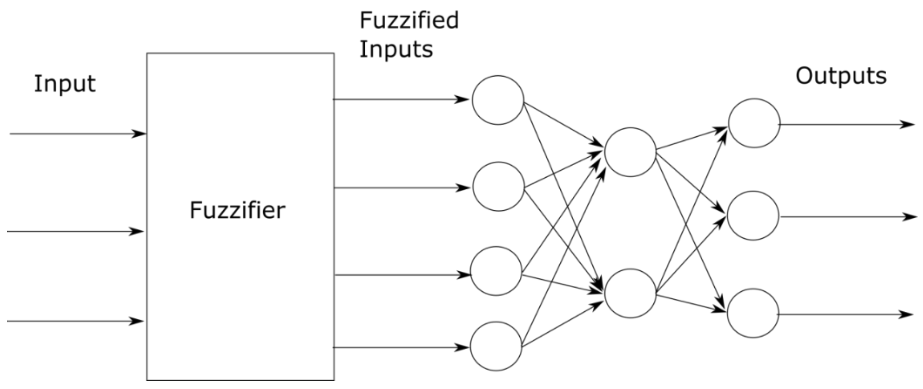

Figure 2.12: Fuzzy neural networks [44]

In summary, different types of ANN could be utilised in different scenarios depending on the complexity of the given task. Due to the nonlinear characteristic, ANN would be feasible to act as nonlinear controllers especially when MIMO are required to design the controller, where a standard SISO controller is not able to achieve.

- ANN training methods

Back propagation is the most widely adopted traditional ANN training method especially for multi-layered feed-forward network, where its first description is presented in [45]. Back propagation is working based on the error signal, which is defined as the difference between the real output and the expect output of the network. During the ANN training, the error signal will be propagated from the output to the input through each layer including all the hidden layers. In this way, the weight value of the network can be regulated by the error feedback and the real output will finally get quite close to the expect output based on the continuous modification of the weight values [46]. The mathematics details of back propagation can be referred to [46].

Apart from back propagation, EA is a new training method for ANN. There are many benefits that EA outperforms back propagation for ANN training. [47] compares EA with back propagation for ANN training based on five different test cases. The result shows that EA converges faster with a more accurate performance than back propagation. Moreover, EA demonstrates a better robustness than back propagation with a better average performance for these case studies when some neurons are lost before the training [47]. As can be seen from this work, EA demonstrates significant benefit over back propagation for ANN training. Although the comparison is conducted on those five test cases in [47], it still indicates that EA could be a new approach as the ANN training method rather than back propagation. The details of the five test cases can be referred to the work in [47].

2.4.1.2 Artificial neural networks in fault tolerant control

In terms of fault tolerant control, ANN has also been investigated widely not only for controllers but also as fault detectors. [48] applied ANN as controller, fault detector and fault compensator all together to achieve a fault tolerant control system. In this work, the neural network controller and fault detector are trained offline. When there is no fault, the controller is able to make the plant work normally. And when a fault is presented in the plant, the fault detector can generate a residual signal which indicates that a fault is detected. After that, an extra neural network based fault compensator will be trained online in order to ensure the closed loop stability [48]. This work demonstrates a wide application of ANN in the fault tolerant control area. However the fault tolerant control capability is based on the online training of a fault compensator in the face of detected fault while the ANN based controller's structure still remains fixed. In this case, if the ANN based controller could be redesigned, it would be interesting to see whether it will be working better other than the dependence on the fault compensator. [49] applied a dynamic radial basis function ANN as the controller to achieve a fighter aircraft fault tolerant control system in the case of severe winds when it is landing. This approach just utilises ANN as a feedback controller to design a fault tolerant control system. This work demonstrates that the ANN controller is capable to be trained online in order to deal with the fault. So this work indicates a possibility to investigate the controller redesign especially the controller reconfiguration with respect to the ANN based controller utilised to design an online fault tolerant control system. [2] made a comparison between a PID controller and ANN controller in order to achieve the fault tolerant control based on a Model Reference Adaptive Control (MRAC) system. In the MRAC system, the process output will be compared with the reference model output and the comparison result, which is the error in this system, will be processed by an adaptation mechanism in order to tune the controller parameters. In this sense, the process output will be following the reference model output until the controller’s optimal parameters are found, which means the system response will be matched with the reference model output and the whole closed loop system will be stable. Based on the MRAC scheme, this work utilised a PID controller and an ANN controller to do the comparison to achieve the fault tolerant control for a heat exchanger system where abrupt and gradual faults will be both injected on the sensors and actuators respectively. As can be seen from the experiment results, ANN controller based MRAC system represents the best performance for both of two types of faults in two different scenarios. ANN controller could be robust to these two faults injected in the sensors but PID controller has to reply on the adaption mechanism to re-tune the parameters in order to make the system response stable. In terms of actuator faults, although both PID and ANN controllers have degraded performance, ANN controller still outperforms than PID controller with a less degraded performance. In conclusion, this work demonstrates the benefits of utilising an ANN controller to do the fault tolerant control other than a traditional PID controller. In this sense, it would be interesting to investigate how ANN controller can be utilised into the robot fault tolerant control scheme.

As can be seen from these three works, ANN displays significant advantages when being acted as controllers in the fault tolerant control system. However the training method of these ANN is the back propagation, which belongs to the most typical traditional ANN training approaches. In order to investigate how EA could be used to train the ANN, [50] utilised GA for the weight retraining of a ANN controller which is used to realize the fault tolerance in single chip or silicon wafer. In this sense, the ANN controller can be reconfigured online to process different faults with the help of GA. This work is actually a typical example of employing EA approaches for the ANN training to design a fault tolerant control system, which belongs to a kind of NeuroEvolution (NE) approaches. Up to now, there are lots of similar works using GA to train the ANN in order to achieve fault tolerant control. However those ANNs have fixed structures and the only factor to be optimised is the connection weight, which is quite fitted for GA in terms of the ANN training. However if the whole structure of ANN could be optimised, those reviewed EA techniques which are working for the structure optimization mentioned in the section 2.3 will be promising approaches to help design a structurally evolvable ANN controller to achieve the fault tolerant control scheme, which would be interesting for further investigation.

2.4.2 NeuroEvolution

2.4.2.1 Connection weight evolution

As is mentioned in section 2.4.1.2, NE [51] [52] is a kind of optimisation technique that it applies evolutionary algorithms into the training of ANN to achieve the given tasks. As is reviewed from [50], this work combines GA with ANN together to achieve a fault tolerant control scheme, which belongs to a typical simple NE approach called Conventional NeuroEvolution (CNE).

CNE [53] [54] is the earliest NE approach which just applies a simple GA into the training of ANN's weights with a fixed network structure. CNE can be utilised to train either a feed-forward or recurrent ANN depending on the required network type for the task. Generally speaking, the genotypes of GA for the CNE are comprised of a string of floating point values which stand for the weight value for each connection in the ANN for training. So the phenotypes would be the corresponding weight values for each connection with a predetermined network structure. The fitness function is determined by the task which evaluates how well the genomes perform onto the ANN. The mutation could change a given percentage of the genomes by new random values and crossover could exchange the corresponding gene proportions of two parents in order to create two children. The initial population is made up of random values describing random connection weights. Finally the evolution could be terminated when an acceptable string of connection weights for the ANN are obtained to achieve the given task or the convergence criterion has been reached. Based on the description of CNE, it indicates that GA could be easily applied into the training of ANN's weights to complete given tasks. However there are also many different NE approaches working beyond the simple application of a simple GA into the training of ANN's weights.

2.4.2.2 Network structure evolution

Apart from CNE based ANN's weight training approach, there is another area that trains the network structure as well as connection weights, which is also important in NE area [51] [52]. [30] mentions that the search space of NE is actually comprised of structure and connection weight; or just the weight space with a given network structure. So only training the connection weights for a fixed network structure may limit the search space depth. In general, it is beneficial to just train the connection weight for a given suitable network structure since the dimensionality of search space is lower than training the whole structure. However a suitable network structure may not be determined in advance before the training and the effectiveness of the search is highly dependent upon a suited selected network structure. Therefore, it is a crucial drawback of just training the connection weights for NE. In other words, the structure training based NE approach can help determine the most suited network structure and sometimes it could also obtain an unusual structure but with a better performance, which otherwise may not be considered by a human designer [30]. In summary, other than just weight training, the network structure evolution can be also considered for NE approaches and many different NE approaches were developed for it.

Symbiotic Adaptive NeuroEvolution (SANE) [55] is a feed-forward, limited network structure evolution based NE approach. In SANE, each individual actually stands for each neuron. So the population is a combination of individuals which represent different neurons and the whole network will be created based on a random selection of these neurons. In terms of the genotype, each one is described as a hidden node with its connectivity and connection weights with the input and output nodes. However the limitation of SANE in the network structure evolution is that the network always just contains one hidden layer with the given number of nodes, arity and connections to output. In this case, only the connection placement and connection weights can be evolved, which restricts the evolution for a more complex network structure [30].

NeuroEvolution of Augmenting Topologies (NEAT) [56] is another typical network structure evolution based NE approach. It can evolve the weights as well as the structure of feed-forward and recurrent ANNs. Moreover, each individual stands for a whole network, which is quite different from SANE of using single neuron as individual. In terms of the genotype, each one is represented by a list of nodes and connections. The node is identified by an ID indicating whether it is an input, hidden or output node. And the connection gene includes an input and output node, a connection weight and whether this connection is enabled or not. In this case, a complete network can be constructed based on these node and connection genes for each individual. Interestingly, the initial individuals are created for simple network structures without any hidden nodes, where input nodes are directly connected to the output nodes. And new nodes or connections will be added only when they are required. However a crucial disadvantage is that this kind of incremental mutation could result in a local research of the network structure evolution, which would make the search trapped in local optima eventually [57]. Even though, a significant benefit of the evolution based on an initial small program size could be a solution to avoid the program bloat [30]. The program bloat is a common issue for many EAs especially for GP where the program size would be growing in an uncontrollable way during the evolutionary search [58] [59]. The main consequence of program bloat is that the training time may be increased and the computational effort could also be expensive for those solutions which would be extremely complex at the end of the evolution. However [60] also mentions that whether NEAT is suffered from program bloat is highly dependent on the choice of its parameters, where the typical parameter values in the early publications still cause program bloat.

2.4.2.3 Recurrent network evolution

In terms of recurrent ANN evolution, all fixed network structure evolution approaches are able to evolve recurrent network structures. Although some approaches like SANE do not allow the evolution of recurrent connection, it is simple to just modify the node connection such as to the input nodes, the previous nodes or even the node itself depending on the user requirement. In this way, it is not difficult for this kind of approaches such as SANE to obtain a recurrent ANN despite that the evolve network structure is still restricted to be a fixed structure [30]. As for the adaptive network structure evolution approaches, it seems that all of them are able to obtain recurrent ANNs. A possible explanation is that the connection placement can be also easily evolved just like the fixed network structure evolution approaches no matter which node it wants to connect as long as it is permitted by the user [30].

2.4.2.4 Transfer function evolution

Besides the recurrent network evolution, the neuron transfer function can be also evolved. Typically, if an ANN just utilises a single type of neuron transfer function, this kind of ANN is called homogeneous ANN. And if an ANN utilises more than two types of neuron transfer function, it is called heterogeneous ANN. [30] mentions a significant benefit of heterogeneous ANN over homogeneous ANN that homogeneous ANN may limit the ANN performance since different ANN training methods restrict the neuron transfer function types. However it has demonstrated that neuron transfer functions affect the capability of an ANN significantly [61] [52]. In this case, NE can be also utilised to manipulate the selection of neuron transfer functions without any restriction during the training especially for heterogeneous ANN as long as sufficient types of neuron transfer functions are given. Theoretically, the transfer functions can be simply described as extra genes added into the genotype for each node in order to be evolved. In this case, evolving a heterogeneous ANN could be possible as long as the extra transfer function genes can be also evolved [30].

2.4.3 NEAT/HyperNEAT

2.4.3.1 Introduction

- NEAT

As is mentioned in section 2.4.2.2, NEAT is one of the most important approaches in network structure optimisation based NE area in spite of a potential problem of program bloat. NEAT was developed by [62] based on ANN structure optimisation in the NE area. Each genome in NEAT is made up of a list of connection genes where each gene connects two node genes. Each connection gene specifies what input and output nodes is connected; what the weight values is; whether the connection is enabled or not and an innovation number which is utilised during the crossover [62]. An example of the genome encoding is shown in Figure 2.13.

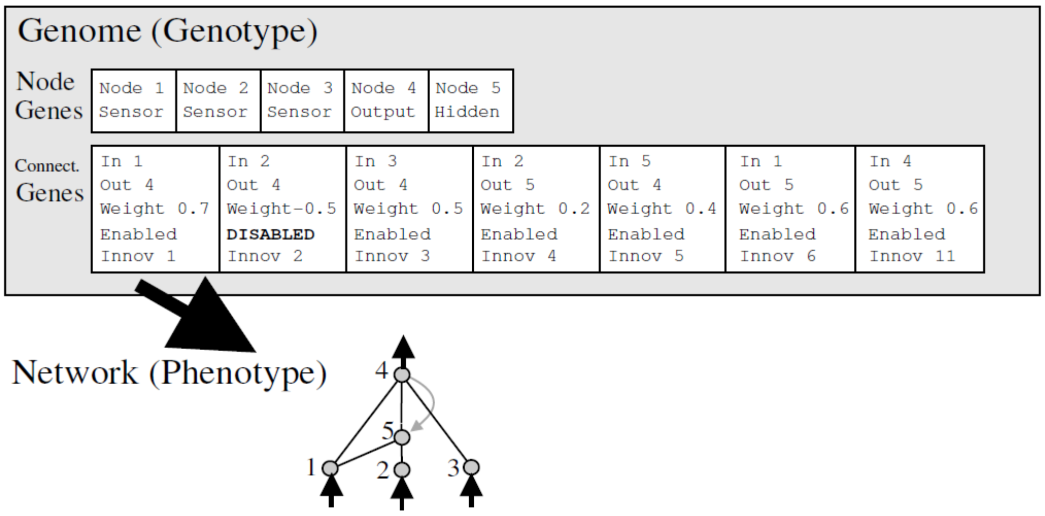
Genome (Genotype)
Node Genes
Node 1 Sensor
Node 2 Sensor
Node 3 Sensor
Node 4 Output
Node 5 Hidden
Connect. Genes
In 1 Out 4 Weight 0.7 Enabled Innov 1
In 2 Out 4 Weight-0.5 DISABLED Innov 2
In 3 Out 4 Weight 0.5 Enabled Innov 3
In 2 Out 5 Weight 0.2 Enabled Innov 4
In 5 Out 4 Weight 0.4 Enabled Innov 5
In 1 Out 5 Weight 0.6 Enabled Innov 6
In 4 Out 5 Weight 0.6 Enabled Innov 11
Network (Phenotype)
4
5
1
2
3

Figure 2.13: An example of NEAT genotype and phenotype [62]

In NEAT, the mutation could be occurred for both of connection gene and network structure. Like the normal weight mutation, each weight gene could be mutated with a fixed probability to a new floating point number. In terms of the structure mutation, each mutation will increase the genome size by adding new genes. To be more specific, there are two ways for NEAT to conduct the structure mutation. One is the adding connection mutation, where a new connection is added with two previously unconnected nodes. The other is the adding node mutation, where a current connection is split into two parts with a new added node [62]. An example of these two way mutations is shown in Figure 2.14.

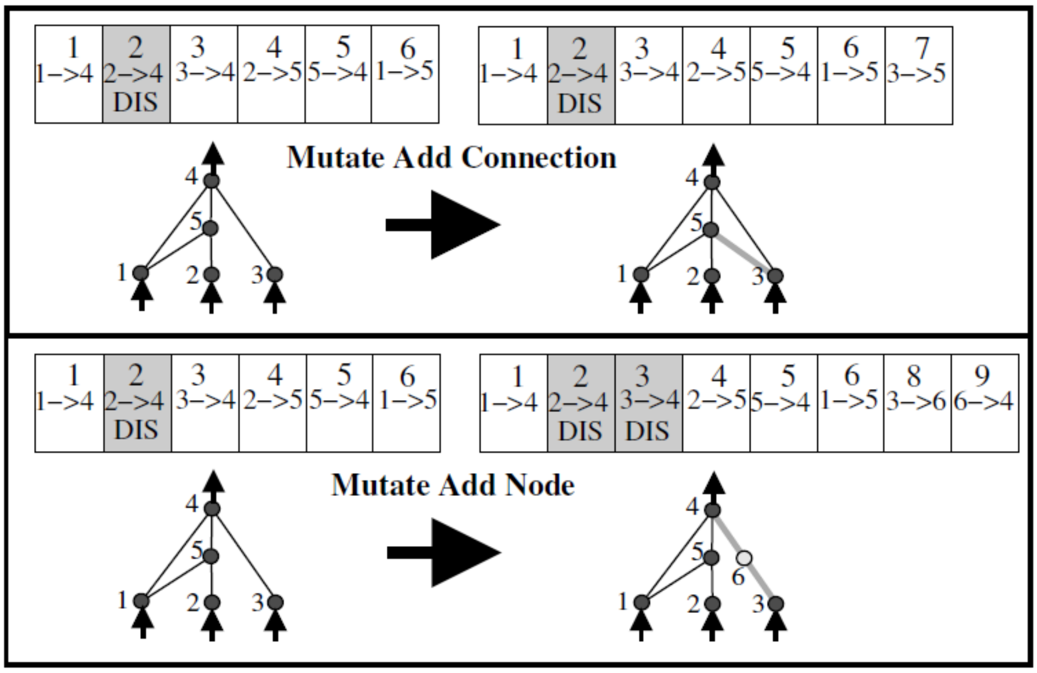

Figure 2.14: An example of two mutation ways of NEAT [62]

Besides mutation, crossover is conducted with the help of innovation number in NEAT. When a new gene appears, an innovation number is incremented and attached to the gene. The benefit is that when crossover takes place, the children will inherit the same innovation numbers from each gene, which guarantees that the historical origin of each gene is known during the evolution. Based on the innovation numbers, the crossover lines up the genes with the same innovation numbers and just chooses them randomly to create the children. If the genes are not shared with the same innovation number, the crossover adds them from a fitter parent into the children. In this way, the genomes with different structure could be combined compatibly during the crossover [62].

One significant benefit of this kind of crossover is that the competing conventions problem could be avoided. “Competing conventions means having more than one way to express a solution to a weight optimization problem with a neural network. When genomes representing the same solution do not have the same encoding, crossover is likely to produce damaged offspring” [56]. That is to say, when crossover is executed on two parents who have the same fitness but different genotypes, their children created from the crossover may lack genetic information and no longer function like either parent. Competing conventions problem is actually a common issue in NE approaches which utilise crossover to create children. However due to the utilise of innovation numbers, NEAT could identify which genetic material is shared or not share between their parents and the crossover could take place by selecting random shared genes or non-shared genes from a fitter parent, which prevents the competing conventions problem [30]. And at the moment, NEAT and the following reviewed HyperNEAT are the only two approaches in NE which utilise crossover but do not suffer from competing conventions problems. An example of crossover with the implement of innovation number is shown in Figure 2.15.

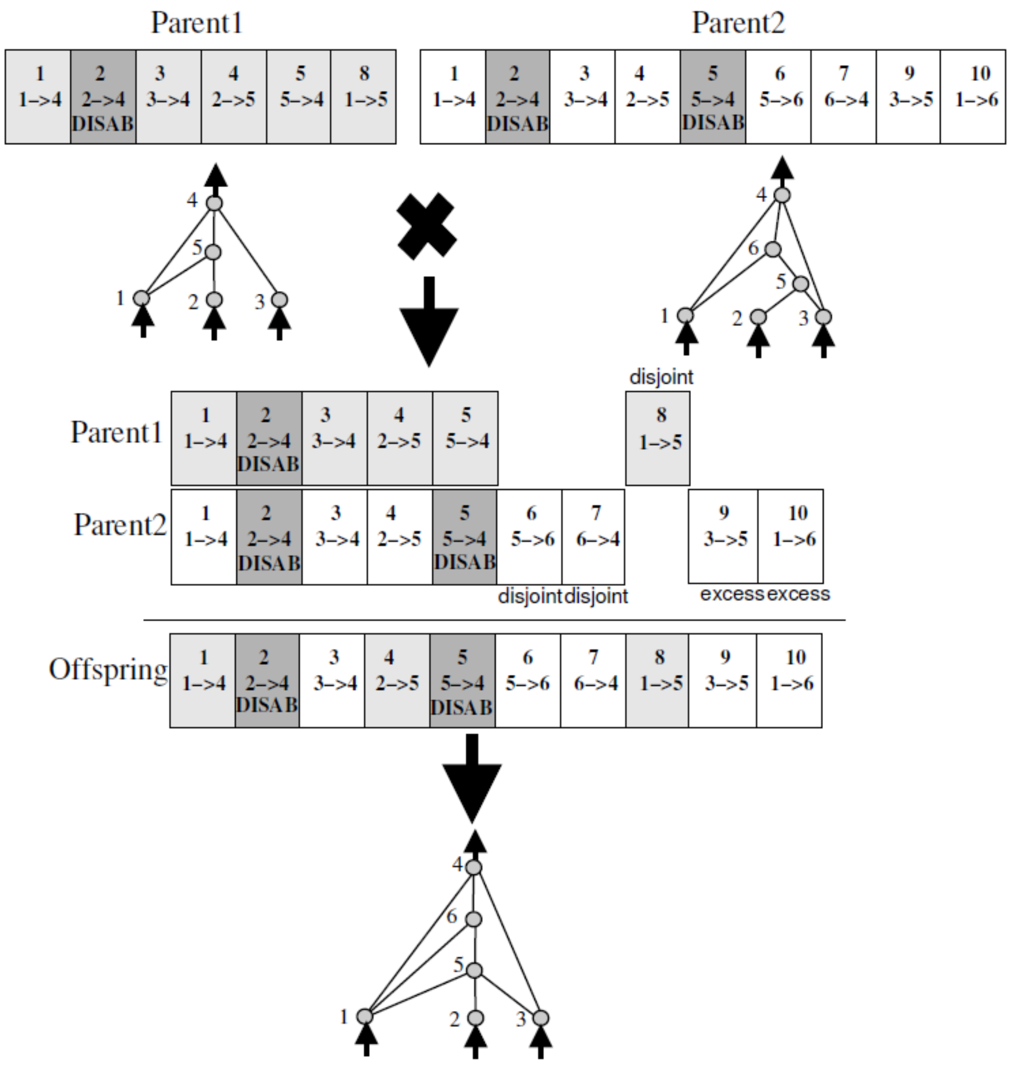

Figure 2.15: An example of crossover based on innovation number in NEAT [62]

- HyperNEAT

Apart from NEAT, Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) is also another important approach in network structure optimisation based NE area. HyperNEAT was first introduced in [63] which was developed based on connective Compositional Pattern Producing Networks (connective CPPNs) as the encoding genotype with NEAT for the evolution. CPPNs are used to represent connectivity patterns since they can produce spatial patterns made up of different basic functions. For this reason, HyperNEAT firstly creates the spatial patterns onto a hypercube whose dimensionality is determined by the dimension of the input coordinates. And then HyperNEAT maps the connection weights onto the neural network whose neurons and connections should be in a spatial location. Now the genomes are created and the genetic operator including mutation and crossover will be the same as NEAT for the network structure evolution [63].

2.4.3.2 NEAT/HyperNEAT in controller structure optimisation

- NEAT

Although [60] mentions that NEAT may also suffer from program bloat if inappropriate parameter values are set for the evolution, NEAT still demonstrates more effective performance than the fixed network structure optimisation NE approaches utilised to highly complex problems, such as the double pole balancing [62]. Moreover, an online evolution with NEAT also produces effective performance in terms of video game characters evolution, where the approach is also referred to a real-time enhancement of NEAT (rtNEAT) [64]. In recent years, a novel online and distributed version of NEAT (odNEAT) is developed by [65] which is quite similar to rtNEAT. The significant difference between them is that reNEAT utilises a centralised manner for evolution but odNEAT is completely decentralised. For this reason, odNEAT is quite suited to the robot control area where the controllers of multiple robots can be evolved independently online and onboard. Each controller is represented by an ANN indicating a candidate solution for the task on the corresponding robot and odNEAT is running on this group of robots with parallel evolution to perform the same task with genomes migrated between each [65], which is actually quite similar to parallel evolutionary algorithm interconnected with migration connections [39] mentioned in work [37].

[65] utilises odNEAT to conduct a simulated collective robotics experiment. odNEAT is used to create an ANN controller loaded to each robot. The input of the ANN is the

robot proximity sensor values and the output is the speed for each wheel. The task selected is an aggregation task where each robot needs to move close to each other in order to create a cluster. The result shows that 22 evolutionary runs are successful to make the group of robots achieve the aggregation task of all 30 runs. Although not all the runs are successful, the result still demonstrates that odNEAT is able to evolve adequate robot behaviours so as to achieve the same goal including the searching, locating and joining other robots in the environment. In conclusion, [65] is a typical work that demonstrates the effectiveness of utilising odNEAT to achieve the online controller evolution for group robot behaviours based on the parallel evolution with genome migration between each robot.

As can be seen from these reviewed works, odNEAT mentioned in [65] is one of the most typical approaches based on NEAT to be successfully utilised in robot controller optimisation. Due to the effective performance of odNEAT, [66] utilises odNEAT to achieve an online ANN controller optimisation but based on the real robots, which additionally demonstrates the performance for robot fault tolerance. [66] utilises the same odNEAT as that mentioned in [65] to evolve ANN controllers but for real robotic hardware rather than simulation robotic platform. The tasks selected in [66] include two single-robot tasks: the navigation with obstacle avoidance and the homing towards a target area. Moreover, aggregation is also selected as a collective robotics task to be conducted with a group of robots. In terms of the fault tolerance, the task is still the aggregation with pre-evolved controllers for the fault-free scenario. The fault is injected into a random robot's wheel within the robot group during the task, so the aim of fault tolerance is to investigate whether the robot will continue doing the aggregation with resumed online evolution for the faulty robot's controller. To be more specific, there are 3 test cases for the online controller evolution for the fault tolerance including one fault, two faults and three faults occurred among the group of robots. Each group consists of 3 robots and the fault is randomly injected in either the left or the right wheel of the randomly selected robot. The result shows that the online evolution with odNEAT is effective to produce a considerable set of successful controller in each run in terms of both of two single robot tasks and a collective robot task for the real robotic hardware. In terms of the fault tolerance, all of these three scenarios successfully evolve controllers online to overcome the injected faults for the selected robots in order to make the group continue doing the aggregation task, which demonstrates that odNEAT is effective to achieve the fault tolerant control with the controller online optimisation based on real robotic hardware [66].

As can be seen from these works, NEAT demonstrates effective performance in NE area especially for ANN controller structure optimisation. Among them, [66] is currently one of the typical works that utilises NEAT into the fault tolerant control field in terms of robot control optimisation. However [66] still displays a potential problem with a time-consuming work based on the online controller evolution for fault tolerance. Although [66] mentions that all the experiments conducted in this work require less than an hour to obtain an acceptable controller, those evolved fault tolerant controllers are just suited to the aggregation task. In other words, an hour evolution time is fine for the robots to overcome fault when they are doing the aggregation task since there is no criteria that the robots have to overcome the fault in order to continue performing the aggregation in a limited time. However if there is a requirement that the robot has to achieve the fault tolerance immediately right after the fault is diagnosed, which refers to a kind of real time fault tolerance, the online fault tolerance will be definitely not a suitable solution. An alternative solution is that the online fault tolerance could evolve the controller before the fault occurs rather than after it, which shortens the evolution time. However this idea needs the fault prediction technique, which is another research topic related to the fault tolerant control area. In this case, a simple but still effective approach is the offline fault tolerance based on the controller evolved offline. As is mentioned in section 2.2, the benefit of the offline fault tolerance is that some real time fault tolerance tasks could be achieved based on either robust or switched control with the offline evolved controllers. However a significant drawback is that all the possible fault types have to be considered in order to conduct the controller evolution. That is to say, if an unplanned fault is occurred during the online task, there is no way to overcome the fault and that is why online fault tolerance is needed. However it can be assumed that the possible fault types have already be considered for evolving fault tolerant controllers offline since this thesis is just focused on the fault tolerant control area, which does not matter whether all the possible fault types are considered or not. For this reason, the offline controller evolution could be a primary approach based on the ANN controller structure optimisation for fault tolerance before the online task is performed, which would be much more efficient than evolving a controller online.

- HyperNEAT

Besides NEAT, HyperNEAT also demonstrates competitive performance in the controller structure optimisation for robot control. [67] [68] and [69] all investigate how HyperNEAT could be utilised to achieve the robot gait learning. [67] implements HyperNEAT for the online gait training for modular robots in simulation and compares its performance with the reinforcement learning method. The result shows that the reinforcement learning method outperforms HyperNEAT where even the best controller in the best run of HyperNEAT produces a worse performance than the reinforcement learning method during the first 400 evaluations and the median value from HyperNEAT is also much lower than the reinforcement learning during the 1000 evaluations. Moreover, HyperNEAT also requires a much longer learning time than the reinforcement learning method to obtain an effective solution for the online gait learning task. For this reason, HyperNEAT may not be a suitable approach for the online gait learning based on modular robots in the task mentioned in [67]. [68] investigates how HyperNEAT could be utilised to achieve the gait training for a legged robot and tests the hypothesis that whether hyperNEAT will outperform the simpler encoding if the gait is firstly evolved in simulation and then transferred to real robot. When the evolution is finished in simulation, the best solution of each of 20 runs is transferred onto the real robot and distance travelled will be the measurement. The result shows that HyperNEAT produces a better performance than the simpler encoding for robot gait training in terms of the simulation task. Furthermore, this work also demonstrates that it is effective to evolve gait in simulation and then transfer the solutions onto real robot, although the solution just performs slightly better that directly evolved on real robot. Nevertheless, evolving gait in simulation first and then transferring onto real robot indicates another option for gait training with real robot based on the implement of hyperNEAT [68]. [69] is actually quite similar to [68] where HyperNEAT is also utilised to train the gait on a legged robot, but the difference is that [69] directly evolves gait on a real robot without any concern from the simulation work. Moreover, this work compares HyperNEAT with locally searching parameterized motion models based on their performance for real robot gait training. The result shows that HyperNEAT outperforms all the parameterized local search methods mentioned in this work and obtains a gait much faster than a hand-designed gait, which demonstrates the benefit of using HyperNEAT for real robot gait training.

Except for these three works, [70] develops a new version of HyperNEAT called iterated evolvable-substrate HyperNEAT (iterated ES-HyperNEAT). And this work shows that iterated ES-HyperNEAT reduces the computational costs compared to the original ES-HyperNEAT. In addition, this works also demonstrates that iterative ES-HyperNEAT outperforms original HyperNEAT in terms of a robot maze navigation task [70]. In conclusion, this work demonstrates that HyperNEAT could also be improved to achieve better performance for robot control. [71] utilises HyperNEAT to evolve controllers so as to achieve organism locomotion with obstacle avoidance. This work belongs to the field of evolution of robot organisms, where the robot organisms usually refer to the structures consisting of physically connected individual robots. In this work, controllers are evolved based on HyperNEAT in order to achieve the locomotion of a quadruped organism composed of 14 simple modules in addition to obstacle avoidance. The result shows that the evolved gaits are smooth and seem natural when the organism moves in a controlled, co-ordinated manner while negotiating obstacles [71]. In conclusion, this work shows that HyperNEAT is effective to develop a reactive quadruped gait with individual robot's controllers acting autonomously to achieve the successful locomotion of a given organism, which demonstrates the benefit of using HyperNEAT in the field of robot organism evolution.

As can be seen from these works, all of them are about how HyperNEAT is implemented for the robot controller evolution, which demonstrates the effectiveness of HyperNEAT in this area in terms of ANN structure optimisation. In addition, these works successfully implement HyperNEAT for controller optimisation of either single or modular robots based on simulation or real robotic platform. Although HyperNEAT demonstrates effective performance especially in robot controller optimisation area, there is currently no literature that implements HyperNEAT for fault tolerant control. Even though, it is not a difficult task for HyperNEAT since NEAT already achieves fault tolerant control with online controller optimisation [66] and the only difference between HyperNEAT and NEAT is the way of genome encoding. For this reason, HyperNEAT could be an alternative approach besides NEAT to achieve the fault tolerant control based on the ANN structure optimisation.

2.4.3.3 Summary

As can be seen from these works, both of NEAT and HyperNEAT demonstrate effective performance in terms of the ANN controller structure optimisation for robot controller evolution. Especially, [66] also utilises odNEAT to investigate how it will be performing for robot fault tolerant control, which is quite related to this thesis topic. However, as is mentioned in 2.4.3.2, the only problem in [66] is from the online fault tolerant control scheme, which needs at least a period of time to obtain an acceptable controller. In this sense, evolving controller offline before robot performs the online task would be another scheme to achieve the fault tolerance, which avoids the problem of a time-consuming work for online controller evolution as long as it is assumed that the possible types of faults have been considered for the evolution. For this reason, offline fault tolerant control could be a primary scheme to investigate how ANN structure optimisation approaches could be performed for it.

2.4.4 CGPANN

2.4.4.1 Introduction of CGPANN

Apart from NEAT, Cartesian Genetic Programming of Artificial Neural Networks (CGPANN) is also another important approach in network structure optimisation based NE area. CGPANN was first developed by [72] to achieve the network structure optimisation of ANN based on the original CGP but in the NE area. CGPANN has the similar framework with CGP to describe structurally evolvable graphs but for ANN training. Figure 2.16 shows an example of a simple CGPANN's genotype [30].

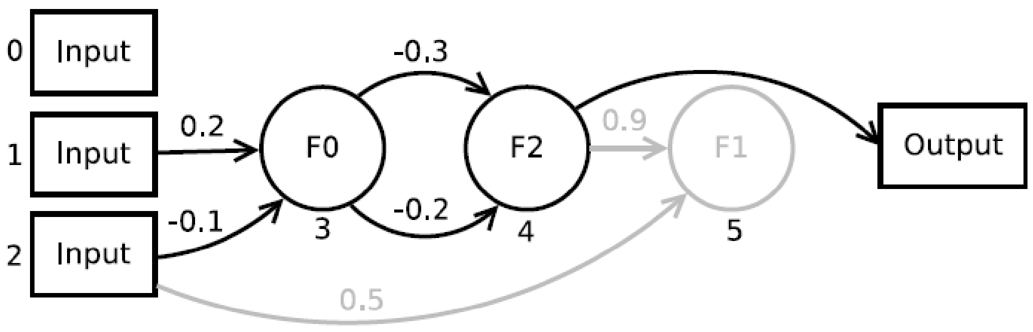

Figure 2.16: An example of CGPANN genotype [30]

In Figure 2.16, this CGPANN has three inputs and one output with three nodes. Each node acts as each neuron in ANN's framework. Moreover, each node has two connections from previous inputs, where the arity of the node is two in this case. Each connection is also coupled with a value to stand for the connection weight for ANN description. Finally, each node also implements its own node function. In this phenotype, the node function will be neuron transfer functions instead of basic mathematics functions. It needs to note that sometimes the node will not be connected to any other nodes including the inputs and outputs. As a result, this node is inactive for the graph description, so this node's relative genes are called redundant genes in this case, which is an important feature for NE and will be talked about in section 2.4.4.5. Therefore, the significant difference between CGPANN and CGP is that CGPANN adds an extra gene used to represent the neuron's connection weight for each connection among different nodes. In addition, CGPANN utilises neuron transfer functions to act as the node functions rather than basic mathematics functions. Other aspects will be remained the same from CGP.

Actually, CGP has some benefits over GP for the ANN's structure evolution. [73] mentions that CGP is more suited for ANN training rather than GP based approaches. The reason is that GP describes the program in a tree based structure, which is not suited to ANN encoding. However, CGP arranges the nodes in a graph based structure, which enables the reuse of nodes so as to make it possible to describe ANN. Apart from that, CGP also has further advantages including the management of explicit genetic redundancy [74] and the ability to overcome program bloat problem [75], which will be discussed in the next section 2.4.4.2 and 2.4.4.5. Moreover, the benefit of evolving network structure and heterogeneous ANN based on CGPANN will also be talked about in the section 2.4.4.3 and 2.4.4.4.

2.4.4.2 Program bloat

Resilience to program bloat is a benefit of CGP as well as CGPANN. Program bloat is a common problem for many GP based NE approaches which were developed with similar features as GP. Although [60] talks about how to ease the program bloat problem based on NEAT, the resilience is still highly dependent on a careful choice of its parameters. Therefore, more works are needed to investigate how to solve the program bloat problems for NE approaches and CGPANN is one of the approaches which present an effective resilience to program bloat.

To be specific, program bloat refers to a phenomenon that the size of evolved program increases dramatically but without significant improvement on the fitness [76]. This definition of program bloat is actually a metric applied in [77] to measure the amount of bloat for each generation. Figure 2.17 shows the comparison of the average bloat amount of the population for a standard tree-based GP (grey line) and DynOpEq GP (black line) from the work of [77]. In this figure, (a) shows the result of a symbolic regression problem and (b) (c) are referred to two real world classification tasks. As can be seen from the comparison results, a standard tree-based GP is seriously affected by the program bloat that the bloat amount is going up continuously with the increase of generations.

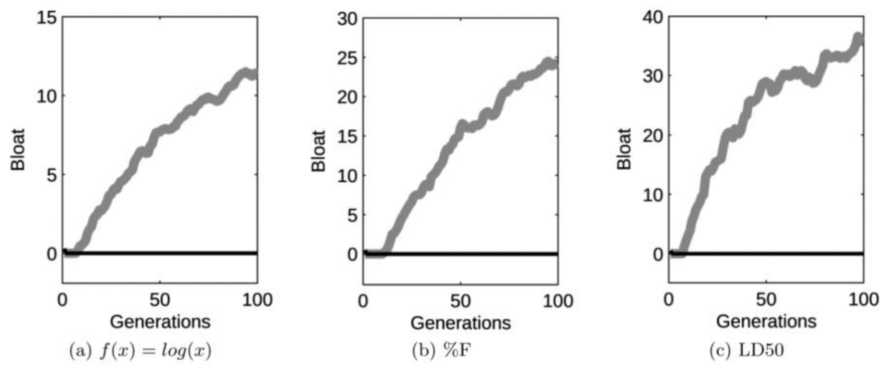

Figure 2.17: The comparison of GP and (gray line) and DynOpEq GP (black line) on (a) symbolic regression and (b) (c) two real world classification tasks in terms of program bloat investigation. [77]

In [30], the program bloat metric is modified slightly to suite the CGPANN framework which is demonstrated in equations 2.17 to 2.19:

$$N(g) = \frac{\hat{A}\,(g) - \bar{A}\,(0)}{\bar{A}(0)} \tag{2.17}$$

$$D(g) = \frac{\bar{F}\,(0) - \hat{F}\,(g)}{\bar{F}(0)} \tag{2.18}$$

$$B(g) = \frac{N(g)}{D(g)} \tag{2.19}$$

Where $B(g)$ is the bloat at generation g, $\hat{A}\,(g)$ is the number of active nodes utilised by the fittest individual of the population at generation g, $\bar{A}(0)$ is the average number of active nodes for each individual in the population at generation 0, $\bar{F}\,(0)$ is the average fitness for the population at generation 0 and $\hat{F}\,(g)$ is the fitness of the fittest individual of the population at generation g. Actually equation 2.19 represents the ratio of the increase of program size to the improvement of fitness from the first generation. So if $B(g)$ is becoming bigger, that means program size is increasing disproportionately to the fitness improvement, which indicates that the program bloat is existed. In other words, if the bloat value is constant, that means there is no program bloat over the past generations [30].

Based on equations 2.17 to 2.19, [30] utilised three benchmarks to investigate whether the program bloat exists in CGPANN. The three experiments include a double pole balancing, ball throwing and the Monks Problem 1. The first two experiments belong to control problems and the last one is a kind of classification task. The details of the experiments setup can be referred to [30]. The results of these three experiments are averaged over fifty runs in terms of fitness, number of active nodes and the bloat value at each generation.

According to the result, CGPANN actually does not suffer from the program bloat at all since the bloat is in a low level with nearly stable values over generations. Moreover, CGPANN is utilised in three different benchmarks, which also demonstrates the generalisation of CGPANN to overcome the program bloat. Since program bloat is a common problem in tree-based GP approaches resulting in slower evaluation time for extremely bloated programs, CGP would be another choice instead of the standard tree based GP in terms of the ANN's structure optimisation tasks.

2.4.4.3 Network structure evolution

Apart from the resilience to program bloat, network structure optimisation is another significant benefit for CGPANN. Generally speaking, network structure evolution based NE approaches are considered to have more advantages than the traditional training method based approaches. A significant benefit is that evolving the whole network structure removes the requirement for users to design a suitable network structure beforehand, which reduces the workload of human designers instead. Another benefit is that the network structure optimization could evolve an ANN structure that cannot be considered by a human designer but may achieve better performance than traditional ANN structures [51] [52].

As is reviewed in section 2.4.2.2, there are some approaches which are able to train ANN not only in connection weights but also in network structure. However there is nearly no literature that really talks about whether evolving network structure brings any benefit for ANN training. The only example is found in [78] where the results indicate that the evolution for a network structure may take more time than just for connection weights in order to find a suitable solution. So the fixed network structure based on weight evolution would be a first choice rather than those based on both of structure and weight evolution. However [30] suggests that it is an unfair comparison. Although it is possible to just compare the performance between these two approaches, it is still quite difficult to make a comprehensive comparison. One possible reason is that these two approaches utilise different genotypes to describe ANN during evolution. Some approaches are working at neuron level like SANE and some are working at network level like NEAT. So it is not clear to demonstrate whether their performance difference is due to the difference between connection weight optimization and network structure optimization or just the genotype difference or other factors between these two approaches. On the other hand, most of the connection weight optimization approaches are working based on a pre-designed network structure, whose effort is not considered into the comparison. But the network structure optimization approaches have to evolve both of structure and weights at the same time. So it does not make any sense when comparing the optimisation time of connection weight evolution based on a pre-optimised structure and the evolution for both connection weights and network structure.

In this case, [30] investigates two problems based on the comparison between network structure optimisation and connection weight optimisation. One problem is whether network structure optimisation is better than connection weight optimisation and the other problem is the relative importance between these two approaches.

- Network structure optimisation better than connection weight optimisation?

[30] conducts a comprehensive comparison between connection weight optimisation based approach and network structure optimisation based approach with CNE and CGPANN respectively for NE. This comparison investigates two possible perceived benefits of structure optimisation in the literature:

1) There is no requirement for network structure optimisation that a suitable structure needs to be obtained in advance.
2) Network structure optimisation could obtain a network structure which will unlikely to be considered by human designer.

In terms of the first perceived benefit, CNE is used to optimise the connection weight for ANN with a series of different network structures based on a number of benchmark tasks. And CGPANN will also be utilised for the same experiment as a comparison in order to investigate whether network structure optimisation approach could alleviate the requirement to choice a suitable structure by the human designer before evolving the connection weights with CNE. However due to the implementation difference for CNE and CGPANN, the comparison will be conducted based on the effect of adjusting the fixed structure for CNE and the impact of tuning the structure limits for CGPANN. In terms of the second benefit, an ANN's structure which could be considered by human designer can be defined as the structure with the standard layers including nodes per layer, which is normally utilised by standard ANN. In this case, if an effective network structure is obtained but cannot be described with a standard type, it can be concluded that those evolved network structures will not be considered by human designer.

[30] conducts the comparison experiments between CNE and CGPANN for the same three benchmarks and the results demonstrate that the network structure has a considerable impact on the CNE performance for ANN training. As for CGPANN, the results demonstrate a reversed phenomenon that different structure limits seem not to influence the evolutionary. That is to say, even if a suitable network structure is not acquired before the evolution, CGPANN will still be able to obtain the fittest network structure with the connection weight as well. In other words, poor performance could be avoided even if a suitable structure is not obtained in advance. So that is why there is no large difference for the performance in terms of different structures. In addition, there is an interesting result from the Double Pole Balancing benchmark that CGPANN obtains better results than CNE for a wide range of structure limits. On the one hand, this result may indicate that CGPANN's performance could be improved with the simultaneous tuning of connection weight and network structure. On the other hand, it also implies that CGPANN could obtain an effective network structure which is not available for CNE to utilise, which may be the reason why CGPANN outperforms CNE. According to the obtained solutions from CGPANN, it shows that all of the solutions do not have the conventional ANN structure based on the layers with a number of nodes per layer, which demonstrates that the network structure optimisation approach is able to obtain effective ANNs whose structure is not considered by human designer.

- Relative importance between them?

Apart from the comparison between structure optimization and weight optimization, the relative importance between them is still an open question. In the literature of NE area, it is assumed that network structure optimization has significant benefit to the evolutionary search. However there is currently no literature that explores the relative importance of network structure optimization to the connection weight optimization. In this case, [30] also investigated the relative importance between them by conducting experiments based on CGPANN in three different scenarios:

1) Only evolving connection weights.
2) Only evolving network structure.
3) Evolving both of them.

In the first case, the network structure is initialised randomly but remains fixed and only the connection weights will be evolved. In the second case, the connection weights are initialised randomly but still remain unchanged and only the network structure will be evolved. And the last case will evolve both of randomly initialised weights and structure. The experiments will be conducted based on a range of benchmarks and the final fitness will be utilised to investigate the relative importance between them.

As can be seen from the results, [30] mentions three interesting features. Firstly, evolving connection weights with random fixed network structure significantly performs worse than evolving network structure with random fixed connection weights with medium or larger effect size. This result indicates that evolving network structure may be more important than evolving connection weights for ANN training. Secondly, evolving both of weights and structure significantly performs much better than just evolving weights with large effect sizes. This result implies that evolving the network structure may have a large impact on evolutionary search. Finally, there is little difference between the performance of evolving both and just evolving structure, which means the evolution of connection weights actually has no such impact as the evolution of network structure.

In conclusion, evolving network structure has a more significant influence on the evolutionary search than just connection weight evolution for ANN training at least for CGPANN compared to CNE. Moreover, the comparison results also indicate that the evolution of network structure may be more important than the evolution of connection weights for the evolutionary search. Although random fixed structure and random fixed weights may be never utilised for real applications, those results still demonstrate the relative importance of network structure and connection weights evolutions for ANN training in NE area [30].

2.4.4.4 Heterogeneous ANN evolution

According to section 2.4.4.3, network structure optimisation outperforms connection weight optimisation at least for CGPANN with CNE in three different benchmarks. However both of these two approaches utilise the same fixed neuron transfer function for each ANN, which is the homogeneous ANN. At present, it is not clear whether evolving heterogeneous ANN with more than two types of neuron transfer functions indeed has any benefit for NE [40] [61] [52]. [52] mentions that there is relatively little research which evolves node transfer function rather than the network structure. [40] also indicates that the current researches of ANN focus on the learning algorithms and architecture, where the importance of transfer function is ignored. What's more, [61] further demonstrates that in terms of complex problems, the evolution of transfer functions displays little benefit to improve the ANN performance. Therefore, there is currently no literature that conducts a fully investigation on the creation of heterogeneous ANNs with the transfer function evolution in NE area [30]. In this case, [30] conducts a comparison between CNE and CGPANN in terms of evolving homogenous and heterogeneous networks respectively based on five different benchmarks.

This comparison in [30] is conducted with two steps for evolving homogeneous and heterogeneous networks respectively. In terms of evolving homogeneous networks, different types of neuron transfer function are available for implementation but only one type will be utilised for ANN evolution to conduct the benchmarks based on CNE and CGPANN respectively so as to achieve the homogeneous network optimisation. This work investigates how different types of neuron transfer functions influence the homogeneous ANN performance and the results demonstrate that the selection of neuron transfer functions has a large impact on the ANN's performance for both of CNE and CGPANN.

On the other hand, heterogeneous networks will also be evolved based on CNE and CGPANN respectively for the same benchmarks. The results will be compared with the evolved homogeneous networks' performance in order to see whether evolving heterogeneous networks has better performance. The results of this comparison indeed demonstrate that evolving heterogeneous network outperforms evolving homogeneous network in the majority of cases, which indicates that evolving heterogeneous network could be a better choice for ANN optimisation at least for CGPANN and CNE unless a suitable neuron transfer function type is known for the given task [30]. Furthermore, [30] mentions that when evolving homogeneous network, the type of neuron transfer function is always not known in advance with respect to the given task. That is to say, the selected neuron transfer function has to be random. However evolving heterogeneous network could select the suited neuron transfer functions during the evolution, which removes the requirement to know a suited neuron transfer function in advance and that could be the most important benefit of evolving heterogeneous ANN [30].

2.4.4.5 Explicit genetic redundancy

Genetic redundancy is also an important feature of CGP and CGPANN, even of the original GP. Actually genetic redundancy is a well-studied topic in evolutionary algorithms(EA) [79]. In GP, genetic redundancy refers to the genes which do not have any contribution to the phenotype output. A typical example is when a section of genes is multiplied by zero, in this sense that section of genes has no influence on the computation of phenotype output. And this type of genetic redundancy is considered as implicit genetic redundancy [30] [29]. Apart from the standard GP, there is another type of genetic redundancy which exists in other forms of GP. In this kind of genetic redundancy, there are some genes which are removed during the decoding of genotype into phenotype such as inactive genes in CGP. This form of genetic redundancy is called explicit genetic redundancy since it removes the redundant genes explicitly during the mapping from genotype to phenotype, which is an important feature in CGP [30] [29].

One typical utilisation of explicit genetic redundancy is preserving the neutral genetic drift. Neutral genetic drift was first proposed and discussed by [80] in the area of evolutionary biology. Neutral genetic drift means that a genotype created by the neutral mutation is preserved through the selection into the next generation [30]. In this case, the genotype is drifting in an unguided way through generations but the drift genes have no influence on the phenotype due to the neutral mutation. Based on different types of genetic redundancy, there are two different types of neutral genetic drift. Implicit neutral genetic drift describes the drift based on implicitly redundant genes whereas explicit neutral genetic drift based on explicit redundant genes [30]. One of the significant benefit of neutral genetic drift is that it ensures the genotype diversity among population and that is helpful for escaping from local optima during the evolution [81] [82]. However, a typical difficulty in studying neutral genetic drift is how to identify which part of genes is redundant. In GP, it is challenging to make sure which part is implicit genetic redundancy since it involves learning how the phenotype contributes to the program output [83]. However in CGP, it is not a difficult task to identify which part of genes is explicitly redundant. If the genes associate a node that does not connect any inputs to outputs in the phenotype, then these genes are explicitly redundant [30]. Because of this benefit, neutral genetic drift has become a widely studied area in CGP. Some works just examined whether CGP benefits from neutral genetic drift by preventing children selected over parents for identical fitness in order to disable neutral genetic drift [29] [81]. Some works investigated whether increasing explicit genetic redundancy would benefit to CGP by increasing the number of available nodes in order to enlarge the portion of inactive genes [74]. [30] summarized his work and presented some new findings based on the previous studies about neutral genetic drift in CGP. [30]mentioned that the explictly genetic redundancy has significant further advantages in explictly neutral genetic drift for CGP since it’s far easier to be controlled than implictly genetic redundancy, which makes the study of explicit neutral genetic drift much simpler. Moreover, [30] also mentioned that preserving the neutral genetic drift not only helps the evolution escaping from local optima but also presents an ability to aid the evolutionary search even if the evolution is not trapped in local optima [81] [82]. Finally, [30] talked about how explicit neutral genetic drift influences Cartesian Genetic Programming of Artificial Neutral Networks (CGPANN). However the results showed no benefit of considering explicit neutral genetic drift into CGPANN. A possible explanation is that the additional weight genes in CGPANN lead to finer mutations to take place than CGP, which indicates that the evolution may less likely get stuck into local optima. So that may be the reason why preserving explicit neutral genetic drift has not demonstrated any benefit in CGPANN [30].

2.4.4.6 CGPANN in controller structure optimisation

One typical application of CGPANN is the controller design for a dynamic control system. [84] investigates how CGPANN will perform to evolve a ANN controller in order to achieve a double pole balancing task. The double pole balancing task belongs to a typical example for designing a control system, where the hinged poles need to be balanced on a wheeled cart with a finite length track. The objective is to make sure that the angle of poles is maintained within a threshold, otherwise task will be failed. In terms of the ANN, the inputs of the network are the pole-angle, velocity of the poles, position of the cart and the velocity of the cart. The ANN output is the force applied to the cart to make sure that the poles could be balanced for 30 minutes. The result shows that CGPANN spends much less evaluation numbers than other NE approaches on the same double pole balancing tasks including CNE, SANE and NEAT. Furthermore, the generalisation of the evolved solutions by CGPANN is also investigated where 28 evolved solutions are tested for 625 different random initial states for the double pole balancing task. The result shows that 532 out of 625 initial states are successful to achieve the double pole balancing task, which indicates that the evolved solutions present effective general behaviour [84]. As can be seen from this work, CGPANN demonstrates effective performance to design a dynamic control system to achieve the double pole balancing task, which displays the capability of CGPANN in the control system design field.

Apart from the control system design, CGPANN also demonstrates effective performance in the pattern recognition field. [85] investigates how CGPANN is implemented for the diagnosis of Breast Cancer from the FNA (Finite Needle Aspiration) data samples. CGPANN in this work is used to classify the data set in order to diagnose whether it is benign or malignant. Firstly, CGPANN needs to be trained based on training data set and then CGPANN will be applied for the diagnosis with unseen FNA data set in order to do the classification. The result shows that the best evolved solution achieves 99.5% successful rate for the training with 200 cases. Moreover, this solution also achieves a 98% accuracy rate for the diagnosis based on new 200 cases, which demonstrates a quite high accuracy for the Breast Cancer diagnosis [85]. This work presents how CGPANN is used for the medical diagnosis based on the data classification, which demonstrates the effective performance of CGPANN applied in the pattern recognition field.

Besides the control system and pattern recognition, [86] demonstrates how CGPANN performs for a prediction task of forecasting the foreign exchange rate. In this work, CGPANN is added with recurrent connections which create RCGPANN implemented for the task. This work uses 500 days of the historical data of foreign exchange from the Australian Reserve Bank to train the ANN for the forecasting model in order to predict the 11th day exchange rate based on the 10 days of historical data. In terms of the testing, 13 different currencies are used for the historical data spanning 1000 days. The evolved ANN is assessed by comparing the estimated values with the actual values from the known historical data. The result shows that the evolved ANN model achieves a 98.872% accuracy rate for this test. Moreover in terms of more than a single day's data rate in advance, the evolved ANN model produces a 92% accuracy rate when the currencies' exchange rates are up to 1000 days (4 years) in advance for the prediction [86]. This work demonstrates how CGPANN is implemented in another field of currency exchange rate prediction with added recurrent genes for the ANN structure encoding. And the result indicates that RCGPANN produces a high accuracy rate for the prediction, which displays an effective performance of CGPANN in the prediction modelling field.

As can be seen from these three works, CGPANN produces effective performance as a controller, a classifier and a predictor. However unlike NEAT, CGPANN hasn't been employed into robot control field as well as the fault tolerant control. Even though, it is still quite interesting to investigate how CGPANN could perform to design a fault tolerant robotic controller in the network structure optimisation space.

2.4.4.7 Summary

This section reviews CGPANN and its benefits in the NE field. The most significant benefit is that CGPANN outperform CNE in a series of benchmarks, which indicates that network structure optimisation approach could produce better performance than just connection weight optimisation approach at least for CGPANN and CNE. In addition, CGPANN demonstrates great resilience to the program bloat and the benefit of evolving heterogeneous networks rather than homogeneous network. Apart from those benefits, explicit genetic redundancy is also an important feature of CGP and the literature demonstrates that preserving neutral genetic drift based on the utilise of explicit genetic redundancy is helpful for the evolutionary search especially to help escape from the local optima during the evolution, although CGPANN has not benefit from it. Finally, CGPANN is effective in a wide application field including the design of controller, classifier and predictor. However, CGPANN has not been investigated in the robot controller optimisation field as well as the fault tolerant control. Even though, CGPANN could still be a promising approach to design a fault tolerant robotic controller based on the optimisation of a structurally evolved heterogeneous ANN controller.

2.4.5 Comparison between CGPANN and NEAT

[73] conducts a comparison between CGPANN and NEAT in terms of the double pole balancing benchmark experiment referred from [54]. Double pole balancing belongs to a typical control problem with only one controller output, whose task is to balance two poles attached to a cart. The result shows that CGPANN needs much less evaluation numbers than NEAT and even SANE, which could demonstrate that CGPANN produce a better performance than NEAT in terms of the convergence for the double pole balancing benchmark. However there is still some weakness in this comparison. A crucial problem is that NEAT utilises a slightly modified sigmoid transfer function to conduct the experiment [56] rather than a normal sigmoid transfer function used in [73]. So this difference might have some influence on the performance comparison, but whether the influence is significant is unknown currently. Even though, [73] is the only work that compares CGPANN with NEAT and other NE approaches such as SANE in terms of the double pole balancing control problem. And the result could demonstrate that CGPANN produces a better performance than NEAT regardless of the slightly modified transfer function problem.

2.4.6 Comparison between EA and NE

Actually, ANN has demonstrated effective performance in wide field not only for the controller design but also for the modelling. [87] and [88] demonstrate the effectiveness of a feed-forward network of logistic sigmoid function and radial basis function respectively for universal function approximation based on a finite number of neurons. Moreover, [89] and [90] both demonstrate the capability of recurrent ANN used for universal dynamical system approximations. Those works indicate that both of feed-forward ANN and recurrent ANN are effective for a wide application range besides the controller design, as long as the ANN could be trained sufficiently no matter in a traditional way or with NE approaches.

On the other hand, structure optimisation based EA approaches also demonstrate effective performance in a wide application range according to reviewed works in chapter 2.3. However there is very little work that really compares EA with NE in terms of the structure optimisation applications. Currently, [91] is the only work that investigates the performance difference between EA and NE approach based on a comparison between GP and GP-Artificial Neural Network (GP-ANN) in terms of the system modelling problem. This work utilises both of GP and GP-ANN for the formulation of mathematical models for vibratory finishing process. The result shows that GP-ANN performs better than GP in terms of modelling accuracy where GP-ANN produces more accurate generalised models. Moreover, when the data samples are few and there is a requirement for rigorous tuning of GP parameters so as to obtain the optimal model, the computation effort will be increased significantly. In order to avoid the high cost analysis, ANN also demonstrates effective performance to improve the modelling accuracy utilised either in parallel or as compensation to the GP model [91].

As can be seen from this work, one significant benefit of NE approach is the more accurate modelling result compared to just EA based result in terms of the structure optimisation problem. Although this work is conducted based on the comparison between GP and GP-ANN, it at least demonstrates that GP-ANN produces a better performance than just GP in the system modelling field, which also indicates the importance of ANN in this task. Nevertheless, this work still demonstrates the benefit of NE over EA for the structure optimisation problems. For this reason, the structure optimisation based NE approach could be considered as a primary scheme utilised for designing a fault tolerant control system in terms of robotic tasks.

2.5 Multi-objective evolutionary algorithms

The multi-objective optimization algorithm (MOEA) was developed driven by the need of multiple objectives in a problem where a set of optimal solutions, which are known as Pareto-optimal solutions, would be obtained rather than a single optimal solution. In terms of the performance measure, convergence to the Pareto optimal front and maintenance of solution diversity are two essential indexes for multi-objective optimisation [92]. Although [39] mentions that parallel evolution algorithms with migration topology is able to simultaneously increase the selective pressure in some populations and improve the diversity of other populations, it is still focused on single objective optimization rather than multi-objective optimisation.

Up to now, there are several MOEA approaches developed [93] [94] [95] [96]. Among them, the non-dominated sorting genetic algorithm II (NSGA2) [92] and the pareto archived evolution strategy (PAES) [97] are two well-studied MOEA algorithms for parameter optimisation. Both of them belong to elitist MOEA and utilise crowding measure to encourage the population diversity [12], which will be talked about in section 2.5.1. In terms of the structure optimisation, multi-objective genetic programming (MOGP) and multi-objective Cartesian genetic programming (MOCGP) are two well-studied approaches based on GP and CGP respectively for the genome encoding with multi-objective optimisation, which will be talked about in section 2.5.2.

2.5.1 Parameter optimisation approach

2.5.1.1 NSGA2

NSGA2 was developed by [92] for the multi-objective optimisation in the parameter space. NSGA2 works in a common elitist EA loop based on GA for the genome encoding but with a capability to evaluate the individuals for multiple objectives. The elitism works during the survival selection where parents and children are combined together in order to be sorted and then survived. The main difference between NSGA2 and GA is that each individual in NSGA2 will be evaluated with multiple objectives. In this case, each individual will be set two extra attributes: the ranking number and the crowding distance, which will be used for the parent selection and the survival selection, rather than just a fitness value in GA.

The ranking number is obtained by the non-dominated sorting where each individual will be compared with each other in order to check what the dominance relationship is between them. The ranking number is decided by their dominance levels. So if the individual is not dominated by anyone in the population, it is ranked in the first front. And if the individual is dominated by everyone in the population, it is ranked in the last front. Especially, if the individuals are non-dominated between each other, they will be set a same ranking number. Based on the ranking number, individuals can be sorted in different fronts so as to achieve the parent selection and survival selection, where the individuals are selected if they are in a prior front. However there is a problem when the individuals are located in the same front. In this case, they cannot be distinguished and that is why crowding distance is required.

Crowding distance was developed along with NSGA2, which belongs to a kind of crowding measure approaches. Crowding distance is working in the fitness domain of each individual. It gives each individual an extra attribute which describes how the individual is crowded with its two neighbours. So if the crowding distance is larger, it means the individual is less crowded. To be specific, crowding distance measure works in two steps: the density estimation and the crowded comparison operator. The density estimation measures the density of a certain solution in the population based on the calculation of the average distance of two points on either side of this point for each objective. This density value is calculated by "the estimate of the perimeter of the cuboid formed by using the nearest neighbours as the vertices (call this the crowding distance)" [92]. The process of crowding distance calculation is shown in Figure 2.18. In Figure 2.18, the length of the dashed box formed by the cuboid is the density estimation result for solution i in its front, where solutions i-1 and i+1 serve as the solution i's nearest neighbours to act as the vertices of the cuboid. When density estimation is completed for each individual, the crowded comparison operator guides the selection to generate a spread set of Pareto-optimal solutions. In this case, when two solutions belong to different fronts, the one with the better rank will be selected. If they belong to the same front, the one with the less crowded region will be selected. In NSGA2, crowding distance measure will also be utilised in the two steps along with ranking number: the parent selection and survival selection. During the parent selection, crowding distance measure will help tournament selection to select the parent from two candidates if they are non-dominated to each other. And during the survival selection, crowding distance measure will help individuals from the same rank to be sorted and then filled into the next generation until the new population is full. For this reason, crowding measure could also maintain the population diversity in the fitness domain which is relevant for multi-objective optimization [98].

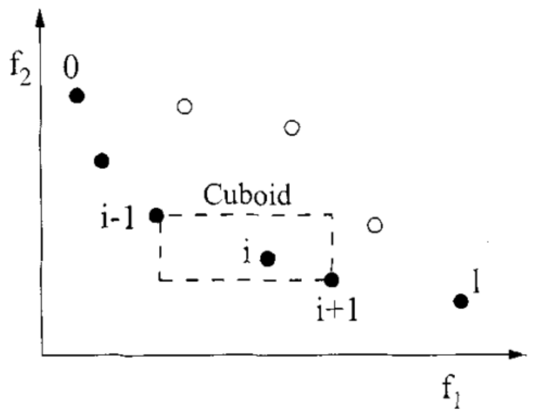

Figure 2.18: Crowding distance calculation [92].

2.5.1.2 PAES

PAES was developed in [97] based on a (1+1) ES. In each generation, PAES creates just one candidate solution based on the mutation of the current solution. After the evaluation of the candidate solution, it will be compared with the current solution based on the dominance relationship. If the candidate solution dominates the current solution, the candidate solution will be accepted as the next current solution and vice versa. However if they are non-dominated between each other, the candidate solution will be compared with the archive solutions which are the best solutions found so far. If the candidate solution dominates any member of the archive solution, the candidate solution will be accepted as the new archive solution and all the other solution will be eliminated from the archive and vice versa. If the candidate solution is still non-dominated with the solutions in the archive, there are two options. If the archive is not full, the candidate solution will be just added to the archive. Else if the archive is already full, the one in the most crowded region will be eliminated. Finally, the candidate solution will still be compared with the current solution in the case that they are non-dominated. If the candidate solution resides in a less crowded region than the current solution, the candidate will be accepted. Otherwise, the candidate solution will still be rejected. This is the whole process of the evolution loop for PAES [97].

In terms of the crowding measure, it is different from that in NSGA2. The crowding measure in PAES is computed based on the solutions' grid location, which is determined by the number of objectives for the problem. The grid location of each solution is generated using recursive subdivision and noted using a tree encoding. A map of the grid is also maintained in order to indicate how the solutions are located in the current archive. In this case, the solutions will be located in a deterministic and pre-specified number of equal-sized cells, where the crowding measure will be working based those cells in the search space [97]. The details of how this crowding measure is utilised in PAES can be referred to [97].

2.5.1.3 Comparison between NSGA2 and PAES

As can be seen from section 2.5.1.1 and 2.5.1.2, although NSGA2 and PAES both belong to the elitist MOEA approaches for the parameter optimisation, they are actually quite different. The main differences include the difference between GA and ES for the population composition, whether the archive is used to preserve elitist individuals and the difference between the crowding measures utilised in these two approaches.

Nevertheless, [92] still conducts a comprehensive performance comparison between these two approaches based on nine different test problems where the experiment details can be referred to [92]. The comparison result shows that NSGA2 performs better than PAES in terms of the diversity preserving mechanism. However, PAES outperforms NSGA2 in terms of the convergence where the non-dominated solutions found by PAES are able to get closer to the true Pareto optimal front than the solution found by NSGA2. One exception is that NSGA2 performs better than PAES when the problem has strong parameter interactions. Furthermore, NSGA2 is also integrated with a simple extension for constraint multi-objective optimisation. And the result shows that this proposed constraint handling mechanism produces more effective performance to solve four different problems than the approach developed in [99]. In conclusion, either NSGA2 or PAES has its own benefit for multi-objective optimisation and NSAS2 is also integrated with an effective constraint handling strategy.

2.5.2 Structure optimisation approach

2.5.2.1 MOGP

Except for NSGA2 and PAES utilised for the parameter optimisation in the MOEA field, multi-objective genetic programming (MOGP) is a typical approach that works for the structure optimisation based on GP. [100] develops a MOGP algorithm based on the integration of GP and NSGA2 utilised for software development effort estimation. This MOGP implements GP for the genome encoding but the whole evolution process is totally the same as NSGA2 based on the ranking and crowding distance instead of just fitness values for the population sorting in order to conduct the parent and survival selection. The crossover and mutation are the same as that employed in GP. The evolution will be stopped if a user set termination criteria is achieved [100]. [101] develops a MOGP algorithm for the figure-ground image segmentation. This MOGP is called non-dominated sorting genetic programming (NSGP), but it is also created based on NSGA2 with GP instead of GA for genome encoding and genetic operator. [102] also develops its own MOGP algorithm for the classification with unbalanced data. Again, this MOGP still utilises NSGA2 for the multi-objective optimisation process including the same dominance ranking and crowding distance as each individual’s attributes in addition to their fitness. [103] presents a scheme of how the UAV navigation controller is designed based on multi-objective GP approach. In this approach, four fitness functions are used for multi-objective GP to evolve controllers to respectively locate three different radar sources. Three different goals need to be satisfied for each type of radar including moving toward the emitter, circling the emitter closely and flying in an efficient way. Four fitness functions can be used to describe the three different goals which are normalized distance, circling distance, level time and turn cost. The MOGP algorithm is still created based on NSGA2 whereby the non-dominated sorting is an effective method to rank solutions in terms of each fitness functions to achieve multi-objective optimization [103].

As can be seen from these works, all of them utilise NSGA2 for the multi-objective optimisation process based on GP for the genome encoding and genetic operator, which produce effective result for obtaining a Pareto optimal front of solutions in terms of corresponding objectives. A possible reason of implementing NSAG2 for the multi-objective optimisation process is that NSGA2 is currently one of the well-studied MOEA algorithms, which outperforms other approaches in the parameter optimisation area [92]. And that may be the reason why NSGA2 is effective to be selected for GP based MOEA algorithm development in these works.

2.5.2.2 MOCGP

Except for MOGP, there are several works that try to develop their own MOCGP algorithm. [104] develops a MOCGP algorithm based on CGP and NSGA2 for the circuit approximation. In this MOCGP, the (1+λ) ES is replaced by the normal population size used in NSGA2 where same number of parents create the same number of children. The non-dominated sorting procedure in NSGA2 is also modified in a way that "when all components of the fitness score of a parent and its offspring remain unchanged, the offspring is classed as dominating the parent, and is therefore ranked higher than the parent" [104]. However this modification is quite unclear that it does not mention what the components are for the fitness score and why the children will dominate parents when the fitness score remains unchanged. Even though, the MOCGP developed in this work still demonstrates effective performance to successfully approximate circuits including adders and multipliers. [105] develops its own MOCGP algorithm to improve the circuit design as well. This MOCGP is also integrated based on NSGA2 and CGP. However this work does not mention how the population is composed such as whether it still utilises ES or not. The only point it mentions is that this MOCGP will use a large population size in order to create a Pareto front for different objectives. Even though, this work still shows that MOCGP is effective to design the circuit especially for multi-objectives including the optimisation of gate count and path length.

As can be seen from these works, MOCGP is also developed based on the integration of NSGA2 and CGP, which further indicates that NSGA2 is well suited for the integration of a multi-objective optimisation algorithm even in the structure optimisation field.

2.5.3 Survival selection

Survival selection in EA is a necessary stage for ensuring the elitism strategy not only in single-objective optimization but also in multi-objective optimization. The survival selection is actually a method to ensure that the elitist will be always preserved. In order to do this, the children need to compete with their parents to make sure that the best current individuals or the so called elitists can be survived into the next generation [29] [92] [11].

As can be seen from section 2.5.1.1, the survival selection in NSGA2 works in two steps based on the ranking number and crowding distance. Firstly, the children will be merged with the parents based on the non-dominated sorting. After the sorting, the individuals can be survived based on their rankings. However when the number of individuals in the current same rank is larger than the left available survival places, crowding measure will be utilised to distinguish them and sort them until the next generation is full. The crowding distance measure utilised for survival selection in NSGA2 is called crowding fill [92].

Similar to NSGA2, PAES has its own survival selection. According to section 2.5.1.2, PAES compares the candidate solution with the current solution firstly. If they are non-dominated between each other, the candidate solution will also need to be compared with the archive solutions. And if they are still non-dominated, crowding measure will be conducted to distinguish them, which is also a kind of crowding fill [97].

As can be seen from these two works, crowding fill is actually an essential stage in the survival selection, which further distinguishes the individuals who are located in the same front but still need to be survived sequentially. In this case, crowding fill based on different crowding measures would be important for a wide range of MOEA approaches where the individuals are ranked based on non-dominated sorting.

2.5.4 Population diversity

Population diversity is always a key issue in EA to overcome premature convergence problems by means of escaping from local optima. Moreover, in multi-objective optimization, maintaining high diversity is also significant to ensure that the Pareto front is large enough to reflect the trade-off among different conflict objectives [98].

2.5.4.1 Genotype diversity

Genotype diversity is a kind of population diversity which considers the diversity in the genomes among each individual. A famous approach to measure the genotype diversity is to measure the distance between individuals [106], or in another word: genetic distance. Genetic distance was firstly proposed by [107] in biology area. However it has the same definition in the EA area that it just demonstrates the distance between individuals in genotype domain [106]. There are actually different ways to evaluate the genetic distance between the current individual and the one that is compared with, such as the mean spatial position of the population [98] [108] [109], the position of the fittest individual [110], or the position of each individual [111]. But no matter which position the individual is compared with, Euclidian distance is the most common approach to estimate genetic distance not only in EA [106] but also in biology area [112].

Hamming distance is another approach for measuring the genetic distance, which was first introduced in [113]. The primitive usage of it is to check the difference between two words in fixed length. In this way, Hamming distance will reflect how far the two words are in terms of how many entries are different in the corresponding positions between each other [114]. Based on this technique, Hamming distance has become an essential metric in coding theory, such as error correcting code. The creation of error correcting code is due to the data transmission in the information technology. The data being transmitted are in the form of binary string, so there will be a chance that some unavoidable error occurs during the process of transmission. As a result, the received data may be different from the original ones. In this sense, an error correcting code was developed based on Hamming distance in order to detect and correct the erroneous messages during the data transmission [114]. The main working mechanism of Hamming distance is based on the computation of an Exclusive-Or operation (XOR, for short) between two binary strings. As a result, the number of the sum of ones will be the number of different bits between each other [115].

2.5.4.2 Phenotype diversity

Besides genotype diversity, phenotype diversity also has its own impact in evolutionary algorithms. A typical example is the crowding measure mentioned in NSGA2 which is actually working in the fitness domain. The crowding measure encourages the phenotype diversity among the individuals in the current front and the individuals will be selected based on its crowding distance from largest to smallest for both of parent selection and survival selection in addition to the non-dominated sorting [92].

Just like the importance of genetic distance used to measure the genotype diversity, fitness distance will be also a possible method to demonstrate the phenotype diversity in the fitness domain. However, fitness distance is actually not a common metric used to measure phenotype diversity. The phrase of fitness distance is always appeared in the problem of fitness distance distribution [116]. Suppose a global optimum is known before GA is executed and the Hamming distance can be used to measure the genetic distance between the current individual and the global optimum if the genome is encoded in binary string. And the fitness value of this individual can be also computed along with the Hamming distance. If the computational effort is sufficient, all the possible genotypes can be presented by the distance and fitness values so that a fitness distance distribution can be displayed [116]. With the help of fitness distance distribution, the GA can be demonstrated to be effective if the Hamming distance is becoming smaller while the fitness is becoming better [117].

However the fitness distance distribution is about the relationship between fitness and genetic distance during evolution, so it is not quite helpful for measuring the phenotype diversity. Even so, the fitness distance may still be a possible metric to measure the phenotype diversity among individuals just like how crowding measure works based on crowding distance among individuals to measure the population diversity. So the usage of crowding measure actually provides a possibility to apply fitness distance to measure phenotype diversity.

2.5.5 Comparison between multi-objective and single objective optimisation

Generally speaking, multi-objective optimisation based EA approaches would have more solution options than single objective optimisation approaches based on the Pareto optimal front to deal with more than just one objective. However, it does not mean that single objective optimisation is not able to obtain a solution that could be suited to different objectives. [118] mentions that the classical approach of using single objective optimisation to deal with multiple objective problems is to develop a tailored fitness function that aggregate different objectives in some way to create a single-valued function, which can be optimised by the single objective optimisation approaches. In terms of the aggregation strategies for multiple objectives, the weighted sum is often utilised to create the single-valued function. However, a serious drawback of the weighted sum strategy is that this strategy may be arbitrary and the weight value is also hard to determine before running the evolution. On the other hand, multi-objective optimisation does not need such a choice since it could let the evolution explore different trade-offs between different objectives and the designer can choose any solution from the obtained Pareto optimal front in terms of the task requirement without determining which objective is more related to the problem in advance [119]. Although the weighted sum based single objective optimisation approaches have some limitations compared to multi-objective optimisation approaches, [119] still presents a comprehensive comparison between these two approaches in terms of two typical robotic tasks based on the robot controller evolution for multiple objectives including the maze navigation task for a single robot and the flocking task for swarm robots.

In terms of the maze navigation task, one objective is to make the robot move straight and fast and the other objective is to keep the robot away from the obstacle and walls. In terms of the flocking task, one objective is to reward the group motion based on the maximisation of the swarm robots' displacement measured from the centre of mass of the group. And the other objective is to maximise the cohesion whereby the average distance of the robots from the centre of mass of the group should be minimised. Both of these two tasks will be utilised for the robot controller evolution based on their corresponding multiple objectives. As a comparison, multi-objective optimisation approach will be compared with weighted sum based single objective optimisation approach in terms of these two objectives for each of these two tasks. The experiment result shows that multi-objective optimisation approach outperforms weighted sum based single objective optimisation approach for robot controller evolution based on these two objectives for each task. Both of these two tasks demonstrate that multi-objective optimisation attains a much wider solution region than weighted sum based single objective optimisation. This result is not surprised since the main advantage of multi-objective optimisation is the capability to explore a wide objective space where single objective optimisation is not able to achieve. However, this work further demonstrates that even a weighted sum based single objective optimisation for multiple objective problems is still not able to obtain that wide objective space despite that a variety of weight values has been tested for it. The only exception occurs when the weight value is set 0.5 for the maze navigation task. For this scenario, the obtained solutions are located in the most trade-off region where each objective is maximised at the same time, which achieves the equivalent performance as multi-objective optimisation if the most trade-off solutions are required for this task. However, that is the only equivalent performance that weighted sum based single objective optimisation is able to achieve compared to multi-objective optimisation, which also indicates that the solutions evolved by multi-objective optimisation have a larger behaviour diversity than that evolved by weighted sum based single objective optimisation [119].

In conclusion, the comparison result of [119] demonstrates that multi-objective optimisation approach is able to produce better performance than weighted sum based single objective optimisation approach for multiple objective problems due to the capability of multi-objective optimisation to achieve a wide exploration of the objective space to meet different solution requirement. Although only maze navigation and flocking tasks are tested as the comparison between these two approaches in this work, it at least demonstrates that multi-objective optimisation approach outperforms weighted sum based single objective optimisation approach to deal with multiple objectives for these two typical robotic tasks. For this reason, multi-objective optimisation could be considered as the main approach in the case that multiple objective problems are needed to evolve robot controllers rather than a tailored single-valued fitness function based single objective optimisation, such as the weighted sum approach.

2.6 Convergence criteria

2.6.1 Termination condition

Convergence criteria is actually used to decide when to stop the evolution in EA [120]. As mentioned before, a common knowledge about the description of convergence was raised by [121] where the best performance values have been stabilized after a particular time. What's more, [11] mentioned that a known optimum may not be a good choice to terminate evolution. Since EA is a stochastic optimization technique, so there is no guarantee that the known optimum will be reached. In this case, the termination condition may not be satisfied and the evolution will never stop [11]. For this reason, [11] presented a list of options of how to certainly terminate the evolution:

1. The allowed CPU time has reached
2. The number of fitness evaluation reaches the highest limit
3. The fitness improvement reaches the lowest limit within the given period of time such as the number of generations or fitness evaluations
4. The population diversity drops below the lowest threshold

[122] also presented two similar options of termination conditions which are the upper limit of number of generations or fitness evaluations has reached and the opportunity to get a significant improvement in next generations is quite low. And [123] gave some comments on these two options. [123] mentioned that the first option needs some knowledge about the maximum search ability while the second one does not. In the second one, there are two different types to represent termination conditions including genotype and phenotype termination criteria. Genotype termination condition means that when a large enough proportion of genes have converged to a certain value among the whole population, the evolution could stop. And phenotype termination condition is expressed in the fitness domain that when the average fitness exceeds a pre-set threshold, the evolution will be terminated [123].

However no matter which termination condition is used for EA, most of the multi-objective optimization convergence measurements still rely on the true Pareto-optimal front [92] [124] [125] or a surrogate of the true Pareto-optimal front obtained by multiple runs if the true front exists but is unknown [126]. If the current Pareto front is approaching the true Pareto-optimal front, the algorithm is supposed to be converged. This problem is quite similar to that in single objective optimization where the true Pareto-optimal front may not be reached at the end because of the stochastic feature in EA. Moreover if the true Pareto-optimal front cannot be obtained before the evolution loop starts, there is no way to estimate the termination condition.

In this case, a new convergence measurement was developed by [127] which just relies on the current Pareto front to terminate the evolution. In this work, the convergence of a multi-objective optimizer is based on the track of the improvement of the number of non-dominated solutions in the population. To quantify the convergence criteria, [127] proposed two metrics which are consolidation ratio and improvement ratio. The consolidation ratio is the proportion of the old solutions which are still remained non-dominated in the current population compared to the whole population. So when this proportion increases to a stable high value approximate 90%, the evolution could be stopped. The improvement ratio is the proportion of the old solutions that are still dominated by new solutions in the whole population. The evolution could also be stopped when this proportion decreases and maintains to a stable value [127].

2.6.2 Performance measure for multi-objective optimisation

Although this mentioned work [127] about the convergence criteria has been discussed to deal with termination condition problem without the need of the true Pareto-optimal front, it has no certain metric to really reflect the performance of a multi-objective optimizer. At the moment, hyper-volume indicator is currently the only known metric to display the performance of a multi-objective optimizer [128] and it is also the only unary indicator which represents the performance in unary values [129]. The indicator is usually used to estimate the goodness that how the current Pareto front is approximated to the true Pareto-optimal front. Moreover, the indicator can be also used for selection since it can measure the distribution of each individual across the current front. [130] firstly applied an indicator into the framework of a multi-objective optimizer in order to do the selection. Hyper-volume indicator was firstly proposed by [131] where it measures the volume covered by all the Pareto-optimal front solutions with a user-defined reference point. The reference point is normally selected as the nadir point of the investigated Pareto front [132] [133] or a point that is slightly worse than the nadir point [134] [135]. In recently years, hyper-volume indicator has become one of the most used techniques among indicators [124] [126]. The hyper-volume based approaches have also been studied extensively where the aim of these approaches are trying to demonstrate the largest hyper-volume of the non-dominated individuals in the current Pareto-optimal front in order to see how it is approximated to the true front [125] [136] [126]. Hyper-volume indicator can be not only used as an offline indicator to evaluate a multi-objective optimizer [134] but also as an online indicator to lead the evolution process [137] [136] [138]. The significant benefit of using hyper-volume indicator is that it is strictly Pareto compliant, which means the hyper-volume will always reflect the size the Pareto-optimal front no matter how many fronts it dominates [126]. Nevertheless, one serious drawback is that the reference point needs to be selected accurately, otherwise it will become an arbitrary point depending on the magnitude of the current Pareto-optimal front [126].

Although hyper-volume indicator is a well-established indicator to represent a front's quality, it is just discussed in statistics literatures [126]. To the best of our knowledge, there is no work proving that the hyper-volume based approaches work well for the convergence analysis when the true or surrogate Pareto-optimal front is not available to obtain. In this case, it would be worth investigating how hyper-volume indicator could be acted as a performance measurement to do the convergence analysis for a multi-objective optimisation algorithm in the case that the true Pareto-optimal front is not available to obtain.

2.7 Statistics analysis

2.7.1 Significant difference test

When the experiment data are obtained, a technique needs to be conducted in order to investigate whether the obtained data are statistically significantly different. In this field, Mann-Whitney U-Test and Vargha-Delaney A-Test are two famous approaches, which will be reviewed as following.

- Mann-Whitney U-Test

In order to see whether the obtained data difference is significant, the Mann-Whitney U-Test [139] would be an effective approach based on how much p value is. Mann-Whitney U-Test is used to check whether the null hypothesis can be rejected or not. The null hypothesis is that there is no significant difference between 2 data sets. If the p value is 5%, it means there is only 5% possibility that the null hypothesis can be accepted. That is to say, there is 95% possibility to reject the null hypothesis. So 5% is a criterion of p value, if p value is less than 5%, we can say that the null hypothesis can be rejected. In other words, there is a significant difference between 2 data sets [139].

- Vargha-Delaney A-Test

In addition, if the investigation is required to measure how large the difference is, Vargha-Delaney A-Test [140] can be used to quantify the difference based on the computation of effect size. The score of Vargha-Delaney A-Test normally returns a value between 0 and 1. If it returns 0.5, that means there is no significant difference between two data sets. Basically, the large effect size is set to 0.21, which means if the score is above 0.71 or below 0.29, there is a large difference between these two data sets. Moreover, 0.06 is set as a small effect size and 0.14 is set as a medium effect size, where the corresponding A-test scores will be 0.56 and 0.64 or 0.44 and 0.36 respectively [140]. In other words, the A-test scores between [0.36, 0.44] and [0.56, 0.64] belong to the small effect size, [0.29, 0.36] and [0.64, 0.71] belong to the medium effect size and the scores below 0.29 and above 0.71 belong to large effect size [140]. This analysis regulation is applicable for all the result analysis where A-test is utilised to estimate the significant difference between different experiment responses.

In a word, the A-test combined with U-test would be a meaningful way to judge whether two data sets are statistically significantly different and how large the difference is, which will be utilised for the required experiment result and discussion throughout the thesis.

2.7.2 Spartan package

[141] developed a package called Spartan, which was designed as a kind of statistical techniques used to help researchers investigate the relationship between their simulation and the real system. There are several techniques developed in Spartan including the cumulative mean approach to assess the sufficient number of experiment runs and the Parameter Robustness approach to investigate how parameter values affect the experiment responses.

- Number of experiment runs

[142] demonstrates a promising approach about how to select suitable number of experiment runs. This approach is based on confidence interval of the cumulative mean of the experiment result, which is a kind of statistical analysis method to estimate where the true mean value would be located. The narrower the interval, the more accurate the estimated data would be located. So if sufficient data are obtained to compute the confidence interval, the interval would become narrower and narrower until the user set criteria is met. In terms of the significance level, 5% is often selected. And that means there is 95% probability that the true mean value will be located in this interval. In other words, there is only 5% probability that the true mean value is not laid inside that interval [143].

In order to measure how narrow the confidence interval is, percentage deviation of either side of the interval against the cumulative mean value would be a solution. In this work, 5% of the percentage deviation is selected as a criterion to indicate the width of the confidence interval. So if the percentage deviation of the current cumulative mean value compared to either side of the interval reaches 5% and also remains below it, the current number of experiment runs would be sufficient and no more runs needed [142].

- Parameter values for experiment responses

Due to the uncertainty of parameter value, the Parameter Robustness technique can be utilised to examine how the parameter alteration influences on the simulation responses. If adjusting a parameter from baseline value has significant influence on the simulation output, then this parameter would be sensitive to the experiment and more efforts should be made to determine a suitable value for it [141]. Parameter Robustness technique only works for the independent parameters by tuning each parameter's value individually by 'one at a time' approach [144]. So when one parameter is being tuned, others remain the same on their baseline values. And when all the parameters are investigated, those simulation responses from the adjusted parameter values will be compared with the one from the baseline values, using the Vargha-Delaney A-Test [140]. In this sense, when there is large difference of the simulation response from perturbed parameter compared with that from baseline values, this simulation response would be sensitive to this perturbed parameter and a suitable value need to be figured out.

2.8 Summary

This chapter firstly reviews what fault tolerant control is and proposes that robust and switched fault tolerant control could both be considered to design fault tolerant robotic controllers in offline scenario. Secondly, different structure optimisation based EA approaches including GP, CGP and GE are reviewed respectively in terms of control system design and robotic controller evolution and all of them produce considerable performance in these two task domains. Next, NE approaches are also reviewed including two famous approaches NEAT/HyperNEAT and CGPANN. Both of them are working in the ANN structure optimisation domain with respective advantages. Especially, NEAT/HyperNEAT has demonstrated effective performance in the robotic controller optimisation field as well as the fault tolerance, which is quite related to this thesis topic. However, this work belongs to online robotic fault tolerant controller optimisation task, where the relatively long online evolution time is still an unavoidable problem rather than the offline fault tolerance scenario. On the other hand, CGPANN could still be a choice since its performance is actually unknown in terms of robotic controller optimisation, which worth a further investigation. Moreover, GP and GP-ANN are compared in terms of a system modelling problem. The result shows that GP-ANN produces a more accurate modelling result than GP, which indicates a potential benefit of NE over EA. Apart from that, different MOEA approaches are also reviewed including NSGA2 with PAES for the parameter optimisation based tasks and MOGP with MOCGP for the structure optimisation based tasks. In terms of MOGP and MOCGP, both of them are developed based on NSGA2 for the multi-objective optimisation, which further demonstrates the effective performance of NSGA2 in MOEA field. In addition, survival selection is reviewed based on different crowding fill strategies, which is a key step for elitism in MOEA. And then, different approaches to encourage the population diversity are also talked about since population diversity is also a significant research field in MOEA. Finally, weighted sum based single objective optimisation is compared with multi-objective optimisation in terms of multiple objective problems. Although weighted sum based single objective optimisation could produce a trade-off solution, its solution diversity is much lower than that of multi-objective optimisation, which is a significant drawback. Apart from the introduction of EA related approaches, convergence criteria topic is also reviewed along with the performance measure in terms of MOEA. And hyper-volume indicator is found to be a promising metric which could also be used to observe the convergence of MOEA. At last, statistics analysis approaches are also mentioned in order to test the significant difference between obtained experiment responses. Moreover, Spartan package is an effective statistics analysis tool which could not only estimate how many runs are sufficient to demonstrate the algorithm performance but also present a technique to help find out the best suited parameter values in order to obtain the optimal experiment responses.

The next chapter will present how CGPANN is implemented for a preliminary robot fault tolerant control experiment and investigate how a single objective optimisation algorithm will be working for evolving a fault tolerant robotic controller.

Chapter 3 CGPANN in fault tolerant control

3.1 Introduction

As is reviewed in the section 2.2, fault would be possibly occurred during the system operation, which could result in serious consequence such as a total failure of the system. In this case, a fault tolerant control system needs to be developed to tolerate the fault especially when it is occurred during the system operation. To be specific, robust fault tolerant control, which belongs to one of the passive fault tolerant control approaches, could be a first choice since only one robust controller is required to be evolved in order to tolerate the predicted fault without any concern about the fault diagnosis. In terms of the methodology, NE could be a better choice rather than EA due to its better performance shown from section 2.4.6. Moreover, CGPANN will be used as the main optimization approach instead of NEAT/HyperNEAT to design a structurally evolvable ANN controller in order to achieve the fault tolerant control due to the benefit of utilising explicit genetic redundancy in CGPANN as reviewed in section 2.4.4.5. However the reason why explicit genetic redundancy is significant for this work will be talked about in section 4.2.2 where crowding measure may have a problem for multi-objective optimisation and the utilisation of explicit genetic redundancy could be a solution for it. The details can be referred to section 4.2.2. It needs to note that the whole work is based on the optimisation of a structurally evolvable controller to achieve fault tolerance, so no fault compensation loop is required for this work and the whole work will just concentrate on the design of the controller in order to develop a fault tolerant control system.

3.2 Experiment setup

The whole experiment design is split into two parts: the evolution experiment and the generalisation experiment. The evolution experiment will design a controller offline. When the evolution is finished, the best evolved controller will be tested online, which will be the generalisation experiment.

Since CGPANN has never been applied into a robust fault tolerance scheme, the controllers can be firstly evolved by CGPANN in two simple scenarios: the fault-free one and the faulty one. And the aim is to investigate which one could achieve the robust fault tolerant control for the online test. In terms of the fault-free scenario, the controllers are evolved without any fault injected. And in terms of the faulty scenario, the controllers are evolved when the fault is injected at the beginning the task.

When the controllers are obtained, they will be tested in the fault-free and faulty scenarios respectively in order to see which one is capable to achieve the robust fault tolerant control. It is normal that the evolved controllers are suited to their own evolution scenarios. However it is worth investigating how they will be performing for the opposite scenario and that is the key for the robust fault tolerant control since it cannot guarantee whether the fault will be definitely occurred or not. So the hypothesis of the experiment is that CGPANN is capable to evolve controllers that are effective to achieve the robust fault tolerant control.

Additionally, the generalisation experiment will be conducted in two conditions: the unlimited time test and limited time test. Basically, the limited time test will be much harder than the unlimited time test for the evolved controllers to achieve the robust fault tolerant control. However it is still interesting to investigate how the evolved controllers will be performing for different time condition tests, especially when the time limit is not a strict restriction.

3.2.1 Robot platform and task

For both of evolution and generalisation experiments, a robot platform simulator ARGoS [145] is used throughout the whole work. ARGoS is a *multi-physics* robot simulator and it can simulate large-scale swarms of robots with some kinds of robots efficiently. In this sense, a foot-bot robot platform is selected to be used as the experimental platform. It has 24 light sensors which are used to detect a light source. In addition, it is also equipped with 24 proximity sensors which are used to observe the surrounding environment in order to avoid obstacles. The robot task selected is a phototaxis mission achieved by a single robot in 1200 ticks, which is 120 seconds. The beacon is placed in the centre of the arena and the robot is placed in 10 different random initial positions and orientations with a fixed distance 4.5 m to the beacon. The light sensors would be used for this task, but 24 sensors make it quite easy for the robot to achieve the phtotaxis. In this case, only 8 sensors are picked evenly distributed around the robot, which are number 1, 4, 7, 10, 13, 16, 19 and 22. A sensor distribution graph of the foot-bot is shown in Figure 3.1 where only those mentioned 8 sensors were selected to do phototaxis. In this case, the controller would be evolved and tested based on these selected 8 sensors to make the robot achieve the phototaxis task in terms of faultless and fault scenarios respectively.

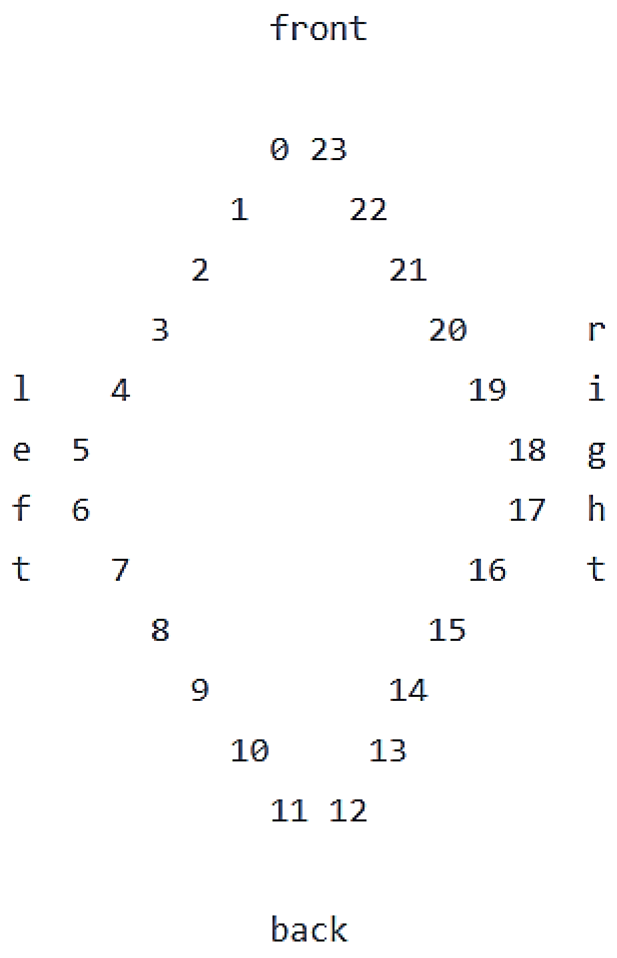

Figure 3.1: Light sensor distribution of foot-bot [145]

3.2.2 Fault type

In terms of the fault type that needs to be tolerated, the fault could be just a complete sensor failure which sets the faulty sensor signal reading to be 0 into the controller and the fault could be just injected from the beginning of the phototaxis task. To be specific, robot sensor 1 and 7 can be selected as the predicted sensor faults which will be utilised to evolve controllers in offline scenarios where both of these 2 sensor's readings are set 0 into the controller. It needs to note that actually any sensor could be failed when the robot is doing the task online. However, it is not an easy task to evolve a fault tolerant controller that is able to tolerate any kinds of fault especially when there is more than one fault occurred at the same time or sequentially when the robot is doing the task online. For this reason, evolving a controller based on predicted possible fault types could be an easier solution. Although this work needs a fault prediction technique from another research area, it can be assumed that a specific fault type has already been predicted in order to evolve a fault tolerant controller to deal with it. In this case, sensor 1 and 7 are selected as the predicted fault types in order to evolve a fault tolerant controller for it. Although this work lacks generalisation to tolerate unplanned fault, it makes the whole work concentrate on this single scenario based on the assumed predicted fault types. That is to say, as long as the fault prediction technique is effective enough, there is no need to evolve controllers to tolerate any fault types and evolving controllers based on the predicted fault types could be the most efficient way.

Moreover, it also needs to note that it is essential to choose 2 sensor faults to evolve controllers rather than just 1 sensor fault as the predicted fault type. Figure 3.2 shows an example of a best evolved controller’s internal structure with just sensor 1 failure. In this controller, there is no connection from the faulty sensor (input 0) to the controller, so it doesn’t matter whether the sensor is really failed or not when testing this controller. In this case, the robot will perform the phototaxis with the left 7 sensors and as long as an acceptable controller is evolved, it will definitely achieve the robot sensor fault tolerance for both of faultless and faulty test.

This idea is fine with controller connections just from the working sensors to design a fault tolerant control system, but it's not a sufficient scheme. Suppose there is more than 1 sensor failed during the task. If a controller was designed without any connection from these failed sensors, it could make robot be robust to the upcoming faults. But the performance in the faultless condition will be definitely degraded compared to the full sensor connection evolved controller especially in the multi-sensor failures situation. In addition, faults will not always be occurred at the beginning of task. So there is always a period that the robot performs the task in a faultless circumstance. In this case, a fully connected controller will definitely be the first choice with all of the sensors working around to achieve tasks.

On the other side, another possible reason to obtain a controller like this one in Figure 3.2 would be that 7 sensors may be already sufficient for robot to perform phototaxis due to the compensation of the neighbouring working sensors besides the failed one. So in order to prevent the sensor compensation effect, 2 sensor faults occurred together would be a feasible solution, which could also reduce the chance to obtain a controller just connected to the working sensors. And that's why sensor 1 and 7 failed together would be used as a primary scenario for this work.

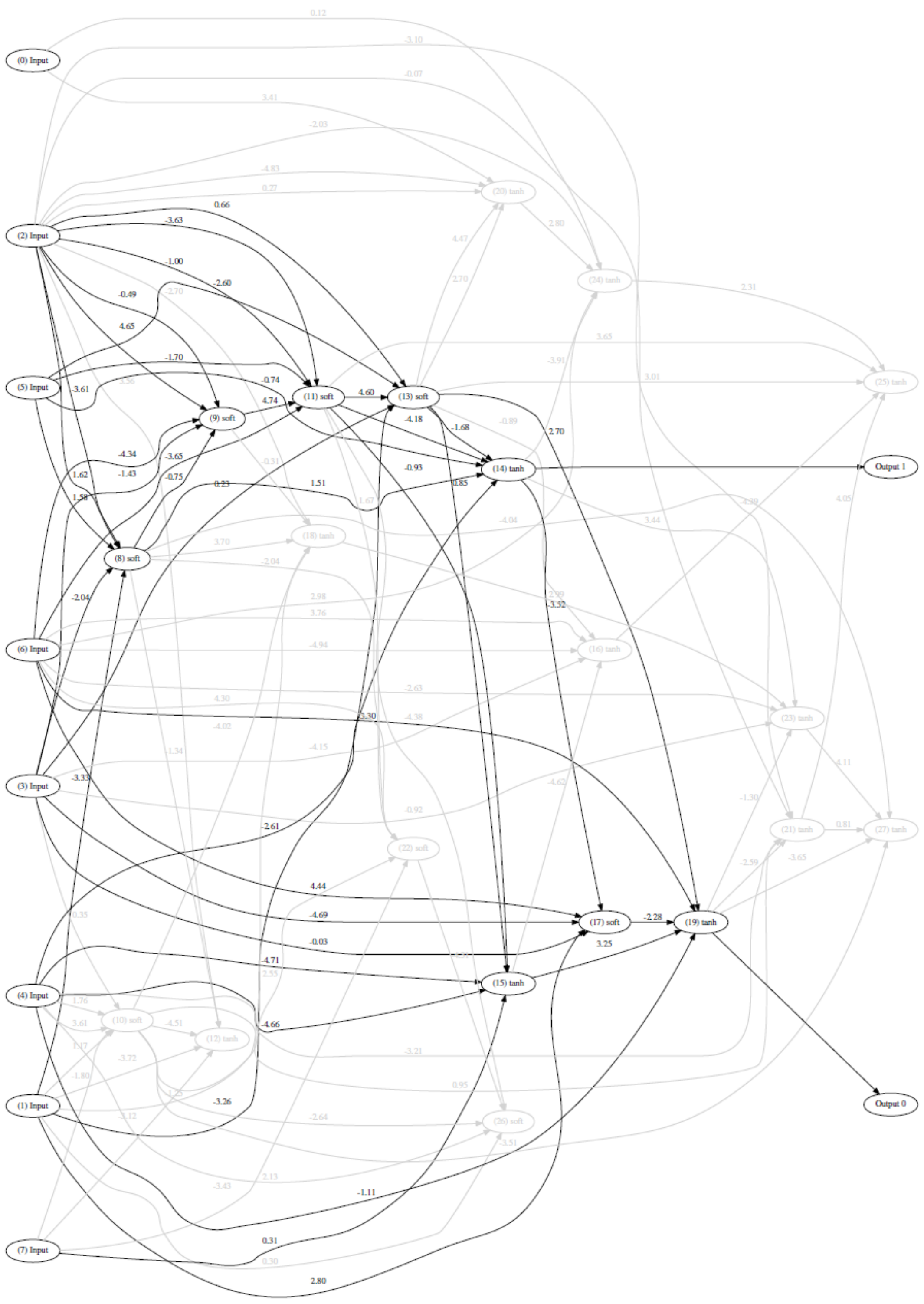
Figure 3.2: An example of CGPANN evolved controller without the connection to the failed sensor

3.2.3 Evolution experiment

In terms of the evolution experiment, each evaluation would choose the worst fitness value among 10 trials as the last fitness value. This kind of evaluation method could minimise the wrong behaviours of the robot in order to make sure that the evolved controller is able to make the robot achieve phototaxis for all of the 10 trials. In other words, if the robot could achieve the phototaxis in the worst case trial, the robot will definitely achieve the phototaxis in the other 9 trials with better performance. However there are some potential drawbacks for this kind of fitness function. One problem is that choosing the worst case fitness value among 10 trials may not fully demonstrate the controller performance since only the worst case is utilised as the final fitness and there is no information preserved for the other 9 trials during the evolution. The other problem is due to the similarity of the individuals if just the worst case fitness value stands for the individual's performance. For example, if two individuals have the same fitness value for the worst case performance but different fitness values for the other 9 trials, there is no way to further rank these two individuals in terms of their final fitness values. This problem will impact the performance of crowding fill during the survival selection in the multi-objective optimisation, which will be talked about in more details in section 4.5.1.2. Even though, choosing the worst case performance as the final fitness value of the current individual could still be a suitable choice for single objective optimisation like this evolution experiment since wrong behaviours could be minimised in this way and extremely poor performance could also be prevented during the evolution.

The fitness function would be made up of two parts: the constraint function and objective function. The constraint function evaluates the individuals to see whether they can make the robot reach the beacon in an area of 0.01 m as the radius within the maximum allowed time: 1200 ticks. So the worst fitness value would be the longest distance of the robot to the beacon after 1200 ticks. If the robot can reach the beacon in that area in 1200 ticks, the individual will be evaluated on the objective function which is the time spent of the robot to reach that area. When an individual can make robot reach the area in all of 10 trials, the worst fitness, which is the longest time spent, will be selected as the final fitness value of this individual. This constraint handling process is quite basic since it just evaluates the individuals for the constraint function first and then for the objective function, which is much simpler than the one developed in NSGA2 [92]. However this basic constraint handling process is already adequate for this single objective CGPANN experiment, so it could still guarantee that the best evolved controller would be able to perform the phototaxis well in all of these 10 trials with these 10 random initial positions and orientations of the robot.

In terms of CGPANN parameters, a (1+4) evolution strategy was used for the population size which is the same as CGP. The number of nodes was set 20, the number of arity was set 5, the weight range was set +/-5, and mutation rate was set 5% with a probabilistic mutation. The selected node functions were hyperbolic tangent and soft sign neuron transfer functions. Both of these 2 functions generate output in the range of [-1, 1], which is suited to robot wheel speed. The robot wheel speed was set 5 times larger of the controller output, which is [-5, 5]. In this case, each of the wheels can move forward or backward with a maximum speed of 5 m/s. So the shortest time of the robot to complete phototaxis task is when the robot moves straightforward to the beacon with the maximum speed, which is 900 ticks. In terms of convergence criteria, 50 generations were set to observe the convergence. So if the fitness value hadn't been changed for 50 generations, the evolution could be terminated.

3.2.4 Generalisation experiment

To check the capability of the evolved controllers for the robust fault tolerant control, 30 best evolved controllers obtained offline from 30 independent different evolution experiments for faultless and faulty scenarios respectively would be tested online for each of these two scenarios respectively. After the generalisation experiments were finished, success rate would be used for the assessment of these evolved 30 controllers and a comparison between them was conducted to see how the evolved controller would be performing for the robust fault tolerant control.

Apart from the fault scenario test, the best evolved controller would also be tested in 10 new different random initial positions and orientations of the robot with the same distance 4.5 m to the beacon, and these 10 new positions and orientations are different from that in the evolution experiment. The motivation of the test is to investigate whether the evolved controller can make the robot do a real phototaxis task no matter what the robot initial position and orientation are. In this case, each controller would be tested to make the robot start with 10 new random different initial positions and orientations and success times among 10 trials would be the final success rate of this controller.

In terms of the online generalisation experiment length, 1200 ticks were utilised as a first choice since the offline evolution experiment utilised the same experiment length for the controller evolution. However due to the experiment difference between offline evolution and online test, 1200 ticks may be too difficult for the robot to complete the phototaix in the online testing scenario. For this reason, 3000 ticks were utilised as another option to test the controller performance in order to find out whether the robot could complete the phototaxis task if more experiment time is given for this online testing scenario. As a result, if the robot cannot complete the phototaxis in 1200 ticks or 3000 ticks, the success rate would be set 0 for this current trial.

3.3 Result and discussion

3.3.1 Faultless scenario evolved controller

3.3.1.1 3000 tick test

Table 3.1 is the test result of success rate comparison from faultless evolved controller. Firstly 30 best evolved controllers were tested for faultless condition and then tested for faulty condition. All of these 30 controllers could make robot achieve phototaxis in 1200 ticks from evolution results. So the generalisation experiment would check whether these 30 controllers could still make robot achieve phototaxis with 10 new different random robot initial positions and orientations for both of faultless and faulty conditions.

Table 3.1: Success rate comparison of faultless evolved controller in 3000 ticks

	Success rate in 3000 ticks														
Faultless test	1	1	1	1	1	1	1	1	1	1	0.8	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Faulty test	1	0	1	1	1	1	0	0.9	0	1	0.9	1	0	1	1
	1	0.3	1	1	1	1	0.8	1	1	0	1	1	1	0	0

As can be seen from Table 3.1, the success rate of faultless test among 30 controllers is much higher than that of faulty test. To demonstrate the data distribution, a boxplots was used for it, which is shown in Figure 3.3.

As presented in Figure 3.3, the data distribution of faulty test has a much lower success rate area than that of faultless test. Based on the Mann-Whitney U-Test calculation, p value is 0.02444. So the difference between the faultless and faulty tests is significant. Based on the utilising of Vargha-Delaney A-Test, the score is 0.67 which is located in the medium effect size interval. In this sense, there is a medium effect between the faultless test and faulty test based on the faultless scenario evolved controller in terms of 3000 tick test.

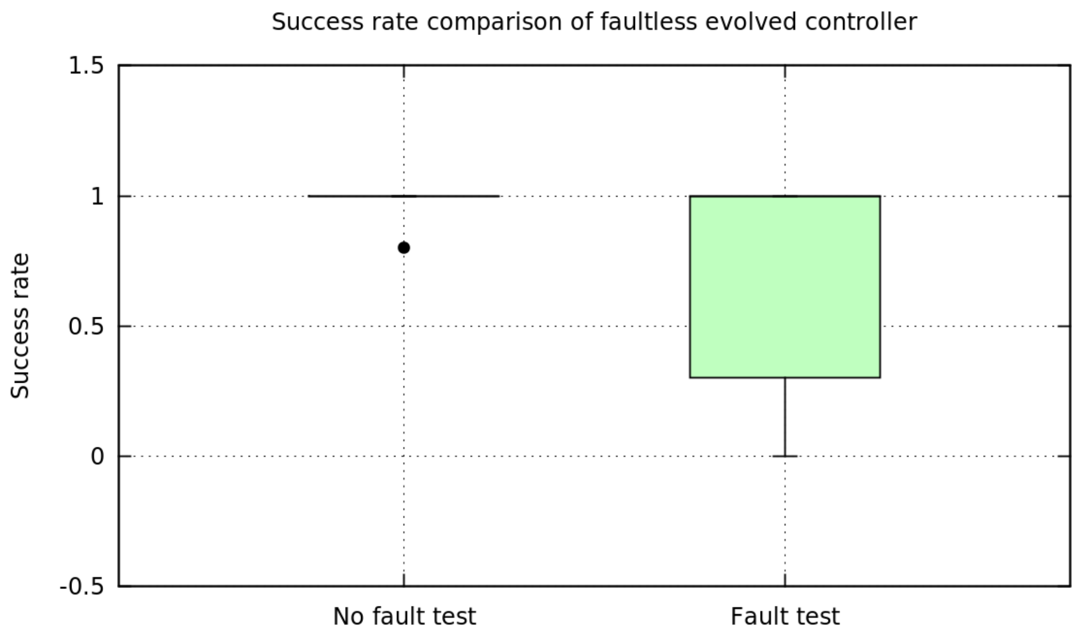

Figure 3.3: Boxplot for success rate comparison of faultless evolved controller in 3000 ticks

3.3.1.2 1200 tick test

As is mentioned in the section 3.2.4, 1200 tick test was also conducted as a comparison of the same 30 controllers. The success rate is listed in Table 3.2 and the boxplot is displayed in Figure 3.4.

Table 3.2: Success rate comparison of faultless evolved controller in 1200 ticks

	Success rate in 1200 ticks														
Faultless test	1	1	1	1	1	1	1	1	1	1	0.8	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Faulty test	0.8	0	0	0	0	1	0	0.9	0	1	0.5	1	0	1	0
	1	0.3	0	0	0.2	1	0.8	1	1	0	0	0	1	0	0

From Table 3.1 and Figure 3.4, the success rate of faulty test now is much lower than the faultless test among 30 controllers with a declined median value compared to the 3000 tick test. The p value calculated is less than 0.00001 from Mann-Whitney U-Test and Vargha-Delaney A-Test score calculated is 0.8427778 which is above 0.71.

Therefore there is a large difference between the faultless and faulty tests for the faultless scenario evolved controller in terms of 1200 tick test.

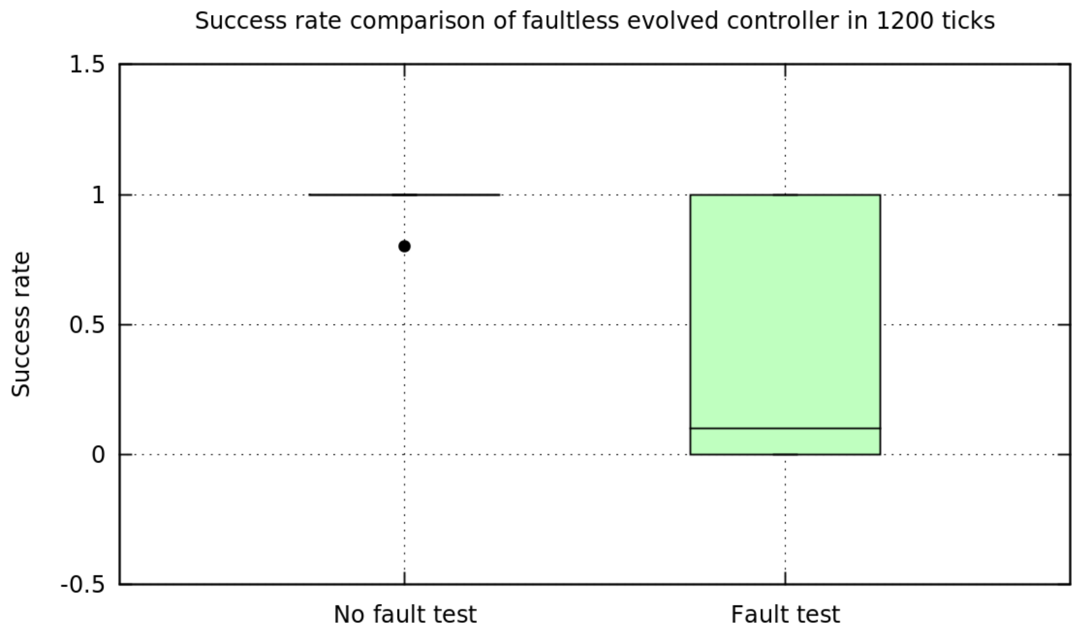

Figure 3.4: Boxplot for success rate comparison of faultless evolved controller in 1200 ticks

3.3.1.3 Conclusion

Based on the comparison results of faultless scenario evolved controller, the faultless tests outperform faulty tests with significant difference for both of 3000 tick and 1200 tick tests. Although there is a medium difference for 3000 tick test, there is a large difference for 1200 tick test. And that means if the time is limited for the robot to perform phototaxis, the robot will have a worse performance in the face of sensor faults.

In conclusion, the faultless evolved controller cannot be robust to the robot sensor faults for phototaxis task especially when there is a strict time limit such as 1200 ticks. In this sense, a fault tolerant control system is really necessary to make robot continue doing phototaxis task in the face of sensor faults. The next section 3.3.2 will consider using CGPANN to evolve a controller with sensor faults injected at the beginning of the phototaxis and investigae whether the faulty scenario evolved controller can achieve the robust fault tolerance control.

3.3.2 Faulty scenario evolved controller

3.3.2.1 3000 tick test

Table 3.3 presents the test result of success rate comparison from 30 best faulty evolved controllers and Figure 3.5 shows the data distribution of them. The p value calculated is 0.00194 from Mann-Whitney U-Test and the Vargha-Delaney A-Test score is 0.7333333.

As can be seen from this result, the faulty condition evolved controllers make the robot work well for faulty conditions but not for faultless condition. The boxplot also demonstrates a large difference between their data distributions. According to Mann-Whitney U-Test, there is a significant difference between these 2 data sets and Vargha-Delaney A-Test also indicates a large difference between them.

Table 3.3: Success rate comparison of faulty evolved controller in 3000 ticks

	Success rate in 3000 ticks														
Faulty test	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Faultless test	0.6	0	1	1	1	0.6	0	0	1	1	1	0	1	0	1
	1	1	0	0.2	1	0	1	0	1	0.2	0	0.5	1	1	1

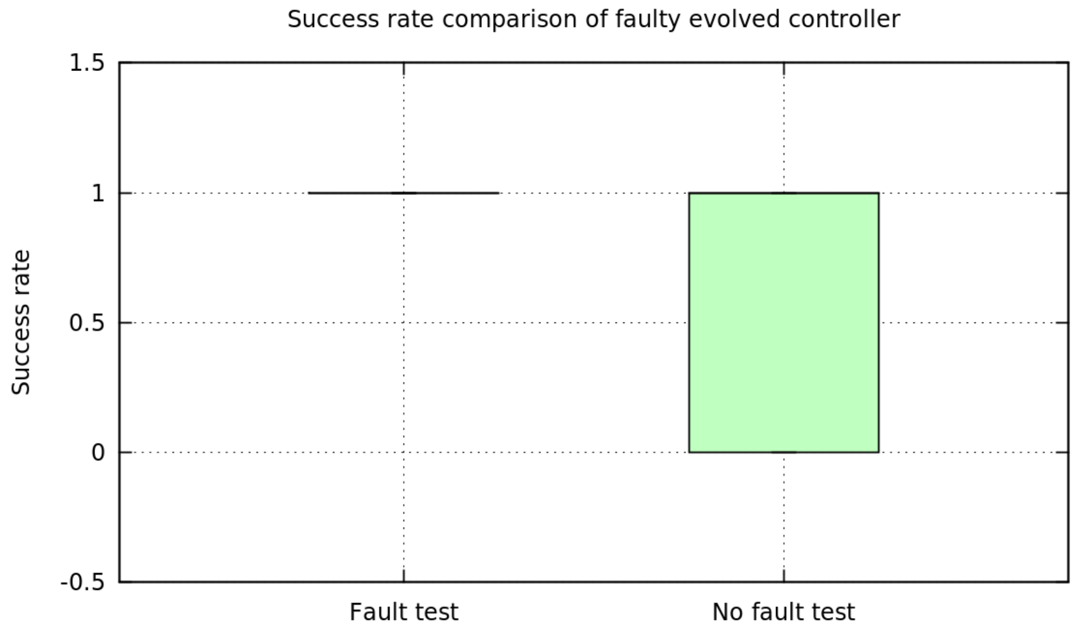

Figure 3.5: Boxplot for success rate comparison of faulty evolved controller in 3000 ticks

3.3.2.2 1200 tick test

In terms of 1200 tick test, Table 3.4 shows the success rate comparison result of 2 data sets and Figure 3.6 displays the distribution of them. Mann-Whitney U-Test gives a p value of less than 0.00001 and the Vargha-Delaney A-Test score is 0.8166667.

As is shown from these results, the faulty test still outperforms than faultless test. The best evolved controllers from faultless test also have a much lower performance with a lower median value compared to the 3000 tick test. Mann-Whitney U-Test implies a significant difference between these 2 data sets and Vargha-Delaney A-Test indicates a large difference between them, which is even larger than the A-test score of the 3000 tick test.

Table 3.4: Success rate comparison of faulty evolved controller in 1200 ticks

	Success rate in 1200 ticks														
Faulty test	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Faultless test	0	0	1	1	1	0.6	0	0	1	1	0.9	0	1	0	0
	0	1	0	0.2	1	0	1	0	0.5	0	0	0.5	1	1	0

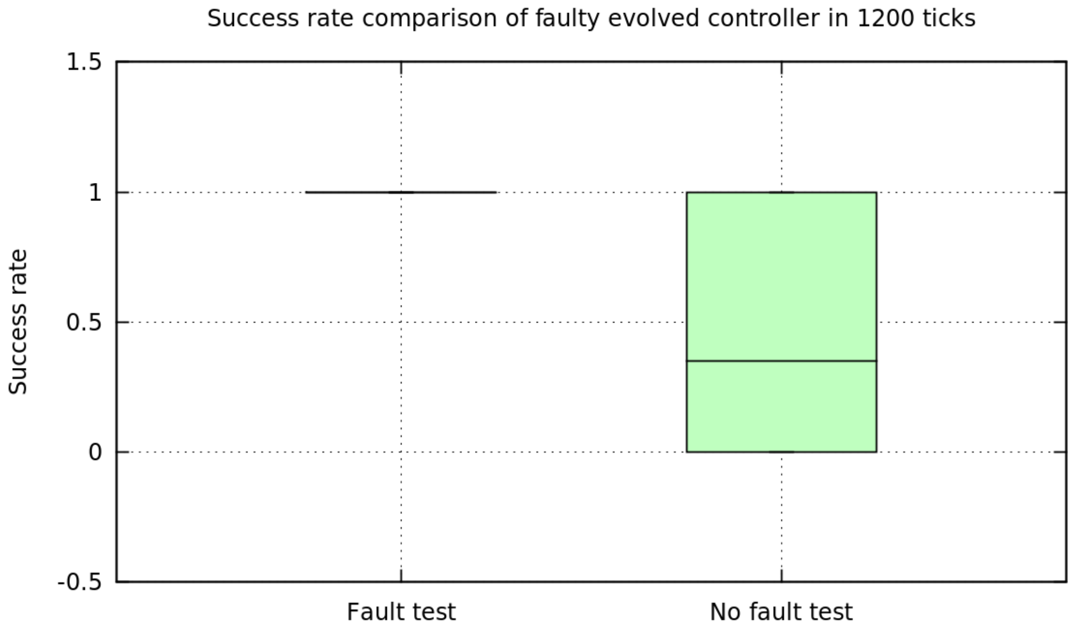

Figure 3.6: Boxplot for success rate comparison of faulty evolved controller in 1200 ticks

3.3.2.3 Conclusion

As can be seen from these two generalisation experiments, the results are quite similar to the previous experiment of faultless scenario evolved controller test. The faulty condition evolved controllers still work well for faulty condition but not for faultless condition especially when the experiment time is limited.

As a result, one scenario designed offline controller seems not sufficient to make the robot perform phototaxie online in both of faultless and faulty conditions so as to achieve robust fault tolerant control. In this sense, one option is to design two or more controllers offline to deal with different objectives. As is reviewed in the fault accommodation area in section 2.2.2, a bank of controllers can be pre-designed offline with CGPANN to achieve the switched fault tolerant control. And when the robot is doing the phototaxis task online, the bank of controllers can be switched to each other whenever there are faults or not. This idea is suitable for the real time fault tolerance situations where the controllers can be just switched online based on the assumption that all the predicted possible faulty conditions have been considered to design offline bank controllers and the fault can be diagnosed immediately right after its occurrence by an effective online fault diagnosis mechanism.

Another option is still designing one controller to be robust to both of faultless and faulty conditions. One solution is to implement a weighted sum based CGPANN where the controllers can be evaluated based on the weighted sum for faultless and faulty objectives. The other solution is to utilise a multi-objective CGPANN in order to obtain a trade-off controller, which could produce an equivalent performance for each of faultless and faulty objectives. One significant advantage of a multi-objective optimization algorithm is that it can obtain a set of controllers for all of the objectives respectively, which is called Pareto-optimal solutions [92]. From the Pareto set, any controller can be selected depending on what the objective is required.

As can be seen from section 2.5.5, multi-objective optimisation has a significant benefit over weighted sum based single objective optimisation for multiple objective problems, which is the larger behaviour diversity. According to [119], weighted sum based single objective optimisation could produce the most trade-off solution as multi-objective optimisation where the evolved solution is able to achieve an equivalent performance for both objectives at the same time. However it is quite hard for weighted sum based single objective optimisation to obtain other trade-off solutions that multi-objective optimisation is able to obtain just from the Pareto optimal front due to its larger behaviour diversity. And this is a serious drawback for weighted sum based single objective optimisation.

In terms of the fault tolerant control in this work, weighted sum based single objective optimisation at least needs three evolution loops in order to obtain a trade-off solution to achieve the robust control or two solutions performing well on each objective in order to achieve the switched control. This scheme may achieve the fault tolerant control for this work but it needs multiple evolution loops to obtain the desired solutions especially when there are more than two objectives, where multi-objective optimisation just needs one evolution loop to obtain a set of solutions based on the Pareto optimal front no matter how many objectives to deal with. And this would be another significant advantage of multi-objective optimisation over weighted sum based single objective optimisation.

In conclusion, a multi-objective CGPANN could be an alternative promising approach utilised for designing fault tolerant controllers rather than just single objective CGPANN since it is able to evolve a Pareto optimal set of solutions not only for robust but also for switched fault tolerant control. Although the performance of the robust control with this trade-off controller will be degraded compared to switched control with bank of controllers, robust control could still be a promising scheme since this strategy doesn’t need a pre-designed controller switch procedure along with a fault diagnosis mechanism and just one controller also saves the memory space when the robot is controlled by an embedded single chip microcomputer for the real world experiment. Nevertheless, robust or switched fault tolerant control could both be worth an investigation by multi-objective CGPANN.

3.4 Summary

This chapter shows how CGPANN is utilised for designing a robust fault tolerant control system for the robot phototaxis task in the face of sensor failures. In terms of the fault tolerant control scheme, robust fault tolerant control could be a first choice since just one controller is required to be evolved without any consideration of controller switch and fault diagnosis. However, as can be seen from the result in section 3.3, CGPANN is failed to obtain controllers that are able to achieve the robust fault tolerant control since only one objective optimisation is not sufficient for CGPANN to obtain a robust controller. For this reason, a multi-objective CGPANN could be an alternative solution which will be able to obtain a Pareto set of controllers working not only for the robust fault tolerant control but also for the switched fault tolerant control depending on which controller is selected for the task requirement.

The next chapter will discuss how to develop a multi-objective CGPANN algorithm and investigate how it will be working for both of robust and switched fault tolerant control based on the robot phototaxis task.

Chapter 4 MOCGPANN in fault tolerant control

4.1 Introduction

As is concluded in chapter 3, CGPANN is not able to evolve controllers that are capable to achieve the robust fault tolerant control scheme in terms of the robot phototaxis task. In this sense, a multi-objective CGPANN could be an alternative choice. One significant benefit of MOCGP is that it could evolve a set of controllers which are capable to make the robot achieve different objectives as mentioned in section 3.3.2.3. In this sense, the MOCGP will be a promising solution to evolve not only a robust controller but also a bank of controllers with respect to different objectives so as to achieve the robust as well as the switched fault tolerant control scheme.

CGP has been successfully implemented in many areas, including the digital circuit design, the image processing and many medical applications [29]. However in the case of multi-objective optimisation, CGP has not been fully explored as well as CGPANN. Although there have been developments of various types of multi-objective CGP (MOCGP) [104] [105], there is no formally published MOCGP library. Therefore the integration of a MOCGP as well as a MOCGPANN library would be essential and interesting to be utilised for the robust fault tolerant control where CGPANN was failed to achieve. Furthermore, there will be two main problems when developing MOCGP, which are the problem of crowding fill strategy during survival selection and the problem of assessing the convergence of MOCGP. And these two problems need to be investigated before developing the library.

4.2 Research gap in MOCGP

4.2.1 MOCGP development

This MOCGP is a combination of CGP and NSGA2. In MOCGP, the whole evolution loop is based on CGP except for the evaluation stage which is created from NSGA2. MOCGP still implements a kind of (1+4) evolutionary strategy with a random parent selection strategy from CGP [29] to create each population. The mutation is also the same as CGP, however the survival selection stage is quite different. Since it evaluates the population in multiple objectives, the survival selection is mainly borrowed from NSGA2 and the crowding measure is also implemented to encourage population diversity in the face of different objectives during crowding fill [92].

- NSGA2:

According to section 2.5.1.3, NSGA2 performs better than PAES in terms of the diversity preserving mechanism. However, PAES outperforms NSGA2 in terms of the convergence except for one case when the problem has strong parameter interactions. Generally speaking, each of these two algorithms has its own benefit, but the advantage of PAES is only suited to the problem whose true Pareto optimal front is known. If the true front cannot be obtained before the experiment is run, then this advantage of PAES will be weak. On the other hand, NSGA2 integrates an effective constraint optimisation mechanism which is suited for a wide range of constraint handling problem. For these reasons, NSGA2 could be a suitable choice along with its crowding measure for the survival selection in MOCGP development.

- CGPANN:

According to section 2.4.5, CGPANN outperforms NEAT and SANE in terms of the double pole balancing control problem. The result demonstrates that CGPANN needs much less evaluation numbers than NEAT as well as SANE, which indicates that CGPANN realises a better convergence. Although the transfer function utilised in NEAT is slightly modified compared to the one used in CGPANN, whether the difference is significant for performance comparison is unknown at the moment. Generally speaking, CGPANN could be a first choice rather than NEAT to be utilised for designing NE based structurally evolvable controllers for multi-objective optimisation due to its effective performance. Moreover, another significant advantage of CGPANN is the implement of explicit genetic redundancy which could also be utilised to improve the crowding fill performance, where NEAT is not able to achieve due to the lack of genetic redundancy. The details of why genetic redundancy in CGPANN could improve the crowding fill will be talked about in the next section 4.2.2.

4.2.2 Crowding fill problem

However there are still some problems inside MOCGP. In multi-objective optimisation, the population diversity of the final Pareto-optimal front and the convergence to it are two main issues, which are still in the research [146] [124]. During the survival selection of NSGA2, the population diversity can be guaranteed based on the computation of crowding distance for each individual. In this case, the individuals in the same set will be ranked based on its crowding distance from highest to lowest in order to be survived into the next generation, which is also called the crowding fill [92]. However the most significant difference between CGP and GA is the representation of individual’s genomes. In GA, all the genes are active for the mapping from its genotype to phenotype. So encouraging the population diversity in the fitness domain is necessary and enough in a parameter based multi-objective optimisation algorithm. And that’s why NSGA2 is famous for its crowding distance technique. This rule also applies to any EA approaches that do not have genetic redundancy in the genome encoding, such as NEAT/HyperNEAT in the ANN structure optimisation domain in terms of multi-objective optimisation. However in CGP, there is a large part of genes which are inactive to the mapping from genotype to phenotype for each individual. Since the inactive genes have no contribution to the mapping from genotype to phenotype, those inactive genes could be quite different among the individuals. Once some of the inactive genes become active during mutation, it is possible that the fitness could have a great change. For this reason, this kind of explicit genetic redundancy is quite useful for CGP based approaches [29] [30]. According to section 2.4.4.5, one significant implement of explicit genetic redundancy is preserving the genetic drift, which is beneficial to not only the evolutionary search but also the escaping from local optima. In this case, it is necessary to distinguish the individuals with the same fitness at least between children and parents in order to preserve the neutral mutated individuals during the crowding fill in the survival selection. However, in terms of those algorithms without genetic redundancy such as GA or NEAT/HyperNEAT, there is no need to distinguish individuals with the same fitness since preserving genetic drift has no effect on them.

On the other hand, the distinction between individuals may be helpful to improve the crowding fill performance as well. As is mentioned in section 2.5.1.1, the crowding distance measure referred from NSGA2 is actually an estimate of the density of the current individual based on its two neighbours around it in the current population. The computation of crowding distance is the value of the cuboid perimeter of this current individual enclosed by its nearest two neighbours as the vertices from each side [92]. However the crowding distance measure may not be working well for those algorithms with the genetic redundancy such as CGP since it cannot tell the difference of individuals with same fitness. In this sense, crowding distance has to pick two random individuals as two neighbours to compute the cuboid perimeter of the current individual. For this reason, some individuals may have the neighbours with the same fitness while some may have the neighbours with different fitness. Therefore it may be unfair to set different crowding distance values for the individuals who have the same fitness based on a random neighbour pick strategy during the crowding fill, which is not beneficial for survival selection in terms of genetic redundancy based genome encoding approaches like CGP. Actually, the random picking problem in crowding measure is not serious for the algorithms which have no genetic redundancy such as GA or NEAT since the individuals with the same fitness will always have the same or quite similar genotypes. In this case, even if the individuals with the same fitness have different crowding distance values, there will be no significant influence on the crowding fill performance. In a word, distinguishing or re-ranking individuals with the same fitness during crowding fill is quite necessary not only to preserve the genetic drift but also to improve the performance of crowding fill to fit MOCGP/MOCGPANN.

In conclusion, due to the explicit genetic redundancy in CGP, the individuals with the same fitness may have huge difference in their inactive genes and the original crowding measure referred from NSGA2 is not working to distinguish them. For this reason, distinguishing individuals at least between children and parents during crowding fill is essential for MOCGP/MOCGPANN development. On the other hand, the distinction between individuals with the same fitness may also be helpful to improve the crowding fill performance so as to avoid the random neighbour pick problem caused by the crowding measure. Although [30] mentions that CGPANN does not benefit from preserving genetic drift rather than CGP, it is still worth a further investigation for MOCGPANN based on an improved crowding fill strategy.

4.2.3 Convergence problem

Another problem is how to set the convergence criteria in MOCGP. In single objective optimisation such as GA, the convergence of population is not difficult to measure. Convergence criteria is actually used to decide when to stop the evolution [120]. A well-known approach about how to stop GA was developed by [121] where GA can be stopped if the best performance values have stabilized. And this is actually the description of convergence measurement [120]. However in multi-objective optimization, there is no straight forward method to measure the convergence since there is no best evolved individual for each population. In [92], the convergence criteria is based on the observation whether the current pareto front is close enough to the pareto optimal front. If it reaches the pareto optimal front, that means NSGA2 has converged. However there is a premise that the pareto optimal front should be known before running the evolution. If the pareto optimal front cannot be obtained or calculated beforehand, then the convergence cannot be measured. In this sense, the convergence criteria should also be considered for MOCGP if pareto optimal front cannot be obtained in advance. So how to find a metric to measure convergence is also an relavant and essential stage for MOCGP development.

4.3 Methodology

4.3.1 Methodology for new crowding fill

As can be seen from section 4.2.2, an improved crowding fill strategy needs to be developed in order to fit MOCGP/MOCGPANN. Firstly, a distinction between children and parents has to be carried out during the crowding fill in order to preserve the neutral mutated individuals into the next generation if there are more than one individual have the same fitness. This idea is feasible to preserve the genetic drift like how CGP does. However the children actually still have quite different inactive genes between each other although they have the same fitness. Due to the random neighbour pick problem in crowding fill, it may be worth a further distinction between these children to avoid that problem. In this sense, if only one child who has the largest genotype diversity among the children with the same fitness is allowed to take part in the crowding measure, the random neighbour pick problem may be solved since there is no need to consider how to choose the neighbours for those individuals with the same fitness.

4.3.1.1 Population diversity measures

- Genotype diversity measure

According to section 2.5.4.1, genetic distance is an effective and a promising metric to assess the genotype diversity for the population based on the implement of Euclidean distance or Hamming distance. Although it is a commonly used approach, there may be some problems to fully represent the genotype diversity especially for MOCGP. The problem is caused from the MOCGP real-coded genotypes where some genes stand for the transfer function of the node, some for the connection of the node and some for the weight of the current connection. In addition, there is another type of genes representing which nodes the outputs connect [29]. This kind of genes always belongs to the active genes so they are not related to the neutral genetic drift. Apart from the output genes, those three kinds of genes could be utilised into the computation of Euclidean distance. The weight genes can be used to calculate Euclidean distance since larger distance means larger changes of the corresponding weight values. However the changes of connection and function genes may not be suitably utilised for Euclidean distance computation. The connection genes are integer numbers of nodes where the current node is connecting and the function genes are also integer numbers demonstrating which function the current node is using. In this sense, Euclidean distance may not be effective to display the real genetic distance between those genes. The reason is that larger distance based on Euclidean distance with those integer numbers may not mean larger distance between the corresponding genes. For example, a child mutates one of its node's connection from number 1 (the parent) to 2 and another child mutates its corresponding node's connection from number 1 (the same parent) to 9. Obviously, the second child has a larger Euclidean distance of this gene from the parent but actually that may not mean the second child has a larger genetic distance of this gene. This result just means these two children have different connection of this node but it cannot conclude that the second child has a larger genetic distance just because the difference between 1 and 9 is larger than 1 and 2. Similarly, it is the problem occurred in function genes.

In this case, Hamming distance could be an alternative choice to deal with this problem. One significant usage of Hamming distance is to measure the genetic distance in DNA sequences [147] [148]. Although the genetic distance mentioned there is for the real genes of alphabetical strings in biology area, it provides a solution to measure the genetic distance for the genotype diversity problem in MOCGP. Since the Hamming distance can be used to measure the difference between two alphabetical strings from DNA sequence, it will possibly be used to reflect the difference of inactive genes between two individuals in MOCGP. Based on Hamming distance as a metric to measure genetic distance, the possible approach to measure the genotype diversity is just to check how many genes have been mutated in the corresponding nodes among individuals. This approach is simpler for computation than Euclidean distance for genetic distance, but it may be more effective to reflect the real genetic distance between two individuals.

- Phenotype diversity measure

Apart from the neutral mutation, there is a special situation occurred during the evolution in MOCGP where the same fitness individuals are also created resulting in the random neighbour pick problem during crowding fill. This special situation is from the normal mutation and it only happens when different fitness parents create the same fitness children between each other. This kind of special situation based on normal mutation may not be occurred frequently as that one based on the neutral mutation, but it does take place by chance as long as some parents create the same fitness children who have the same fitness with those created by neutral mutation. In this special situation, the population diversity cannot be improved by utilising genotype diversity measurement since the genetic distance only demonstrates the genotype diversity among the children who are created by neutral mutation from their parents.

In this case, phenotype diversity measurement could be an alternative solution to deal with this special situation. According to section 2.5.4.2, Fitness distance can be directly used for the individuals who have the same fitness but created from the normal mutation rather than neutral mutation. The computation of fitness distance of an individual could be just the difference between its fitness and its parent’s fitness. And Euclidean distance can be a primary metric to calculate the fitness distance since the fitness is presented in real number. In this way, the phenotype diversity could be possibly maximised during survival selection for individuals created from normal mutation but with identical fitness.

- Procedure of a complete population diversity measure

As mentioned before, genotype diversity measurement uses Hamming distance as the metric of genetic distance to compute the difference between the current individual created by neutral mutation and its parent, who are the same in fitness. Hamming distance counts the number of different genes between these two individuals for inactive genes. In this way, large Hamming distance means this individual has a large difference of its inactive genes compared to its parent and vice versa. So the genotype diversity can be maximised based on the ranking of the Hamming distance of those individuals.

On the other hand, phenotype diversity measurement uses Euclidean distance as a fitness distance metric for the computation of the difference between the individual and its parent. This situation only applies to the individuals who are created by normal mutation but still have the same fitness with others in the current generation. In terms of the computation, it is not just the Euclidean distance between the fitness of two individuals since each one has at least two objective values. For this reason, the dominance will firstly be checked. If the individual dominates its parent, the Euclidean distance between them will be the real Euclidean distance. If they are non-dominated, which means they are the same in fitness, the Euclidean distance will be zero. And finally if the individual is dominated by its parent, the Euclidean distance will be the negative value of real Euclidean distance. The reason to set a negative value is that since the individual is dominated by its parent, it means the individual is worse than its parent in fitness. So in this way, all the individuals can be ranked from large to small in fitness distance, which means the phenotype diversity can be maximally preserved.

In addition, parents with the same fitness can be also available for the computation of the genotype and phenotype diversity measurements. Although parents have a zero distance between itself in spite of genotype or phenotype, it still retains the diversity information from last generation. So as long as the diversity information from previous generations are still kept, the parents with the same fitness in the current generation can be also compared and ranked depending on its preserved genotype or phenotype distance. However the ranking of parents are executed after the ranking of children to maximise the population diversity during crowding fill.

It needs to note that the individuals from normal mutation may need to be ranked before those from neutral mutation. The possible reason is that the individuals created from normal mutation may have more chance to still produce children with normal mutation. So the fitness could have more chance to be changed no matter improve or decline, which is beneficial to the evolutionary search. However those created from neutral mutation may have less chance to produce children in normal mutation, which means the fitness will possibly be the same into next generation and that is not helpful for evolutionary search. However this is just an intuitive strategy, so whether it is beneficial for the survival selection to guide the evolutionary search will still need to be investigated by the experiment.

In conclusion, when the individuals have the same fitness in the current generation during survival selection, the children with the same fitness from normal mutation will be ranked with Euclidean distance in fitness domain and survived first. And the children from neutral mutation will be ranked with Hamming distance in genotype domain and survived following. Finally the parents with the same fitness will be ranked and survived depending on normal or neutral mutation from previous generations.

4.3.1.2 New crowding fill in the survival selection

Based on the genotype and phenotype diversity measurements, this new crowding fill strategy involves some improvements in crowding distance sorting during the survival selection. Before the children and parents are merged for the non-dominated sorting, every individual needs to be checked if it has the same fitness with each other in the current generation. If so, each child will be attached an extra attribute of fitness distance or genetic distance from its parent by means of Euclidean distance or Hamming distance and the parents will keep their distance information from previous generations.

During the new crowding fill strategy, each individual in the current front will be compared to each other and separated into different groups based on their fitness. In this sense, the individual with the same fitness will be classified into the corresponding group depending on what the objective value is. Then in each group, the individual who ranks the first in the population diversity measurement will be removed from this group in order to take part into the crowding distance sorting while others still remain unchanged in their own group. After that, all the left individuals in their groups will be merged together into a new bigger group. Now the crowding distance sorting will work on the individuals in the current front including the first ranking individual of population diversity picked from the corresponding group. When the crowding distance sorting is finished, the left group members will be sorted based on their population distance values and filled one by one until the next generation is full. This whole process will be the modified crowding fill strategy for MOCGP instead of the original one developed for NSGA2.

This new crowding fill strategy not only solves the random neighbour pick problem in crowding distance computation, but also ensures the population diversity maximisation in survival selection. The individual ranking the first in population diversity measurement is survived before the others with the same fitness, which guarantees that only the individual with the largest distance value will be allowed to take part in the crowding distance sorting. And the left group members will be survived later on depending on their distance values, which also maintain the population diversity for the new generation. In this sense, this new crowding fill strategy designed especially for MOCGP will be utilised as the main approach for the evolution experiment throughout the thesis.

4.3.2 Methodology for convergence assessment

Based on the review of termination conditions in section 2.6, the upper limit of the number of generations or fitness evaluations is not a suitable choice since the limit cannot be obtained before the experiment starts to run. This termination condition only works if the experiment has run multiple times, so the estimated upper limit could be acquired based on them. The population diversity could be a choice, but it will be working better for single objective optimization problem. The multi-objective optimizer has already maintained the population diversity for each generation from the Pareto-optimal front based on the optimization of different conflict objectives, so there is little chance that the individuals in the whole population will converge to a certain solution. For this reason, the threshold could be the only way used as a convergence criterion when there is no significant improvement for fitness during a number of successive generations.

In terms of performance measure, it will be adequate to use hyper-volume indicator to assess the performance of MOCGP. The threshold will demonstrate the hyper-volume improvement among those continuous generations. Therefore, if the hyper-volume improvement approaches this threshold, then the evolution will supposed to be converged and then stopped. The reference point will be selected based on the supposed maximum objective value depending on what the fitness function is for this experiment.

4.4 Experiment setup

The whole experiment will be designed to investigate how MOCGP will be performing to evolve feasible Pareto sets of controllers so as to achieve the robust as well as switched fault tolerant control and how hyper-volume indicator will be working to assess the convergence of MOCGP. So the hypothesis of the experiment is that MOCGP and hyper-volume indicator can be integrated together in order to obtain effective Pareto sets of controllers so as to achieve both of robust and switched fault tolerant control in generalised scenarios.

The whole experiment of MOCGP was designed in a similar way as the one for CGP in section 3.2 where evolution experiment and generalisation experiment would be both conducted. The evolution experiment was designed to obtain feasible Pareto sets of controllers by MOCGP and the generalisation experiment was designed to test the evolved controllers for the robust fault tolerant control where CGP was failed to achieve and the switched fault tolerant control as well. Due to the task of evolving a neural network controller, MOCGPANN would be implemented rather than a general MOCGP with an additional weight gene and the node functions would be neuron transfer functions instead of simple mathematics functions.

Except for the optimisation algorithms used for evolution experiment, all the other factors of the experiment framework were totally the same as that in section 3.2 including the same robot task and the same fault type. The robot task was still a phototaxis task designed in section 3.2.1 and the fault type was also the same mentioned in section 3.2.2 where two complete sensor failures would be occurred together.

4.4.1 Evolution experiment

The aim of the evolution experiment is to investigate whether MOCGPANN could evolve feasible Pareto sets of controllers so as to achieve both of robust and switched fault tolerant control.

4.4.1.1 Baseline parameters

The individuals created by MOCGPANN in each generation would be evaluated in 2 constraint functions firstly and then 2 objectives functions, which implemented the same constraint evaluation process of NSGA2 [92]. The constraint functions still utilised the area of 0.01 m as the radius and the objective functions were still the time spent when the individuals met the constraint condition, which were the same as that designed in section 3.2.3. Since the constraint handling process is referred from NSGA2, so it would be more effective than the basic one utilised for CGPANN experiment in chapter 3. Moreover, due to the multi-objective optimisation features, the individuals would also be evaluated for the faultless and faulty objective respectively. The faultless objective refers to the normal condition where there is no fault for the robot to perform the phototaxis. And the faulty objective is the same condition designed in section 3.2.2 where sensor 1 and 7 will be completely failed with 0 sensor reading signal as the controller input. In order to obtain sufficient individuals to represent the Pareto optimal front, (1+4) ES was still utilised but 20 times larger which would be (20+80) ES. In this case, 20 individuals would be survived from the combination of 80 children and 20 parents in each generation, which could be used to create the Pareto front.

- MOCGPANN parameters

In terms of the MOCGPANN baseline parameters, they were the same as the ones in CGPANN experiment designed in section 3.2.3. The number of nodes was 20, the arity was 5, the mutation rate was 5% with a probabilistic mutation, the weight range was +/- 5 and the selected neuron transfer functions were hyperbolic tangent and soft sign functions which generate output within [-1,1]. In addition, recurrent connections were also included into the controller evolution. As is shown in [30], recurrent connections could make the evolution find recurrent solutions, which sometimes perform better than the feed forward solutions evolved by a standard CGP. Even if a standard CGP could solve the task, the solutions evolved by CGP with recurrent connections would be still worth investigating compared to the ones evolved without recurrent connections. In this case, recurrent connections could be an additional option if a recurrent artificial neural network controller would be considered as well to achieve the fault tolerance. The recurrent connection probability is a value between 0 and 1, which presents the probability of mutation to create recurrent connections. In this experiment, the recurrent connection probability was set 0.10, which means there is 10% possibility that the mutation could create recurrent connection. In other words, there is 90% possibility that the connections are still feed forward in the controller.

- Convergence parameters

In terms of the convergence criteria, a hyper-volume indicator would be a feasible way to demonstrate the performance of MOCGPANN as mentioned in section 4.3.2. The reference point was set (1200, 1200) in ticks, which is the maximum time point when a feasible solution can achieve. And in terms of the convergence criteria used to terminate the evolution, 30 generations would be used to look back of the observation of the hyper-volume indicator result. It needs to note that it’s still the same constraint optimization problem with MOCGPANN, so the unfeasible solutions would be not included in the 30 generations. That is to say, the 30 generations would only be working when an individual has no constraint violation. The convergence rate, which is the threshold mentioned previously, would be the division result of the current hyper-volume compared to the one 30 generations before. The convergence rate was set 1.001, which means if the current hyper-volume divided by the one 30 generations ago is less than 1.001, the evolution could be terminated. In other words, if the percentage deviation is less than 0.1% within 30 generations, the evolution would be supposed to be terminated.

4.4.1.2 Number of experiment runs and parameter adjustment

Except for the baseline settings of the mentioned experiment parameters, there are two problems that also need to be considered. One problem is the randomness inside the evolution experiment for both of optimisation algorithm and the robot task. The other problem is the parameter uncertainty for those mentioned parameters for this evolution experiment. Both of these two problems need to be considered before assessing the MOCGPANN performance for the evolution experiment.

- Cumulative mean approach

Figure 4.1 shows the evolved population from the final generation for one evolution run and Figure 4.2 shows the first Pareto optimal front solutions for it. The points on these figures represent the evolved solutions which are the obtained controllers for this evolution experiment in terms of faultless scenario objective and faulty scenario objective. The difference between these two figures is that Figure 4.1 presents the whole population for the last generation and Figure 4.2 just shows the first Pareto optimal front solutions from the last generation. Basically, it seems to be successful for MOCGPANN to obtain those controllers which can do the switched control or the robust control for fault tolerance depending on which controller to be selected from the first Pareto optimal set described in Figure 4.2.

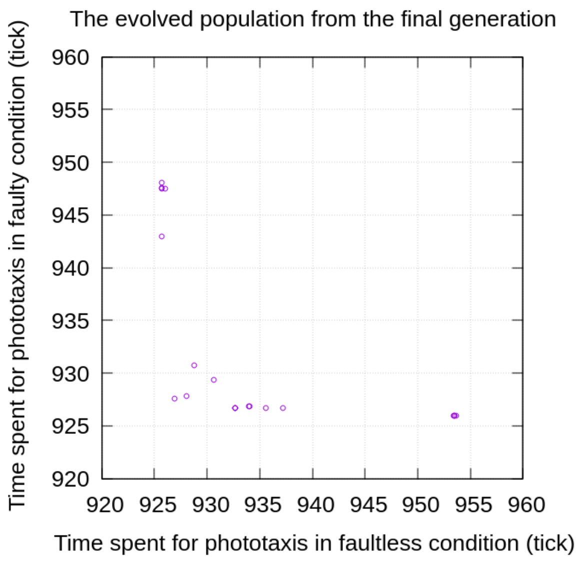

Figure 4.1: The evolved population from the final generation for one evolution run

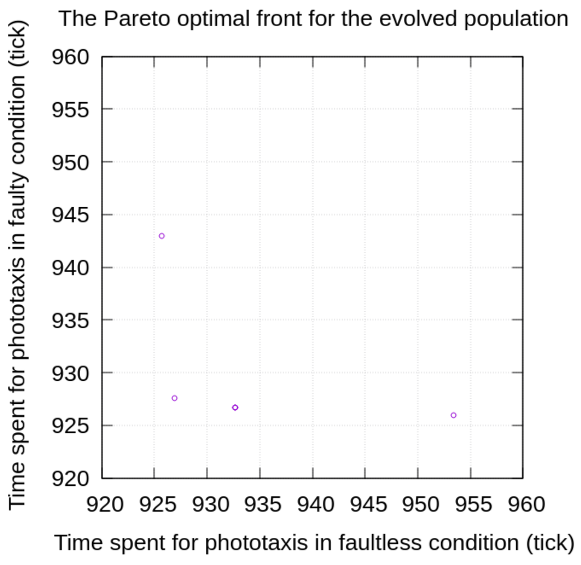

Figure 4.2: The first Pareto optimal front solutions for the evolved population from the final generation for one evolution run

However there is some randomness inside the evolution experiment. The first one is due to the random seed used to place the robot into random positions with random orientations. Since there are 10 trials to evolve controllers to make robot achieve phototaxis, so those 10 trials are based on the selected random seed utilised throughout the whole evolution experiment. That is to say, if the random seed is changed, the robot will be placed into 10 new different random positions with random orientations. In order to get the average performance, the evolution experiment has to be conducted multiple times with different random seed used for the robot placement. The second randomness also comes from the random seed but inside the MOCGPANN.

This random seed is also used throughout the evolution experiment, but it’s just used in this optimisation algorithm, such as the creation of the initial population and the genes selected to do the mutation. In this sense, this random seed also need to be changed for each evolution experiment. If both of these 2 random seeds remain constant, the evolution experiment will get the same results no matter how many times it runs. And that’s also the reason to use different random seeds for experiment.

In order to get the average performance, multiple runs need to be conducted based on different random seeds not only for robot placement but also for the MOCGPANN itself. However how many runs are sufficient to do the experiment need to be considered. For sure the experiment can obtain absolute average performance if it can be run for long enough, but it’s not realistic due the limited experiment time. In this case, cumulative mean approach [142] mentioned in section 2.7.2 could be considered as an effective approach in order to determine how many number of runs is sufficient and no more runs are required.

Table 4.1 shows part of experiment runs based on the percentage deviation. As is mentioned in [142], if there are more than one experiment output, the number of runs should be selected based on the output which needs the most number of runs. In this case, this table shows the 2 experiment response observed by percentage deviation, which are the experiment performance: hyper-volume and the convergence: number of generations. Hyper-volume measures the performance from the Pareto optimal front, so the larger the better. Number of generations indicate when the evolution is converged, so the fewer the better. As can be seen from this table, hyper-volume has already reached below 5% for percentage deviation but number of generation is not. So the total number of runs would be determined by the number of generations. From experiment index 218, the percentage deviation of generation reaches below 5% in the first time and still remains below it in the following experiments. In this sense, 218 experiments are sufficient to present the experiment results in terms of both of hyper-volume and generation number.

As can be seen from this example, cumulative mean approach would be feasible to assess how many runs are sufficient in order to get the average performance, which solves the problem of randomness for the evolution experiment and would be utilised for all of the evolution experiments throughout the thesis.

Table 4.1: Experiment index in terms of percentage deviation of cumulative mean result

Experiment index	Percentage deviation of hyper-volume	Percentage deviation of generation
208	1.0334699	5.205573
209	1.0284702	5.181981
210	1.0254355	5.165366
211	1.04011	5.147227
212	1.0351514	5.124026
213	1.0303678	5.098388
214	1.0277646	5.07409
215	1.0240977	5.05089
216	1.0202708	5.025861
217	1.0165072	5.01336
218	1.0166136	4.990012
219	1.0119224	4.983859
220	1.007815	4.973926
221	1.0037402	4.961616
222	0.9991721	4.956487
223	0.9954098	4.939457
224	0.9910932	4.922203
225	0.9871541	4.899249
226	0.9832456	4.883306
227	0.9791804	4.870532
228	0.9754496	4.84759

- Parameter Robustness technique

Apart from the randomness for the evolution experiment, the parameter uncertainty is another problem that needs to be solved before getting the optimal performance.

Figure 4.3 shows the hyper-volume response of 218 runs obtained in Table 4.1 in terms of the baseline parameters and Figure 4.4 shows the number of generation response of that 218 runs. From these 2 figures, MOCGPANN seems to be working for evolving a bank of controllers which can achieve the switched or robust control for robot sensor fault tolerance. However this evolution result is based on the baseline parameters listed in Table 4.2, which was set in section 4.5.1.1 before the experiment was conducted. There are 7 parameters for this experiment, 2 of them are set for convergence observation: Nconv and Rconv. Nconv is the number of generations looking back to observe the convergence. And Rconv implies the convergence rate which is the division result of the current hyper-volume by the one Nconv generations ago. The other 5 parameters belong to the optimization algorithm, which are NumNode:

number of node, NodeArity: number of arity for each node, WeightRange: connection weight range between each node, MutationRate: mutation rate for probabilistic mutation and RecurrentProbability: recurrent connection probability between each node. However, those parameters may not be the best combinations to demonstrate the experiment responses. In this sense, those parameters need to be adjusted in order to see whether they have any influence on the experiment responses. If so, a better combination of these parameters needs to be figured out.

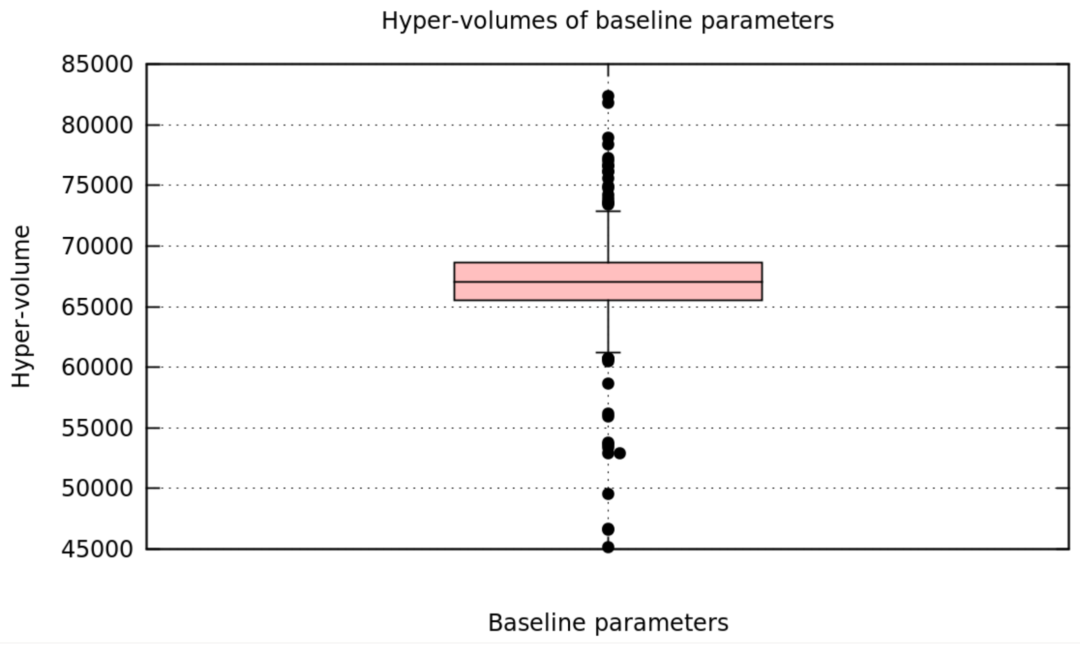

Figure 4.3: Hyper-volumes of baseline parameters

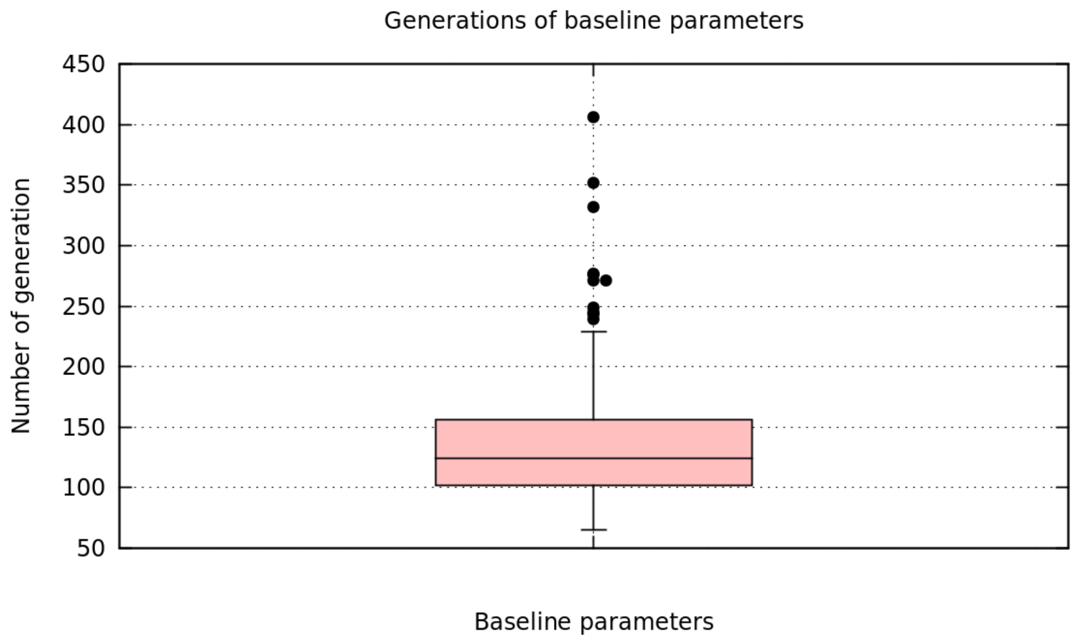

Figure 4.4: Number of generations of baseline parameters.

Table 4.2: Baseline parameter values for evolution experiment

Baseline parameters	Nconv	Rconv	Num Node	Node Arity	Weight Range	Mutation Rate	Recurrent Probability
Parameter values	30	1.001	20	5	+/-5	0.05	0.1

In this case, Parameter Robustness technique from the Spartan package [141] reviewed in section 2.7.2 would be a promising approach to help investigate how the parameters could affect the evolution experiment responses based on the utilise of A-test analysis. In terms of the evolution experiment in this work, each parameter has independent effect on the experiment responses, so Parameter Robustness technique could be an effective approach to help find out the most suited value for each parameter. To be specific, each parameter was tuned by several different values and the simulation responses of hyper-volume and generation number were compared with that from the baseline values. As is mentioned in section 2.7.1, the A-test scores could indicate whether the data set has better or worse performance compared to that of the baseline values. If the A-test score for a perturbed parameter data set is above 0.5, it means the response is below that of baseline value. If the A-test score is below 0.5, this perturbed parameter has higher response than that of baseline value. This rule is applicable no matter the response is hyper-volume or generation number. To be more specific, a higher response for hyper-volume indicates a better performance for the evolutionary search but a higher response for generation number implies a longer convergence. In other words, a lower A-test score demonstrates a better hyper-volume response and a higher A-test score indicates a better generation number response. This analysis regulation is applicable for all the experiment result analysis throughout the thesis where hyper-volume and generation number are employed as the experiment responses based on Parameter Robustness technique applied to find out the suitable parameter values.

4.4.1.3 Variants of crowding fill strategies

As is mentioned in section 4.3.1.2, the new crowding fill will make sure that only the individual with the largest distance to its parent will be allowed to take part in the crowding measure during the survival when there are more than one individual with the same objective values. And then if there are still places available, the left individuals will be survived one by one based on their distance values. The benefit of this new crowding fill is that the random neighbour pick problem could be solved and the population diversity could be preserved as well, which may be considered as a better strategy than the original crowding fill developed in NSGA2. For this reason, this new crowding fill would be the first version utilised in the evolution experiment as the baseline performance.

As a comparison, two more variants would also be utilised to conduct the same evolution experiment. The second version of crowding fill strategy just picks one random child into the crowding distance measure when more than one individual have the same fitness and the left individuals with their corresponding same fitness will also be randomly survived. It needs to note that the children will be still survived ahead of parents until the next generation is fulfilled, which is able to preserve the genetic drift.

The last version is similar to the second one and the only difference is that when more than one individual have the same fitness, one random individual is allowed to participate in the crowding measure rather than just one random child from the second version and the other individuals will also be survived randomly with their corresponding same fitness. This version disables the population diversity as well as the genetic drift preservation. So it would be suited as a comparison with the first two approaches and the comparison may be also helpful to investigate whether preserving genetic drift or population diversity has any impact on the experiment responses.

To make it clear, the pseudo code for each version of the crowding fill strategies for MOCGP is listed as below along with the original crowding fill strategy from NSGA2. Generally speaking, the first version has three steps to complete a whole crowding fill during the survival selection while other three options only have one step. The first version calculates the distance values first for the individuals with the same fitness. And then it ranks them according to their distance values. Finally, the new crowding fill will be executed. In terms of the other three options, there is no need to differentiate each individual with the same fitness as the first version. For this reason, only the crowding fill is executed without any information about the distance values for the individuals.

- The first version

```
1. // set distance for the individuals if they have the same fitness with their parents
Travers each individual, if the current individual has the same fitness with its parent {
  If the individual is created from normal mutation {
    Set Euclidean distance for the individual as fitness distance
  }
  Else if the individual is created from neutral mutation {
    Set Hamming distance for the individual as genetic distance
  }
}
2. // rank the individuals with their distance values
Traverse each individual, if more than one individual have the same fitness {
  Rank those individuals with the order:
  1.  child from normal mutation
  2.  child from neutral mutation
  3.  parent from normal mutation
  4.  parent from neutral mutation
}
3. // new crowding fill
Execute the normal survival selection
If the current Pareto front size is larger than the left available places for the individuals
to be survived into the next generation {
  // execute the new crowding fill
  Traverse each group of the same fitness and pick the individual ranking the first into the new Pareto set combined with the other individuals of the unique fitness in the original Pareto set
  Execute crowding measure for the new Pareto set
  After the new Pareto set is survived, if there are still places available for individuals to be survived into the next generation {
    Merge all the left individuals together and survive them based on their distance values from large to small no matter whether they have the same fitness or not until the next generation is full
  }
}
```

- The second version

```
Execute the normal survival selection
If the current Pareto front size is larger than the left available places for the individuals to be survived into the next generation {
  // execute the new crowding fill
Traverse each group of the same fitness and pick a random child into the new Pareto set combined with the other individuals of the unique fitness in the original Pareto set
  Execute crowding measure for the new Pareto set
  After the new Pareto set is survived, if there are still places available for individuals to be survived into the next generation {
    Merge all the left individuals together and survive the children first and then the parents no matter whether they have the same fitness or not until the next generation is full
  }
}
```

- The third version

```
Execute the normal survival selection
If the current Pareto front size is larger than the left available places for the individuals to be survived into the next generation {
  // execute the new crowding fill
Traverse each group of the same fitness and pick a random individual into the new Pareto set combined with the other individuals of the unique fitness in the original Pareto set
 Execute crowding measure for the new Pareto set
 After the new Pareto set is survived, if there are still places available for individuals to be survived into the next generation {
   Merge all the left individuals together and survive the individuals randomly no matter whether they are parents or children and whether they have the same fitness or not until the next generation is full
 }
}
```

- The original crowding fill

```
Execute the normal survival selection
If the current Pareto front size is larger than the left available places for the individuals to be survived into the next generation {
 // execute the original crowding fill
 Execute crowding measure for the current Pareto set
 Survive the individuals based on their crowding distance values from large to small until the next generation is full
}
```

4.4.2 Generalisation experiment

Due to the failure of CGPANN evolved controllers to achieve the robust fault tolerant control in chapter 3, the aim of this generalisation experiment is to investigate how the evolved controllers by MOCGPANN will be performing for the robust and switched fault tolerant control as well in terms of the robot phototaxis task based on a series of generalised test scenarios. As is mentioned in section 3.3.2.3, the benefit of a multi-objective optimization algorithm is to obtain a Pareto set of controllers in just one evolution loop, which could not only achieve the robust fault tolerant controller but also realise the switched fault tolerant control. In this case, the evolved controllers will be tested for robust and switched fault tolerant control respectively. Moreover, in order to obtain a generalised performance, the fault tolerance settings as well as the phototaxis task framework will both be set in a more general way rather than the original scenario utilised for the evolution experiment. The aim for the resetting scenarios is to test how the evolved Pareto sets of controllers will be working in a different scenario and whether they have the capability for the generalisation.

4.4.2.1 Fault tolerant control type

The whole experiment will be conducted with two approaches: the robust fault tolerant control and switched fault tolerant control. The switched control utilises the Pareto set controllers as bank controllers. Basically only two controllers are sufficient to achieve the switched control since there are only two objectives. So one controller could be the one that works best for no fault condition and the other could be the one that works best for the faulty condition, no matter what the performance is for the opposite objective. When the experiment starts, the controller for the normal condition is loaded at the beginning. Once the fault occurs, the other controller will be loaded to replace the current one in order to tolerate the fault. It needs to note that this work is mainly about the controller redesign in fault tolerant control mentioned in section 2.2.2, so it is assumed that the fault has already been diagnosed. In this sense, when the fault is occurred, the other controller will be loaded immediately without any delay for the fault diagnose. In terms of the robust fault tolerant control, it just utilises one controller to be robust for both of no fault and faulty conditions. In this case, it is like a trade-off that the selected controller has to perform well for both of these two objectives. Definitely the performance for each objective will be degraded compared to the switched control, but there is no need to carry another controller on board especially for the real world experiment and no need to consider the controller switch as well, which are the significant benefits for robust control.

4.4.2.2 Generalised test scenarios

The evolution experiments designed in section 4.5.1 will investigate whether MOCGPANN could evolve a Pareto set of controllers for two objectives respectively in terms of different parameters and crowding fill strategies. Those obtained Pareto sets of controllers are evolved based on the objectives that the robot is failed from the beginning or totally no fault during the experiment, which will be an effective way to solve the problem based on the selection of a trade-off controller to achieve the robust fault tolerant control where CGPANN was failed as mentioned in the conclusion of chapter 3. It needs to note that if MOCGPANN will be able to obtain feasible Pareto sets of controllers, a suitable robust controller can be just selected from the Pareto set without any further test since this controller is already evolved based on the robustness against both the faultless and faulty objectives. However, it is still not clear whether the robot could achieve the fault tolerance when the faults are occurred during the experiments, which is a common situation for investigating fault tolerance problems. Moreover the sensor faults are just complete failures with input reading as 0 for the faulty sensors. Actually in real world scenarios, the complete failure may be any constant random input reading signal, so it is also worth trying random faulty sensor readings as the faulty signal and see whether the evolved controllers are also working for this situation. Apart from that, each run of the experiment is based on its own randomly selected robot initial positions and orientations with fixed distance from the beacon location. And those robot initial conditions will not be changed during the evolution. However the phototaxis task designed in section 3.2.1 actually refers to any initial conditions for the robot with random distance to the beacon position as long as the light can be detected by the robot light sensor. In this case, trying different robot initial conditions also needs to be considered with different distance to the beacon position as well. In summary, if all the above scenarios could be considered, the evolved controllers can be tested in a more general way not only fault tolerance but also for phtotaxis experiment.

- Initial scenario

The initial scenario set the different fault occurrence time. This scenario utilised 300 ticks and 700 ticks respectively to trigger the fault rather than no fault and fault from beginning designed in section 3.2.4 in order to investigate whether a basic fault tolerant control scheme could be achieved when the fault is occurred at different time during the experiment. If the fault could be tolerated for this initial scenario, there would be three more different conditions added to the initial scenario respectively to test the evolved controllers.

- New faulty signal scenario

Firstly, the sensor faulty signal will be changed to another constant value such as 0.5 instead of 0 for complete failures. This situation will investigate whether the evolved controllers will also tolerate a different faulty sensor signal to achieve the phototaxis.

- New robot starting position and orientation scenario

Secondly, 10 robot initial position and orientation combinations from the evolution experiment will be changed to another different 10 combinations in order to investigate whether the evolved controllers could make the robot achieve phototaxis in different initial conditions, which is the same as the additional scenario designed in section 3.2.4.

- New beacon location scenario

Finally, the beacon location will also be modified. In the evolution experiment, the beacon is located at the origin of the arena, which is (0, 0). And the robot is placed in 10 initial conditions to evaluate controllers with a fixed distance 4.5m away from the beacon. However in this generalisation experiment, the beacon will be moved to (2, 2) of the arena. In this case, the distance between the robot initial position and the beacon will be varied depending on how far the robot is away from the new position of the beacon. There are two points to be mentioned here that (2, 2) actually could guarantee that the robot can detect the light from their initial positions. However, if the robot moves in a reserved way towards the light, the robot will not detect the light finally and will not achieve the phototaxis forever. In other words, the evolved controller is not capable to make the robot achieve phototaxis in the face of fault if the robot is moving away from the beacon. The other point is that since the distance between the robot and light is varied, so 1200 ticks will not make sense to judge the controller performance for the generalisation experiment. In this case, 3000 ticks will be used instead of 1200 ticks as the maximum experiment time limit and the success rate will be the new criterion instead of time to the beacon in order to assess the evolved controllers whereby whether the robot can reach the beacon finally or not.

These three extra conditions would be added to the initial scenario of the generalisation experiment one at a time. So the result will demonstrate which one has the most significant impact on the evolved controllers in terms of the robust and switched fault tolerant control.

4.4.2.3 Controller acquirement

In terms of the selected controllers used to conduct the generalisation experiment, section 4.5.1.2 will present more details about the comparison result among different versions of crowding fill strategies utilised for MOCGPANN in order to select the best one. In summary, the first version of MOCGPANN would be an initial choice to evolve the Pareto set of controllers in order to achieve the switched and robust fault tolerant control respectively. Although there is actually no significant difference among the four versions of MOCGPANN, the first and second one achieved the relatively more stable performance. In this sense, the first version could be utilised as a primary approach to obtain the Pareto sets of controllers. The comparison details for different MOCGPANN versions can be referred to section 4.5.1.2. On the other side, the aim of the generalisation experiment is actually a test of the evolved controllers to see how they will perform in more general cases, so which version of MOCGPANN to be used for the controller evolution is actually not the point at this stage. For each of the switched and robust fault tolerant control, five Pareto set of controllers from five evolution runs are utilised for the generalisation experiment and their performance will be assessed in terms of the above mentioned different scenarios.

4.5 Result and discussion

4.5.1 Evolution experiment

In terms of the evolution experiment, firstly the parameters for both of MOCGPANN and convergence criteria were adjusted in order to obtain the optimal performance. And then, different crowding fill strategies were compared to conduct the same evolution experiment so as to investigate whether preserving population diversity or genetic drift has any significant benefit on the evolutionary search based on the evolution experiment.

4.5.1.1 Sensitivity analysis for MOCGPANN parameters

As is mentioned in section 4.4.1.2, the baseline parameter values may not be the best combination to display the optimal performance for evolution experiment. In this case, the parameters should be adjusted before analysing the experiment result. With the help of Parameter Robustness technique, those parameters were tuned to their optimal calibration values. The details of how these parameters were tuned by the Parameter Robustness technique can be referred to Appendix B.

Based on the result of the sensitivity analysis, an ultimate comparison could be conducted between the baseline parameter values and the calibration parameter values in terms of both hyper-volume and generation number. The calibration parameter values are the ones based on the sensitivity analysis results. The hyper-volume comparison is shown in Figure 4.5 and generation number for that is shown in Figure 4.6. The U-test scores for them are listed in Table 4.3 and the A-test scores for them are listed in Table 4.4.

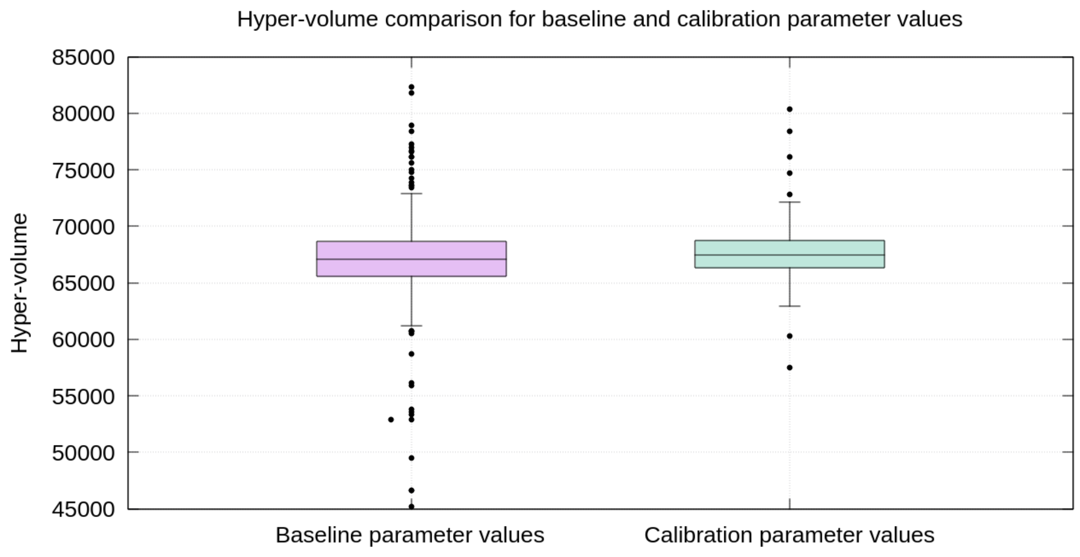

Figure 4.5: Hyper-volume comparison between baseline and calibration parameter values

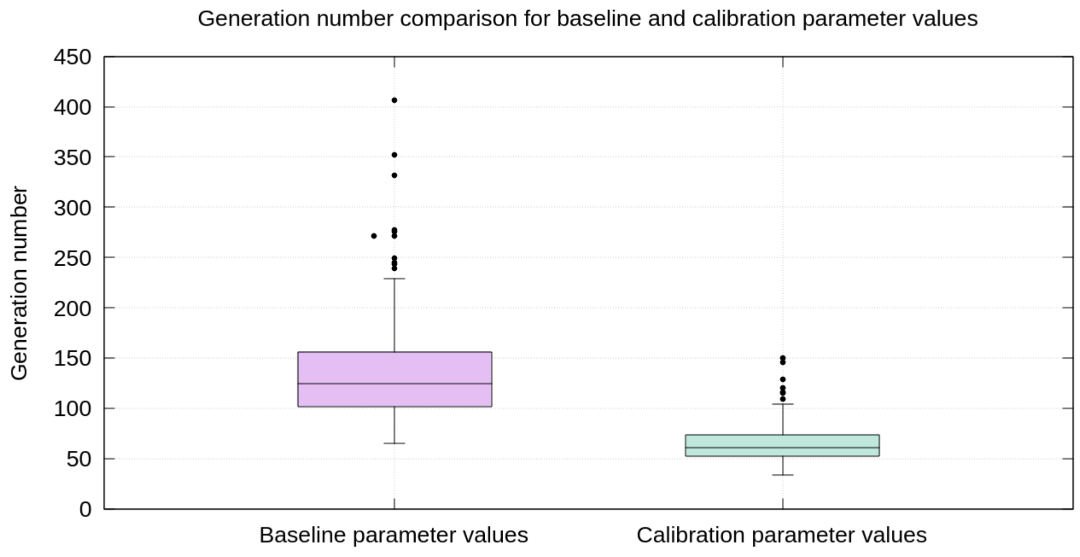

Figure 4.6: Generation number comparison between baseline and calibration parameter values

Table 4.3: U-test scores for the comparison between baseline and calibration values

	Baseline parameter values	Calibration parameter values
Hyper-volume	1	0 .11876
Generation number	1	< 0.00001

Table 4.4: A-test scores for the comparison between baseline and calibration values

	Baseline parameter values	Calibration parameter values
Hyper-volume	0.5	0.455692
Generation number	0.5	0.954808

According to the ultimate comparison result, the calibration parameter values still outperform the baseline parameter values. In term of hyper-volume, although they are quite similar, calibration values still achieve a slightly better performance, where the A-test score is below 0.5. Even if they have got similar hyper-volume responses, calibration values spend much less generations to make evolution converged than baseline values, which demonstrates that the sensitivity analysis is quite essential before analysing the experiment responses. In a word, the calibration parameter values outperform the baseline parameter values, especially for the response of generation number.

In summary, all the parameter values are now determined based on this sensitivity analysis in terms of Parameter Robustness technique developed in Spartan and the optimal values for each parameter are listed in Table 4.5. The following experiments will be conducted based on these calibration parameter values throughout the thesis.

Table 4.5 Calibration parameter values for evolution experiment

Calibration parameters	Nconv	Rconv	Num Node	Node Arity	Weight Range	Mutation Rate	Recurrent Probability
Parameter values	20	1.01	20	5	+/-10	0.05	0

4.5.1.2 MOCGPANN comparison based on modified crowding fill strategies

As can be seen from the previous results of sensitivity analysis in terms of parameter robustness technique, all of the parameters have been adjusted to their optimal values in order to get the best simulation responses for both of Hyper-volume and generation number. This section will investigate whether the population diversity could really make a contribution to the experiment responses based on the modified crowding fill strategy for survival selection in MOCGPANN.

To obtain a sensible comparison for how population diversity affects the experiment response, three different versions of crowding fill strategies were utilised as mentioned in section 4.4.1.3. Figure 4.7 shows the hyper-volume comparison for these three different versions of MOCGPANN. Figure 4.8 shows the generation number comparison for that. Table 4.6 lists the U-test scores for these two comparisons and Table 4.7 presents the corresponding A-test scores for them, where the first version of crowding fill was utilised as the baseline performance.

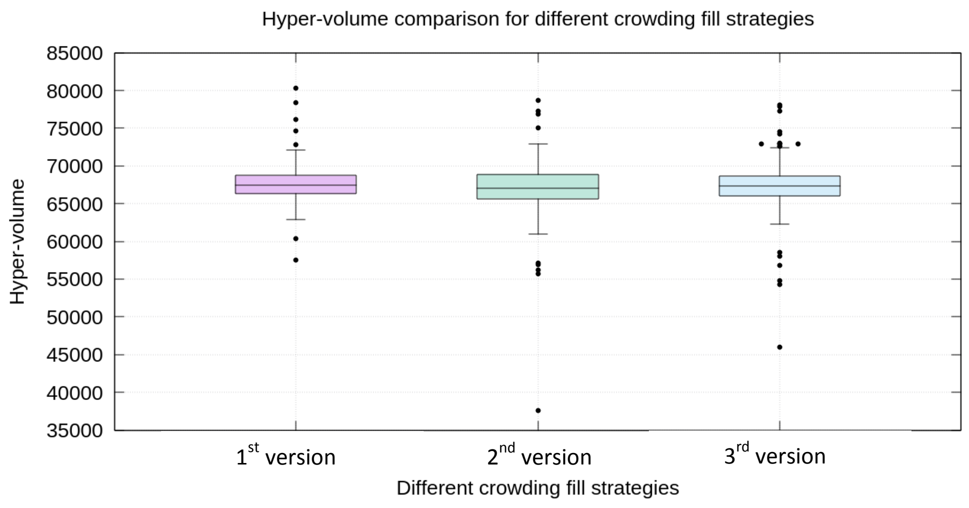

Figure 4.7: Hyper-volume comparison for three different MOCGPANN

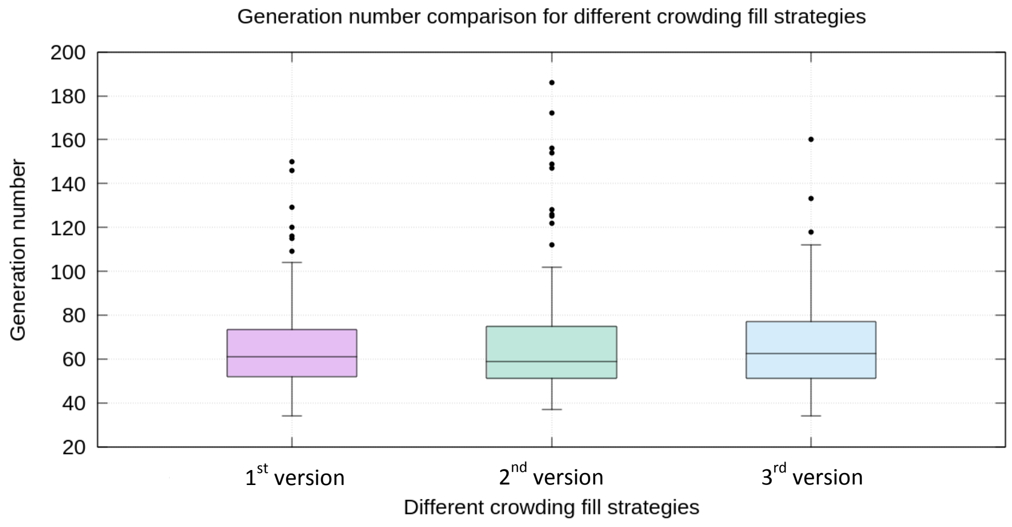

Figure 4.8: Generation number comparison for three different MOCGPANN

Table 4.6: U-test score for three different MOCGPANN comparisons

	1st version	2nd version	3rd version
Hyper-volume	1	0 .05614	0 .4965
Generation number	1	0 .33204	0 .92034

Table 4.7: A-test score for three different MOCGPANN comparisons

	1st version	2nd version	3rd version
Hyper-volume	0.5	0.554296	0.520363
Generation number	0.5	0.528675	0.494143

As can be seen from Table 4.6, only the second version achieves a p value near 0.05 in terms of hyper-volume, which means its hyper-volume may have a significant difference compared to the first version. However according to Table 4.7, its corresponding A-test score is 0.554296 above 0.5, which indicates that although there may be a significant difference, the hyper-volume of second version is lower than that of the first version. Actually the A-test score of 0.554296 for the second version's hyper-volume response is even not located in the small effect size interval [0.56, 0.64], so this difference between the second and first version for hyper-volume is actually quite weak. Apart from the second version's hyper-volume response, other responses' A-test scores for both of second and third are much more close to 0.5 with nearly no effect sizes. In conclusion, all of these three versions of crowding fill strategies spend quite similar generations to make the evolution converged. In terms of the hyper-volume, the second version obtains the worst response but still could be ignored and the third one is still quite similar to the first one.

Generally speaking, all of these three versions actually obtained similar experiment responses, which indicates that the preservation of genetic drift as well as population diversity may not improve or even effect the experiment responses. A possible explanation is that the objective value is the worst performance during 10 trials for the phototaxis. In other words, although the worst case is the same, the performance of the other 9 trials may be quite different among each individual. So it may be not a suitable choice to evaluate individuals by using the worst case performance as the final objective value. In addition, utilising the worst case performance as the final objective value could also result in the individuals with identical fitness in spite of different performance for the other trials.

In this case, a higher precise objective value need to be considered based on another alternative fitness function for each individual evaluation instead of the current one. Generally speaking, the mean approach would be more sensible than the worst case approach as the alternative fitness function in order to reflect how these trials are distributed since the final objective value will be the average value among all the trials. However the low precise objective value problem is not fully resolved since the obtained objective value is still in a low precise scope, even if the average value is more sensible than the worst case value as the final objective value. In this case, a weighted sum approach would be more effective to not only reflect the trial value distribution but also solve the low precise objective value problem. The only problem that needs to be solved is how to set the weight value for each trial. In general, the weight is determined by the frequency that the result is repeated. However in this experiment, each trial may have its own value that is different between each other, so the frequency may be meaningless to act as the weight value. In this case, the weight value could be set by how large the result is using the current trial value divided by the sum of all the trial values. In this way, a larger trial value will have a larger weight value for this trial and vice versa. This approach is feasible since the final objective value will be mainly affected by the large trial values due to their large weights so that the solutions with extremely large trial values could be eliminated during the selection, which may have similar effects with the worst case approach.

For example, there are two solutions to be evaluated and there are three trials for each individual evaluation. The first solution's trial values are 910, 950 and 990 and the second solution's trial values are 930, 950 and 970. If the mean approach is utilised to act as the final objective value, both of them will get 950 since 950 are the average value for both of them. In this case, there is no way to differentiate these two solutions. However if the weighted sum approach is utilised, the result will be quite different. The final objective value of the first solution will be 951.1228 and the value of the second solution will be 950.2807. In this case, the two solutions will be easily distinguished due to their high precise objective values. Moreover, the second solution also outperforms the first one since the first one has got a largest trial value of 990 among the trials in these two solutions, which guarantees that the solution with larger trial values will be eliminated during selection due to its larger final objective value.

In conclusion, this kind of weighted sum approach is actually utilising the sum of squares of the trial results divided by the sum of the trial results. This fitness function will produce a measure that is more sensitive to the large values than the small ones with a higher resolution objective value. In this case, the final objective value may be around the average value among those 10 trials but slightly closer to the worst one. In addition, this fitness function will also result in a higher resolution of the objective value with the addition of decimal part instead of a total integer value. In this way, the distinction among identical fitness individuals will also be promoted that the individuals will have a lower chance to get stuck into the same fitness with the others. It needs to note that this modified fitness function is still working coupled with the constraint function so that only the individuals with no constraint violation will be allowed to be evaluated on the fitness function. On the other hand, the original crowding fill strategy referred directly from NSGA2 [92] in the survival selection for MOCGPANN algorithm will be also used as an additional comparison with the current 3 different versions. In this sense, it will be more sensible to demonstrate whether the modified crowding fill strategy will be working better or not compared to the original one.

Figure 4.9 shows the hyper-volume comparison for four different crowding fill strategies with modified fitness function and Figure 4.10 shows the generation number comparison for them. Table 4.8 lists the U-test scores for the comparisons and Table 4.9 lists the A-test scores for them, where the first version of crowding fill is still considered as the baseline performance.

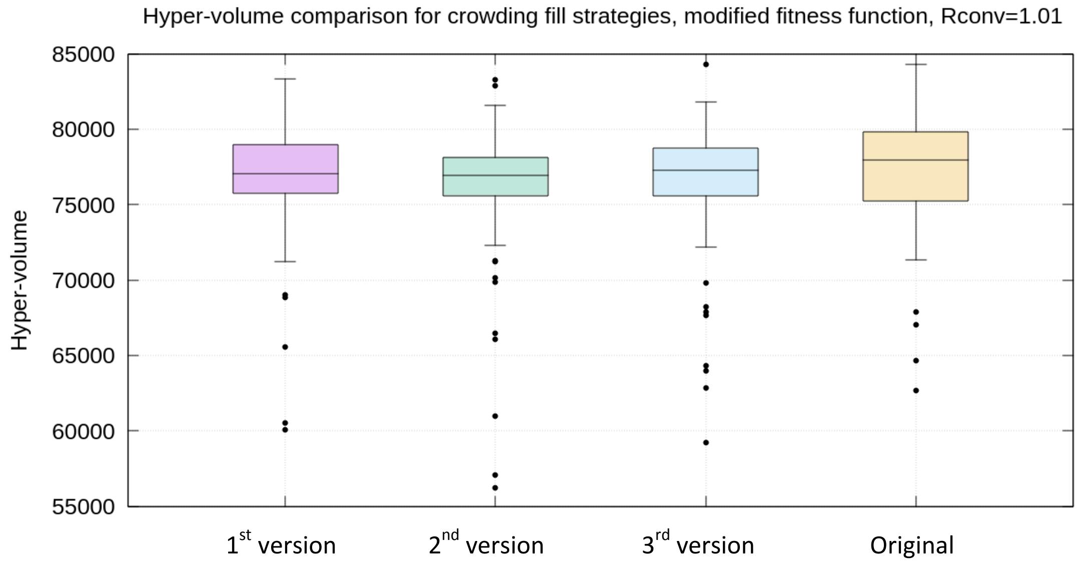

Figure 4.9: Hyper-volume comparison for four different crowding fill strategies with modified fitness function

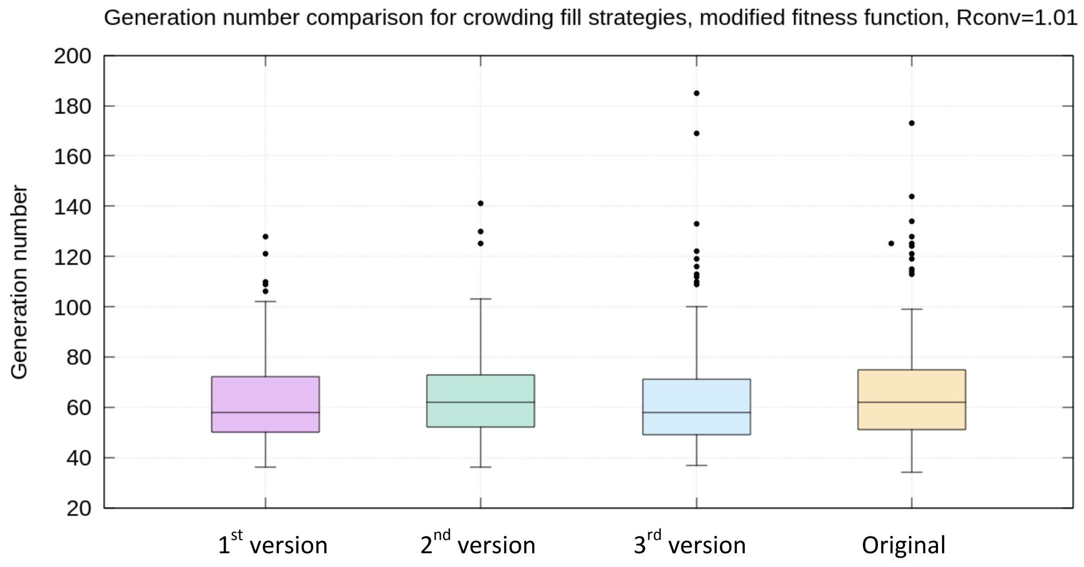

Figure 4.10: Generation number comparison for four different crowding fill strategies with modified fitness function

Table 4.8: U-test score for four different crowding fill strategies comparisons with modified fitness function

	1st version	2nd version	3rd version	Original
Hyper-volume	1	0.37886	0.58232	0.12602
Generation number	1	0.3843	0.64552	0.27572

Table 4.9: A-test score for four different crowding fill strategies comparisons with modified fitness function

	1st version	2nd version	3rd version	Original
Hyper-volume	0.5	0.5313329	0.4838644	0.4520385
Generation number	0.5	0.4682333	0.5130248	0.4614715

As can be seen from Table 4.8, neither of second or third version has significant difference compared to the first version in terms of hyper-volume and generation number. Although the fourth one has the smallest p values, they are still above 0.05 to some extent. According to Table 4.9, all of them have nearly no effect compared to the first one where all the A-test scores are even smaller than the small effect size interval of [0.36, 0.44] and [0.56, 0.64] in terms of hyper-volume and generation number. To be more specific, second version spends more generations to obtain lower hyper-volume than the first version and third version spends slightly less generations to achieve slightly higher hyper-volume than the first one. The fourth version actually achieves the highest hyper-volume, but the generation number is also the largest. Even though, all of these four versions obtain similar experiment responses in terms of both of hyper-volume and generation number. So it could be concluded that these four versions of crowding fill strategies have no significant difference between each other for the impact of survival selection in MOCGPANN even the fitness function is modified to obtain more precise objective values.

A possible explanation for this result may be due to the effect of connection weight genes which make the evolutionary search less likely to be trapped into local optima than those algorithms without weight genes [30]. As is mentioned before in section 2.4.4.5, [30] conducted a comprehensive investigation on how explicit neutral genetic drift impacts CGPANN for evolutionary search. However the benefit of neutral genetic drift is much lower for CGPANN than for CGP, where the benefit of preserving explicit neutral genetic drift is totally absent. Based on the analysis of the comparison experiment, the only difference between CGP and CGPANN in that work is the existence of connection weight genes and a higher node arity. Other aspects between these two algorithms are the same for the comparison experiment. However [30] demonstrates that a higher arity may not have any influence on the average number of explicitly inactive genes, which is not the cause for explicit neutral genetic drift being useless. Moreover, [30] also indicates that increasing the available number of nodes may not increase the number of inactive nodes as well, which is also not the reason for promote the benefit of explicit neutral genetic drift. In this case, the only reason that results in explicit neutral genetic drift being absent is the utilisation of connection weight genes, which maybe acts as a compensation for evolutionary search in CGP. [30] infers that due to the additional mutation occurred on connection weight genes, CGPANN may be not so easily trapped into local optima and that's why explicit neutral genetic drift does not present the benefit to aid the escape from local optima during the evolutionary search.

Based on the analysis from [30], it could be concluded that MOCGPANN may also not benefit from the preserving of explicit neutral genetic drift as well as the population diversity. Although the investigation in [30] is based on CGPANN, it can still be inferred that MOCGPANN will be suffered from the same problem due to the existence of connection weight genes. In conclusion, the preserving of explicit neutral genetic drift and the population diversity will not make any impact on the evolutionary search in terms of MOCGPANN. Just as mentioned in [30], it can be also concluded that any approach in NE area will not benefit from the neutral genetic drift and other form of gene redundancy in spite of single or multiple objective optimisation.

Although four versions of crowding fill strategies have no significant difference among each other in terms of both the hyper-volume and generation number, it still needs to note that the number of experiment runs measured from the cumulative means approach is different for each version. Table 4.10 lists the cumulative mean approach result for each of four versions.

Table 4.10: Result of number of experiment runs required from cumulative mean approach

	1st version	2nd version	3rd version	Original
Experiment runs	110	129	189	173

As can be seen from Table 4.10, the first and second versions need less experiment runs than the other two to sufficiently present the experiment responses. This phenomenon may indicate that the first version achieves the most stable performance with the least required number of experiment runs to fully demonstrate its experiment responses. And the first version also requires the similar number of runs as the second one. A possible explanation would be that the first two versions may have more stable responses for both the hyper-volume and generation number with much less required number of experiment runs than the other two. In other words, the other two versions may not have so stable responses as the first two with some extremely bad responses and that's why more runs are still required to sufficiently present their performance.

In this case, it may be concluded that preserving genetic drift and population diversity might have potential benefit to help MOCGPANN achieve a more stable performance. However as mentioned before, these four different versions actually have no significant difference without any effect size among each other in terms of both the hyper-volume and generation number. So whether preserving genetic drift or population diversity really has any significant benefit for the evolutionary search is still not clear at the moment and further investigation is still required as the future work.

4.5.2 Generalisation experiment

In terms of the generalisation experiment, robust and switched fault tolerant control would be both discussed respectively based on their performance. The obtained five Pareto sets of controllers were tested 10 times for each of 300 and 700 ticks as the fault occurrence time based on four different scenarios including a basic scenario and three additional scenarios. The data of these five Pareto sets' generalisation experiment result can be referred to the Appendix A.1 and their success rate for each scenario would be discussed as following to assess the obtained controllers' performance for robust and switched fault tolerant control respectively.

4.5.2.1 Robust fault tolerant control

In terms of the robust fault tolerant control, only one robust controller is required to be tested for the generalisation experiment. This controller has to be working relatively well for both of faultless and faulty conditions, so its performance will be possibly degraded compared to two bank controllers working best for each objective respectively. The robust fault tolerant control utilises the robust controllers from 5 Pareto sets and each robust controller is selected based on its trade-off performance in terms of each objective from the controllers in the Pareto optimal front. Furthermore, each robust controller from the corresponding Pareto set will be tested 10 times in terms of different scenarios. The success rate result of the selected 5 robust controllers from these 5 Pareto sets is listed in Table 4.11 in terms of four tested scenarios.

Table 4.11: Success rate for generalisation experiment results in terms of robust fault tolerant control

	Initial scenario		Fault signal		Robot condition		Beacon position	
	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700
Pareto 1	1	1	0	1	1	1	1	1
Pareto 2	1	1	0	0	1	1	1	1
Pareto 3	1	1	0	1	1	1	1	0.6
Pareto 4	1	1	0	0	1	1	0.7	0.7
Pareto 5	1	1	1	1	1	1	0.7	0.7

As can be seen from Table 4.11, all the 5 robust controllers achieve 100% success rate for initial scenarios. What's more, they also obtain 100% success rate for new robot initial conditions in addition to different fault occurrence time. However they are also not working well on new fault signal and new beacon position scenarios. In terms of new fault signal scenario, only Pareto set 5 realises 100% success rate for both the different fault occurrence time but the other 4 sets are all failed to reach the beacon within 10 trials when fault occurs at 300 ticks and half of them also cannot make it when fault occurs at 700 ticks. As for the new beacon position scenario, only the first two Pareto sets achieve 100% success rate but the other three have more or less failed trials for both the different fault occurrence time.

As can be seen from the result, MOCGPANN is successful to evolve controllers so as to achieve the robust fault tolerant control where CGPANN was not able to complete. In addition, different fault occurrence time for the initial scenario has no influence on the evolved controllers' performance and new robot initial condition scenario also does not affect those controllers. However both of new fault signal and new beacon position scenarios have more or less impact on the evolved controllers, which demonstrates that the obtained controllers are not suited to these generalised scenarios. In this situation, those scenarios need to be considered for evolution experiment including a set of random fault signals and varied beacon positions for each solution evaluation so as to obtain more robust controllers to accomplish the design of a robust fault tolerant control system for the robot phototaxis task, which could be investigated as future works.

4.5.2.2 Switched fault tolerant control

In terms of the switched fault tolerant control, two controllers are selected as bank controllers from the same 5 Pareto optimal sets. One controller is selected working best for the normal situation and the other one is selected working best for the faulty situation. These two bank controllers will be switched when the fault occurs during the robot online phototaxis task based on the assumption that the fault has already been diagnosed. Moreover, the switched fault tolerant control will also be tested 10 times for each generalised scenario respectively and the success rate result for these 5 pairs of bank controllers from these 5 Pareto sets is listed in Table 4.12 in terms of four tested scenarios.

Table 4.12: Success rate for generalisation experiment results in terms of switched fault tolerant control

	Initial scenario		Fault signal		Robot condition		Beacon position	
	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700
Pareto 1	1	1	1	1	1	1	1	1
Pareto 2	1	1	0	0	1	1	1	1
Pareto 3	1	1	0	1	1	1	1	1
Pareto 4	1	1	1	1	1	1	0.7*	0.7*
Pareto 5	1	1	1	1	1	1	0.9	0.8

Note: * means that in terms of the failed trials, the robot stays still until the other controller is loaded. However these trials all make the robot reach the beacon finally, so it is actually 10/10 if the criterion is whether the robot achieves the phototaxis eventually or not.

As can be seen from Table 4.12, all of the 5 pairs of bank controllers from these 5 Pareto sets achieve 100% success rate for initial scenarios in terms of both of 300 and 700 ticks for fault occurrence time. However, only Pareto set 1, 4 and 5 obtain 100% success rate for the new fault signal scenarios. Pareto set 2 just performs a 0% success rate at all and Pareto set 3 is only 100% successful for 700 ticks as fault occurrence time but 0% for 300 ticks. In terms of the new robot initial condition scenario, all of the 5 Pareto sets realise 100% success rate for both of 300 and 700 ticks as fault occurrence time. Finally, only Pareto set 1, 2 and 3 achieve 100% success rate for the new beacon position scenario. Neither Pareto set 4 nor 5 realises a 100% success rate in terms of different fault occurrence time. It needs to note that Pareto set 4 is not really 70% successful for the new beacon position scenario since all the trials make the robot reach the beacon eventually. However there are 3 trials that the robot stays still at its initial position in the faultless condition and starts to move towards the beacon when the fault occurs with the loaded new controller. So these 3 Pareto sets actually suite the robot faulty condition but not for the normal condition, which may not be considered as successful phototaxis task.

As can be seen from the result, the controllers evolved by MOCGPANN could not only achieve the robust fault tolerant control but also realised the switched fault tolerant control as well, which also demonstrates the benefit of multi-objective optimisation algorithm mentioned in section 3.3.2.3. These 5 Pareto sets of controllers perform well on the initial scenario and the new robot initial condition scenario but also do not perform very well in the new fault signal and the new beacon position scenarios with more or less declined success rate. This phenomenon indicates the same conclusion in section 4.5.2.1 that the evolved controllers from the evolution experiment are capable to perform well no matter when the fault occurs during the experiment and regardless where the robot initial condition is as long as the distance from the beacon is fixed. However, if the fault signal is altered rather than 0 from the evolution experiment, some of the controllers are not able to make the robot complete phototaxis within 1200 tick time limit. In addition, when the beacon is moved to a new location, some of the controllers are also not capable to make the robot reach the beacon even if there is no time limit. As a consequence, if a more effective switched fault tolerant control system for phototaxis is required in terms of varied fault signals and beacon positions, those scenarios need to be considered during the solution evaluation, which is the same as that mentioned in the conclusion of section 4.5.2.1 for future works.

4.5.2.3 Comparison

Based on the generalisation experiment result obtained by the five evolved Pareto sets of controllers for robust and switched fault tolerant control respectively, this section will conduct a comparison between these two different faulty tolerant control schemes and find out which one achieved a better performance.

Table 4.13 lists the A-test scores for the four different scenarios' result in terms of robust and switched fault tolerant control comparison. This comparison also follows the A-test analysis rule mentioned in section 4.4.1.2 that a lower A-test score means a higher response. In this comparison, robust fault tolerant control result was used as the baseline performance, so the A-test score will reflect the switched fault tolerant control performance compared to the robust one.

Table 4.13: Comparison between the success rate of robust and switched fault tolerant control based on the controllers evolved by MOCGPANN

Scenario	Initial scenario	Fault signal	Robot condition	Beacon position
A-test score	0.5	0.388889	0.5	0.37037

As can be seen from Table 4.13, the A-test score for the initial scenario and the new robot condition scenario is both 0.5 which indicates an identical performance for robust and switched fault tolerant control for these two scenarios. However in terms of the new fault signal and new beacon position scenarios, the A-test scores are both below 0.5 and located in the small effect size interval [0.36, 0.44]. This result means that switched fault tolerant control outperforms robust fault tolerant control in terms of the success rate for these two scenarios. Although the effect size is small for these two scenarios, at least it demonstrates that switched fault tolerant control produced a better performance than the robust one, which could be considered as a main approach for the offline designed controller in this work.

Actually it is normal to obtain this comparison result since two bank controllers will definitely outperform a single robust controller in terms of each objective as mentioned in section 3.3.2.3. But the result does not indicate that robust control is not suited to fault tolerance. As is also mentioned in section 3.3.2.3, robust fault tolerant control saves the memory to store one more controller on board and there is no need to design a controller switch mechanism. The most important aspect is that robust control belongs to the passive fault tolerant control mentioned in section 2.2.1 which does not need fault diagnose procedure if it is required, when fault occurs during the system operation. So robust fault tolerant control saves lots of work to do, but the degraded performance cannot be ignored as well.

In conclusion, MOCGPANN demonstrates the capability to evolve controllers which could be working in more general cases in terms of both the switched and robust fault tolerant control based on robot phtotaxis task. Although not all the scenarios are working well for MOCGPANN evolved Pareto sets of controllers, MOCGPANN still demonstrates the potential capability to evolve promising controllers for generalisation. On the other hand, switched fault tolerant control produced a better performance than the robust one, which could be considered as a main scheme for fault tolerant control in this work. Future work would be considering certain scenarios such as random fault signal and varied beacon position to evolve controllers in order to see how the evolved Pareto sets of controllers will be working for those generalised scenarios.

4.6 Summary

This chapter fills the gap that controller structure evolution has not been investigated into the fault tolerant area based on the implementation of a multi-objective network structure optimisation based NE approach, which is MOCGPANN in this work. The motivation of investigating MOCGPANN is due to the failure of CGPANN to achieve the robust fault tolerant control referred in chapter 3. So that is why MOCGPANN needs to be developed for fault tolerant control.

However there are two problems when developing the MOCGPANN algorithm. One is the problem occurred in the survival selection where the individuals with the same fitness but different inactive genes cannot be distinguished by the crowding distance measure. Although it is fine to pick a random individual due to the identical fitness, it is still worth developing a new crowding fill strategy driven by the significant benefit of preserving neutral genetic drift based on the genetic redundancy of inactive genes in each individual. The other is the convergence problem that a multi-objective optimisation algorithm normally relies on the convergence against a true Pareto optimal front obtained before the evolution. However the true Pareto optimal front may not always be acquired before the evolution is conducted. In this case, a new convergence criterion needs to be developed based on a performance measurement for a multi-objective optimisation algorithm.

Based on the investigation of population diversity and hyper-volume indicator, the MOCGPANN is developed with a modified crowding fill strategy rather than the original one from NSGA2 and a new convergence criterion is also developed based on the performance of hyper-volume indicator. However there is no significant difference between the modified and original crowding fill strategy including the comparison of two more different versions. A possible reason is due to the weight mutation which has already helped the evolutionary search and that is why the benefit of preserving population diversity and genetic drift is absent. Even though, preserving population diversity and genetic drift during the crowding fill strategy still achieved more stable performance than the original and a random preservation version. However it is still not clear whether a more stable performance could result in any further benefit for the evolutionary search at the moment. On the other hand, hyper-volume indicator demonstrated excellent performance to observe the convergence without acquiring the true Pareto optimal front in advance, which is quite useful for multi-objective optimisation algorithm convergence problem.

Apart from the evolution work, a more significant problem is to investigate how MOCGPANN could be used to evolve feasible controllers so as to achieve fault tolerant control, where CGPANN was failed to complete. Based on the generalisation experiment result, MOCGPANN demonstrates capability to obtain Pareto sets of controllers which achieved not only robust but also switched fault tolerant control, which fills the gap that controller structure evolution has not been investigated into fault tolerant area. Additionally, switched fault tolerant control outperforms robust fault tolerant control for the generalisation experiment as expected if fault diagnosis is already accomplished on the assumption for this work. However, not all the generalised scenarios are suited for the evolved controllers especially when the fault signal is changed or the beacon is moved to a new position for the online test.

The future work will be comprised of two parts including the evolution and generalisation experiments respectively. On the one hand, preserving population diversity or genetic drift has not presented significant benefit. Although they achieve more stable performance, this advantage is so weak compared to the evolutionary search. In this case, more work needs to be conducted to investigate whether preserving population diversity or genetic drift for the survival selection will result in any further benefit. On the other hand, the evolved controllers are not working very well for new fault signal and new beacon position scenarios. For this reason, these scenarios may need to be considered during the solution evaluation to obtain further optimised controllers so as to achieve not only a more effective fault tolerant control scheme but also a more effective robot phototaxis task.

The next chapter will investigate whether network structure optimisation still outperforms connection weight optimisation in the NE based multi-objective optimisation in terms of fault tolerant control scheme, which is driven by the conclusion in section 2.4.4.3.

Chapter 5 NSGA2 for ANN in fault tolerant control

5.1 Introduction

As can be seen from chapter 4, MOCGP demonstrates capabilities to achieve the robot sensor fault tolerant control based on NE in terms of network structure optimisation, which fills the research gap of controller structure optimisation based EA approach not investigated into fault tolerant control area. In terms of the evolution experiment, although preserving genetic drift and maximising population diversity have not demonstrated significant benefits to aid the evolutionary search for the crowding fill strategy during the survival selection, MOCGP still obtains feasible Pareto sets of controllers which would be promising to realise the fault tolerant control for robot phtotaxis task. According to the generalisation experiment, although just one type of crowding fill strategy is tested for its evolved Pareto sets of controllers, those controllers still demonstrate considerable performance in some of the generalised scenarios for both of robust and switched fault tolerant control based on the phototaxis task. In addition, switched fault tolerant control also outperforms robust fault tolerant control for the generalised experiment, which verifies the proposed benefit of switched fault tolerant control for this work mentioned in section 3.3.2.3.

In this sense, this chapter will investigate whether connection weight optimization based NE approach could also achieve the equivalent performance to obtain the feasible Pareto sets of controllers so as to realise the fault tolerant control for robot phototaxis task, which is a further investigation based on the conclusion in section 2.4.4.3 where network structure optimisation outperforms just connection weight optimisation in a series of basic NE benchmark experiments. In order to achieve the connection weight optimization for NE, NSGA2 could be the first choice for the multi-objective optimization for ANN's connection weights. As is mentioned in section 2.5.1, NSGA2 is a GA based multi-objective optimization algorithm and it has already demonstrated competitive performance in the parameter optimisation area. In addition, connection weight optimisation based fault tolerant control has been investigated extensively, so it is worth trying NSGA2 for the same experiment and see whether it can also achieve the fault tolerant control based on multiple objectives.

In this case, NSGA2 will be utilised to just train the ANN's connection weight for multiple objectives. However it needs to note that connection weight optimization and network structure optimization are actually two different approaches that weight optimization is just working to evolve the weight values but structure optimization will evolve the network structure and connection weight values at the same time in order to obtain a complete ANN. Therefore, to get a sensible comparison, the NSGA2 based ANN optimization will evolve the connection weight in different network structures firstly and then the ANN with the optimal structure will be used to do the comparison with MOCGP evolved ANN. The comparison will be conducted based on the evolution experiment and the evolved Pareto sets of controllers will be also investigated for the generalisation experiment.

5.2 Experiment setup

5.2.1 Evolution experiment

The aim of the evolution experiment is the same as that in section 4.4.1 where the controllers will be evolved in terms of two different objectives including the fault-free and faulty conditions so as to investigate whether the evolved Pareto set of controllers based on NSGA2 could achieve the robot sensor fault tolerant control. So the hypothesis of the NSGA2 based evolution experiment is that a feasible Pareto set of controllers could be obtained eventually based on the ANN's connection weight optimisation for fault-free and faulty conditions respectively.

The experiment setting with regard to the fault tolerant control based robot phototaxis task framework will be totally the same as that designed in section 3.2.1. The evolved controllers' performance will be compared to each other in terms of the same responses: hyper-volume and generation number. The ANN controllers will be evaluated in the same two objectives which are the fault-free condition and faulty condition where sensor 1 and 7 will be failed completely with zero signals as the sensor input reading to the controller. And the numbers of input and output nodes are still the same which are 8 and 2 respectively.

The only difference is the optimisation algorithm where NSGA2 will be utilised to just train the ANN's connection weight values instead of MOCGP for both the network structure and connection weight optimisation. Therefore the ANN's structure needs to be optimally adjusted in order to obtain the best evolved ANN's performance with NSGA2. On the other hand, NSGA2 has different parameters for the optimisation algorithm especially for the crossover operator which is not utilised for MOCGP, so it is also required that those algorithm parameter values should be optimised as well in order to maximise NSGA2's performance.

As is mentioned in section 2.4.4.3, CGPANN outperforms CNE for a series of benchmark experiments. In this case, a comparison will be conducted based on the best tuned ANN's structure for NSGA2 in order to investigate whether MOCGP will still outperform NSGA2 for multi-objective optimisation task in terms of NE based robot fault tolerant control, which will also demonstrate the benefit of network structure optimisation over just connection weight optimisation.

5.2.1.1 ANN parameters

As can be seen from [149], a single hidden layer ANN will be considered as a primary choice since just one hidden layer ANN will normally solve the majority of problems. However whether just one hidden layer is still suited for this experiment is unknown at the moment. Even though, it may be a time consuming work to try different hidden layers with different hidden neurons. For this reason, one hidden layer could be considered as the main structure for the ANN training with NSGA2 and the experiment of different hidden layer along with different hidden neuron comparison could be put in the future work. In this case, the only remaining problem is how many neurons are sufficient for this hidden layer. Although [149] mentions that the number of neurons in the hidden layer is basically between the number of input nodes and output nodes, it is just an empirically-derived conclusion. Therefore it is still worth setting more hidden nodes and see how those ANNs will perform for the fault tolerant control experiment. In a word, this evolution experiment utilises 5 different numbers of nodes in this single hidden layer, which are 6, 8, 10, 12 and 14 nodes respectively. The optimal number of hidden nodes will be utilised as the basic ANN's structure in terms of NSGA2 based connection weight optimisation. In addition, the connection weight range will be the same as that in section 4.5.1.1 in the interval [-10, 10] so as to achieve a sensible comparison with MOCGP. Moreover, since MOCGPANN performs better with feed forward than recurrent ANN for the controller evolution mentioned in section 4.5.1.1, NSGA2 will still implement feed forward ANN for the evolution and no recurrent connections will be considered in this work as well for a reasonable comparison.

Apart from those mentioned aspects, the last one that needs to be considered is the neuron transfer function. The MOCGPANN experiment conducted in section 4.4.1

utilises two different neuron transfer functions to create heterogeneous ANN since MOCGPANN is able to optimise not only network structure but also transfer functions. However NSGA2 is considered to be working well for the weight optimisation but it is currently unknown whether it is still working well for the transfer function optimisation. To simplify the comparison, the logistic sigmoid function, which is also referred to sigmoid function, is selected as the only neuron transfer function utilised for the ANN in the NSGA2 experiment. On the other hand, a new evolution experiment with MOCGPANN will be conducted along with NSGA2 based evolution experiment. This new MOCGPANN based evolution experiment is quite similar to that in section 4.4.1, the only difference is that the MOCGPANN disables the neuron transfer function optimisation. That is to say, MOCGPANN will only evolve a homogenous ANN with just one type of neuron transfer function, which is the same as this NSGA2 based evolution experiment. In this case, the sigmoid neuron transfer function will be implemented for MOCGP as the only function type for ANN as well as for NSGA2. For this reason, the comparison result will demonstrate the only difference between connection weight optimisation and network structure optimisation for ANN in terms of fault tolerance, where other aspects are totally the same.

5.2.1.2 NSGA2 parameters

In terms of the NSGA2 parameters, 20 individuals will constitute each population. The reason to set the same population size as MOCGP is due to the hyper-volume indicator result. As is mentioned in section 2.6.2, hyper-volume indicator is a famous approach to display the performance of a multi-objective optimisation algorithm. However the result of hyper-volume indicator is basically affected by the number of solutions in the first Pareto-optimal front. That is to say, more solutions in the first Pareto front will result in a higher indicator value and vice versa, which will disturb the performance comparison between NSGA2 and MOCGP. In this sense, NSGA2 still applies the same population size as MOCGP in each population, which guarantees that both of NSGA2 and MOCGP will utilise 20 individuals at most to demonstrate their performance. However, one problem is that whether 20 individuals are the best choice for both of NSGA2 and MOCGP is unknown at the moment. So it would be interesting to try different population size for NSGA2 and MOCGP at the same time and investigate the influence on their performance. However it is also a time consuming work, so 20 individuals could be still used as the population size and the investigation of different population size impact on their performance could be put in the future work.

The crossover used in this work is simulated binary crossover and the mutation is polynomial mutation, both of them are working for real number encoded weight values [150]. Simulated binary crossover was developed with respect to the one-point crossover for binary coded GA. And polynomial mutation is working in a way of probabilistic mutation where each variable will have a chance to be mutated to a new value. In addition, there are two more parameters only used in the simulated binary crossover and polynomial mutation, which is called distribution index. The distribution index will influence how far the children are from their parents. That is to say, a large distribution index will result in the children with higher probability to be closed to their parents and a small distribution index will lead to a lower probability [150]. It needs to note that MOCGP does not utilise crossover but NSGA2 does. The reason may be that CGP actually does not benefit from crossover [29] but NSGA2 relies on it no matter for real number or binary number encoding [92]. In this case, it is still a fair comparison even though NSGA2 utilises crossover and MOCGP does not.

In terms of the parameter values, the crossover probability is set 0.9, the mutation probability is set 0.02, and the distribution index is set 20 for both the crossover and mutation respectively. These values will be considered as the baseline parameter values. So when the optimal ANN's structure is found, those baseline parameter values will be adjusted based on the Parameter Robustness technique [141] developed in Spartan in order to achieve the sensitivity analysis along with the cumulative mean approach [142] for the number of experiment runs determination. Both of Parameter Robustness technique and cumulative mean approach are the same as that utilised in section 4.4.1.2. It also needs to note that the parameters for convergence criteria will be the same from the conclusion in section 4.5.1.1 where Nconv is 20 and Rconv is 1.01. Although these two parameters could be also tuned to obtain the optimal values, different convergence parameters will result in different responses for both the hyper-volume and generation number. In this case, it is more sensible to remain the calibration values for convergence parameters in order to conduct a fair comparison between NSGA2 and MOCGPANN.

5.2.1.3 Summary of the difference between NSGA2 and MOCGP parameters

As is referred in section 2.4.4.3, network structure optimisation outperforms connection weight optimisation based on the comparison work between CNE and CGPANN. In this case, it is interesting to explore whether network structure optimisation still outperforms connection weight optimisation in terms of multi-objective optimisation with NSGA2 and MOCGP respectively for robot fault tolerant control. Table 5.1 presents a conclusion about all the different aspects including the ANN parameters and the optimisation algorithm parameters mentioned above between NSGA2 and MOCGP for the ANN controller evolution experiment.

Table 5.1: Different aspects between NSGA2 and MOCGP for ANN evolution

	NSGA2	MOCGP
ANN parameters	Hidden layer number	Node number
	Hidden node number	Arity number
	Weight range	Weight range
	Feed forward/Recurrent	Feed forward/Recurrent
	Neuron transfer function	Neuron transfer function
Algorithm parameters	Population size	Population size
	Mutation probability	Mutation probability
	Crossover probability	
	Distribution index for crossover	
	Distribution index for mutation	

In Table 5.1, all the different aspects have been discussed. As is mentioned in section 5.2.1.1 and 5.2.1.2, both of ANN and algorithm parameters need to be adjusted in order to maximise the performance for these two algorithms. However, The ANN parameters are investigated to maintain the consistency between NSGA2 and MOCGP in order to obtain a sensible comparison result between the connection weight optimisation and network structure optimisation for fault tolerant control. In this case, the Hidden node number is adjusted for NSGA2 (Hidden layer number could be set 1 at the moment) and the Node number with the Arity number are adjusted for MOCGP. Other parameters will be remained consistent including the same Weight range, the same Feed forward ANN type and the same Neuron transfer function implemented to create homogeneous ANN. The optimisation algorithm parameters are tuned for each algorithm respectively since they have no influence on the ANN composition, which just benefit to the evolutionary search in order to obtain the best responses. Except for the parameter difference in ANN and the algorithm itself, other aspects will be totally the same for the phototaxis task framework designed in section 4.4.1. So the aim of this comparison is to investigate whether the network structure optimisation will outperform the connection weight optimisation in terms of the multi-objective robot fault tolerant control.

5.2.2 Generalisation experiment

The generalisation experiment setup is actually totally the same as that designed in section 4.4.2. Since the generalisation experiment is just a test for the evolved controllers, so it does not matter how the controller is evolved, for example structure optimisation or weight optimisation, and that's why the generalisation experiment setup can be remained fixed as that in section 4.4.2. The aim of the generalisation experiment is to test whether the evolved Pareto sets of controllers by NSGA2 instead of MOCGP could also achieve both of switched and robust fault tolerant control in more generalised scenarios. Basically, the fault should be injected during the robot phototaxis task as a primary principle for the generalisation experiment, so the controllers should be switched between each other for the switched fault tolerant control when the fault is occurred. And for the robust fault tolerant control, only one controller is loaded all the way through the phtotaxis task no matter when the fault is occurred. So the hypothesis of the generalisation experiment is quite similar with that in section 4.4.2 where the obtained feasible Pareto sets of controllers by NSGA2 could make robot achieve both of switched and robust fault tolerant control based on the phototaxis task in more generalised scenarios as well.

The generalised scenarios are also the same as that designed in section 4.4.2 where the obtained controllers will firstly be tested in the same basic scenario as the evolution experiment but the fault will be injected during the task including 300 and 700 ticks respectively. And then if the evolved controllers are working well in this basic scenario, three more generalised scenarios will be utilised to test the evolved controllers including the different fault signal, different robot starting position with orientations and different beacon position. The new faulty sensor signals will be set 0.5 instead of 0. Ten new robot starting positions and orientations will be utilised rather than the original robot starting conditions. Finally the beacon will also be placed in position (2, 2) instead of original position (0, 0) and the experiment time limit will be set 3000 ticks since 1200 ticks is meaningless when the new beacon position is varied to different robot starting points and the success rate will be utilised as the new controller assessment criteria rather than the time to the beacon. All of the mentioned above generalisation experiment setup is totally the same as that in section 4.4.2 and five different Pareto sets of controllers obtained by NSGA2 will be utilised for the generalisation experiment. In terms of switched fault tolerant control, two controllers are selected working well for each objective respectively. And in terms of the robust fault tolerant control, only one controller is required with a similar performance for both of the two objectives. The result of switched and robust fault tolerant control will be presented and discussed respectively in next section 5.3.2.

5.3 Result and discussion

5.3.1 Evolution experiment

As is mentioned in section 5.2.1, the obtained solutions should be investigated in two steps: different ANN's structure and different NSGA2 parameter values. Therefore, only the best tuned ANN's structure and NSGA2 parameter values could be utilised to maximise the evolved controllers' performance so as to conduct the final comparison with the solutions found with MOCGPANN. In this case, the ANN's structure adjustment and NSGA2 parameters' tuning will be discussed as following.

5.3.1.1 Number of hidden nodes selection

In terms of the number of hidden nodes, five different options were selected including 6, 8, 10, 12 and 14 hidden neurons for this hidden layer in terms of hyper-volume and generation number responses in order to find out how many are sufficient. This investigation was conducted based on the U-test and A-test as well in order to find out the significant difference between these options. The result shows that 12 and 14 hidden neurons obtained the best and quite similar performance, which indicates that 12 hidden neurons may be already sufficient for this hidden layer. For this reason, 12 nodes could be selected for the ANN's hidden layer and this type of structure will be utilised throughout this chapter for the evolution fault tolerant experiment based on connection weight optimization with NSGA2. The details of this hidden neuron investigation can be referred to Appendix C.1.

5.3.1.2 Sensitivity analysis for NSGA2 parameters

Apart from the ANN's structure optimisation, the NSGA2 parameters also need to be adjusted to their optimal values in order to obtain the best responses for the evolved controllers. As is mentioned in section 5.2.1.2, there are four parameters for NSGA2 that needs to be tuned including the crossover probability (PCrossover), mutation probability (PMutation), distribution index for crossover (DICrossover) and mutation (DIMutation) respectively. The baseline values for them are listed respectively in Table 5.2 which will be utilised for the Parameter Robustness technique for the sensitivity analysis. All the comparison results will be displayed in boxplot and their corresponding A-test scores will also be presented in graphs created by Parameter Robustness technique. The details of the sensitivity analysis can be referred to Appendix C.2.

Table 5.2: Baseline values for NSGA2 parameters

NSGA2 parameters	PCrossover	PMutation	DICrossover	DIMutation
Baseline values	0.9	0.02	20	20

Now all the parameters have been calibrated to their optimal values which are listed in Table 5.3. The next step is to conduct the final comparison between the responses of baseline parameter values against the calibration parameter values for NSGA2 in order to investigate whether all the NSGA2 parameters in their calibration values will outperform their baseline values.

Table 5.3: Calibration values for NSGA2 parameters

NSGA2 parameters	PCrossover	PMutation	DICrossover	DIMutation
Calibration values	0.9	0.05	20	5

Figure 5.1 shows the hyper-volume comparison between the NSGA2 parameter baseline values and calibration values and Figure 5.2 shows the generation number comparison results for them.

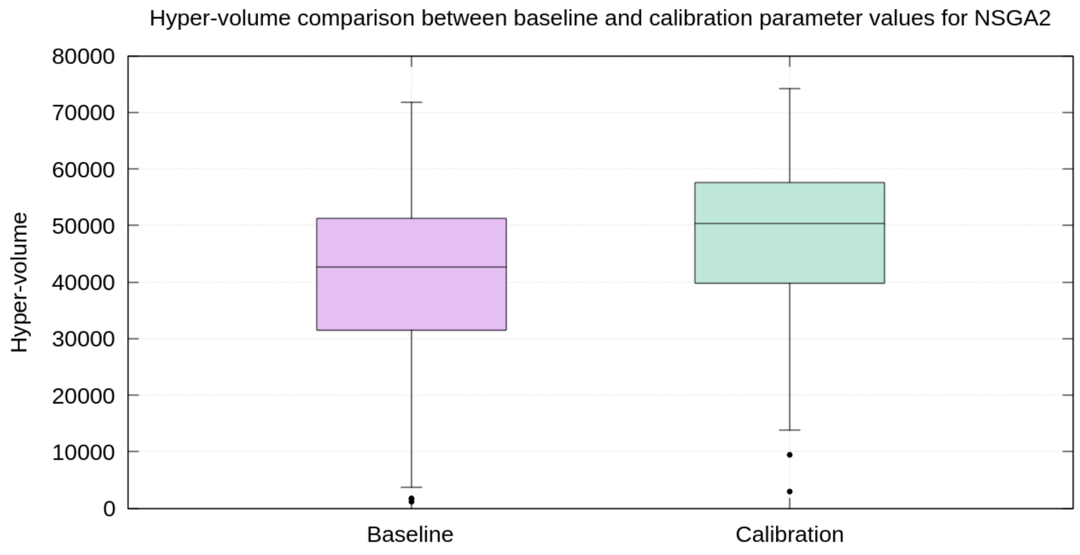

Figure 5.1: Hyper-volume comparison between the NSGA2 parameter baseline values and calibration values

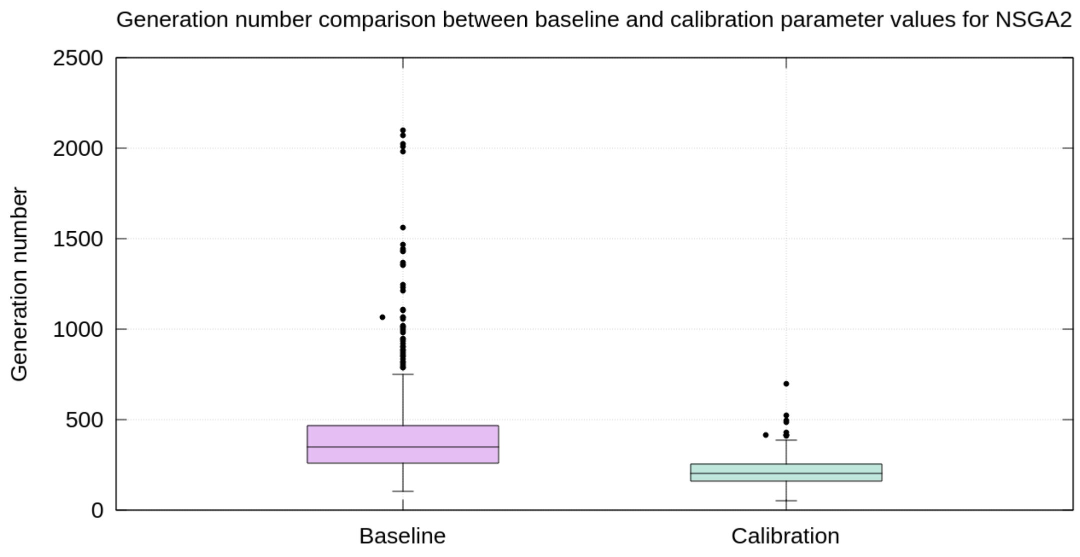

Figure 5.2: Generation number comparison between the NSGA2 parameter baseline values and calibration values

Table 5.4: U-test scores for the comparison between NSGA2 parameter baseline values and calibration values

	Baseline values	Calibration values
Hyper-volume	1	< 0.00001
Generation number	1	< 0.00001

Table 5.5: A-test scores for the comparison between NSGA2 parameter baseline values and calibration values

	Baseline values	Calibration values
Hyper-volume	0.5	0.34126
Generation number	0.5	0.846786

As can be seen in Figure 5.1 and Figure 5.2, NSGA2 calibration parameter values outperforms baseline parameter values with a higher hyper-volume and a lower generation number. According to Table 5.4, there is a significant difference between these two approaches where the p values for both of these two comparisons are less than 0.0001%. Finally Table 5.5 also demonstrates the same result that the calibration values achieve a higher hyper-volume with a medium effect size and it also achieves a lower generation number with a large effect size. In conclusion, the NSGA2 parameters in their calibration values outperform their baseline values and those calibration values will be utilised for the comparison between NSGA2 and MOCGP in the next section 5.3.1.3 in terms of multi-objective robot fault tolerant control with ANN optimisation.

5.3.1.3 Comparison between NSGA2 and MOCGP

After the ANN and NSGA2 parameters have been adjusted optimally, the NSGA2 based evolution experiment results will be compared with MOCGP based evolution experiment results. This comparison will demonstrate whether network structure optimisation will be able to produce a better performance than just connection weight optimisation for NE based multi-objective optimisation in terms of robot fault tolerant control.

There are two points that needs to be mentioned here. One is that NSGA2 and MOCGP are actually working in two different ways where NSGA2 is working in parameter level and MOCGP is working in both of parameter and structure levels. Due to the different evolutionary search areas, it is meaningless to compare the generation or evaluation number response for these two approaches. However the hyper-volume response is still suited to measure their performance for evolved solutions as long as the solutions are feasible to solve the task, which is the robot phototaxis task in the face of sensor failures, no matter which evolutionary search level the algorithm is working. Although NSGA2 is working in parameter level, the ANN's structure is already optimised in section 5.3.1.1. In this case, the NSGA2 evolved solutions with optimal parameter values could be considered with optimal performance.

The other is that the MOCGP parameters utilised for this comparison is slightly different from the one in section 4.5.1.1. As is mentioned in section 5.2.1.1, NSGA2 implements sigmoid neuron transfer function to create the homogeneous ANN for connection weight optimisation. So in order to maintain the consistency for a sensible comparison, MOCGP also utilises sigmoid neuron transfer function as the only function type to create the homogeneous ANN for network structure optimisation rather than two different function types implemented for heterogeneous ANN in section 4.5.1.2. Other aspects including the weight range and the feed forward network type are both the same between these two approaches. Only in this way, the comparison result will demonstrate the difference between network structure and connection weight optimisation for the evolution experiment.

Figure 5.3 shows the final comparison between NSGA2 and MOCGP in terms of hyper-volume response. And Table 5.6 lists the corresponding U-test and A-test scores.

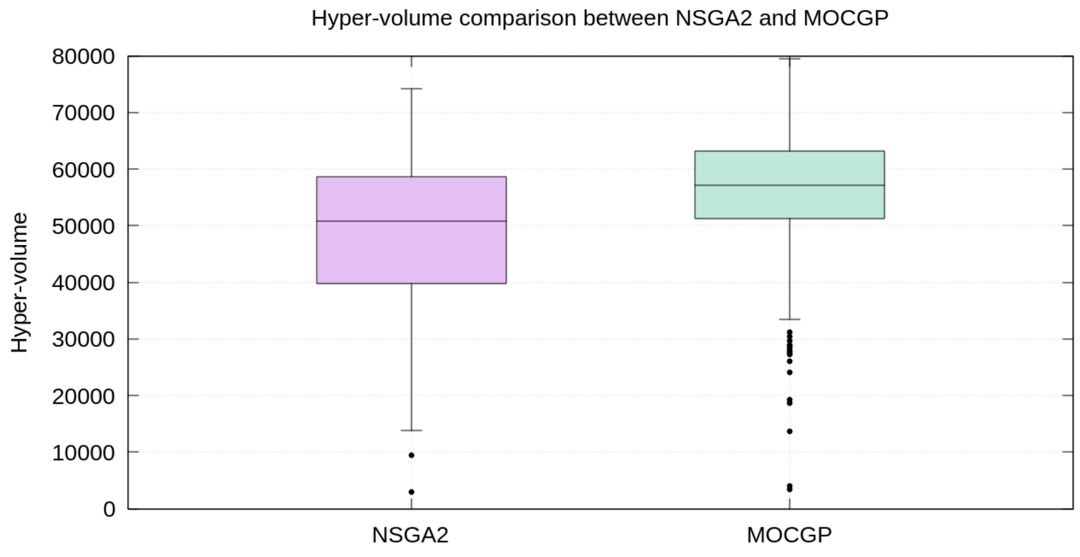

Figure 5.3: Hyper-volume comparison between NSGA2 and MOCGP

Table 5.6: U-test and A-test scores for hyper-volume comparison between NSGA2 and MOCGP

	NSGA2	MOCGP
Hyper-volume with U-test	1	< 0.00001
Hyper-volume with A-test	0.5	0.337994

As can be seen from Figure 5.3, MOCGP outperforms NSGA2 in terms of hyper-volume response. According to Table 5.6, there is a significant difference between these two approaches where the p value of U-test is less than 0.00001%. Moreover, the A-test score also indicates a higher hyper-volume response for MOCGP where its score of 0.337994 belongs to the medium effect size interval of [0.29, 0.36]. In a word, MOCGP produces a better performance than NSGA2 in terms of hyper-volume for NE based multi-objective robot fault tolerant control, which also demonstrates that network structure optimisation performs better than just connection weight optimisation for this experiment.

However as is referred in section 2.4.4.3, the comparison between NSGA2 and MOCGP is actually not quite convincing to display the performance difference between connection weight optimisation and network structure optimisation. One possible reason is that the computational effort of these two algorithms is quite different due to their different search space. That is to say, NSGA2 optimises the connection weight values in a pre-designed network structure but MOCGP has to optimise both of connection weight and network structure at the same time, which is just the same as what the comparison work is conducted in this section. In this case, it may be considered as an unfair comparison to show the performance difference between connection weight optimisation and network structure optimisation due to their different search space. However, this comparison result still demonstrates that NSGA2 is not able to produce an equivalent performance with MOCGP even if the ANN's structure has been well adjusted in advance.

Generally speaking, this comparison is still able to present the performance difference between these two approaches where MOCGP outperforms NSGA2 in terms of the hyper-volume measurement. A possible explanation for this comparison result is that MOCGP could obtain an efficient ANN's structure which will not be considered by human designer. That is to say, the ANN's structure for NSGA2 is normally designed in a usual way, so it may not be available for NSGA2 to utilise those structures which are obtained by MOCGP. And that may be a main reason why MOCGP will outperform NSGA2 for the multi-objective robot fault tolerant control experiment. However as mentioned in section 5.2.1.1, the ANN structure utilised for NSGA2 just contains a single hidden layer. Although a single hidden layer ANN is able to solve a majority of problems, it is unknown that whether more than one hidden layer will be helpful to increase the ANN performance for this experiment. On the other hand, the ANN's structure evolved by MOCGP is actually not a usual ANN's structure which may contain lots of hidden layers with different hidden neurons for each layer. So it is still not clear whether the better performance of MOCGP is due to the unusual structure or due to more than one hidden layers evolved in the ANN that may contribute to the ANN's performance for this experiment. Apart from hidden layer problem, population size may also influence the evolution result. As is mentioned in section 5.2.1.2, NSGA2 and MOCGP both utilise 20 individuals as the population size. Although same population size could maintain the measure consistency for hyper-volume indicator, whether 20 individuals are the best choice to demonstrate the performance of either NSGA2 or MOCGP is still unknown at the moment. For these reasons, it is still worth a further investigation on the evolved ANN performance with more than one hidden layer utilised for NSGA2 to train the ANN's weights and with different population sizes for both of NSGA2 and MOCGP, which will be considered as future works.

Even so, the comparison result still demonstrates that MOCGP removes the requirement to obtain a suitable network structure in advance even if the obtained ANN's structure is unusual, where NSGA2 has to consider the network structure manually by human designer for the given task when training ANN's connection weight. Actually, those mentioned benefits for MOCGP is actually the same as CGP for NE which is mentioned in section 2.4.4.3 and this comparison result still indicates that MOCGP is benefit from the network structure optimisation for NE.

In conclusion, the comparison result demonstrates that network structure optimisation will be able to produce better performance than just connection weight optimisation for NE based multi-objective robot fault tolerant control task, at least for MOCGP and NSGA2 respectively. The next section 5.3.2 will investigate how the NSGA2 evolved solutions could impact the ANN controllers' performance in generalised scenarios.

5.3.2 Generalisation experiment

As mentioned in section 5.2.2, the whole generalisation experiment is similar to section 4.5.2, which was conducted with robust and switched fault tolerant control respectively based on five Pareto sets of controllers but evolved by NSGA2 instead of MOCGP in terms of four generalised scenarios. The success rate would also be used to assess the evolved controllers' performance based on 10 different trials. The data of these five Pareto sets' generalisation experiment result can be referred to the Appendix A.2.

5.3.2.1 Robust fault tolerant control

In terms of the robust fault tolerant control, there is just one controller that is loaded for the robot, so there is no need to consider the controller switch. However the controller has to perform relatively well for each objective, so a degraded performance could be accepted compared to the switched fault tolerant control.

Table 5.7: Success rate for five obtained Pareto sets by NSGA2 in terms of robust fault tolerant control

	Initial scenario		Fault signal		Robot condition		Beacon position	
	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700
Pareto 1	1	1	0	1	1	1	1	1
Pareto 2	1	1	0	0	1	1	0.6	0.6
Pareto 3	1	1	0	0.5	1	1	0.5	0.5

Pareto 4	1	1	0	0	1	1	0.5	0.5
Pareto 5	1	1	0	1	1	1	1	1

Table 5.7 lists the success rate for five obtained Pareto sets by NSGA2 in terms of robust fault tolerant control. As can be seen in this table, all the five evolved Pareto sets are working well for basic initial scenario and new robot starting conditions as well with 100% success rate for both of 300 and 700 ticks. However in terms of new fault signal and new beacon position scenarios, none of the five Pareto sets achieves 100% success rate no matter the fault occurs at 300 or 700 ticks with better or worse performance. Moreover, it is serious that all the five Pareto sets perform 0% success rate for the new fault signal scenario when the fault occurs at 300 ticks, which means all the evolved Pareto sets of controllers are actually failed to achieve the robust fault tolerant control.

According to the result, the controllers are working very well for basic scenario and new robot starting condition scenario. That is to say, no matter when the fault occurs or where the robot initial condition is, the evolved controllers by NSGA2 are capable to make robot continue performing the phototaxis for robust fault tolerant control, as long as the distance between robot initial position and beacon position is fixed as that in the evolution experiment. However the evolved controllers are not working well for new fault signal scenario and new beacon position scenario. What's more, all of the five Pareto sets perform 0% success rate for new fault signal scenario when fault occurs at 300 ticks for robust fault tolerant control. That is to say, the evolved controllers are completely failed to make the robot continue performing the phototaxis when fault occurs at an early stage, for example 300 ticks in this experiment. In conclusion, the evolved controllers are suited to the initial and new robot condition scenarios, but not suited to the new fault signal and new beacon position scenarios. In this case, it is also required that more controllers need to be evolved as future works to deal with those two unsuited scenarios so as to achieve the robust fault tolerant control, which is the same as the conclusion in section 4.5.2.1.

5.3.2.2 Switched fault tolerant control

In terms of switched fault tolerant control, two controllers from each Pareto set evolved by NSGA2 will be selected based on their performance for each objective. Basically, each controller should achieve the best performance for its corresponding objective among the controllers in this Pareto set. So it does not matter what the performance is for the opposite objective of the selected controllers since they can be switched to each other when the fault is occurred.

Table 5.8: Success rate for five obtained Pareto sets by NSGA2 in terms of switched fault tolerant control

	Initial scenario		Fault signal		Robot condition		Beacon position	
	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700
Pareto 1	1	1	0	1	1	1	1	1
Pareto 2	1	1	0	0.1	1	1	0.6	0.6
Pareto 3	1	1	0	0.9	1	1	0.5	0.5
Pareto 4	1	1	0	0	1	1	0.5	0.5
Pareto 5	1	1	1	1	1	1	1	1

Table 5.8 lists the success rate for five obtained Pareto sets by NSGA2 in terms of switched fault tolerant control. As can be seen from the table, all the five evolved Pareto sets of controllers by NSGA2 are capable of performing the phototaxis task with totally 100% success rate for the basic initial scenario and the new robot condition scenario when the fault is injected at 300 and 700 ticks respectively during the task. However the performance for new fault signal scenario is declined dramatically where only the 5th Pareto set achieves 100% success rate for both of 300 and 700 ticks. In addition, all the other four Pareto sets obtain 0% success rate for 300 ticks and more or less success rate for 700 ticks. Finally in terms of new beacon position scenario, only the 1st and 5th Pareto sets realise 100% success rate for both of 300 and 700 ticks and the other Pareto sets reach more or less success rate respectively.

According to the result, it can be seen that the evolved Pareto sets of controllers by NSGA2 realise a similar performance of switched fault tolerant control compared to robust fault tolerant control where the evolved Pareto sets of controllers are working very well not only for the basic scenario of different fault occurrence time but also for the new robot starting condition scenario. However in terms of new fault signal scenario and new beacon position scenario, only the 5th Pareto set realises 100% success rate for both of these two scenarios and all the other Pareto sets achieve different more or less success rate. This result demonstrates that the evolved Pareto sets of controllers by NSGA2 are working well to achieve the switched fault tolerant control no matter what time the fault occurs and where the robot initial condition is as long as the distance from the beacon is fixed as the evolution experiment. However, if the fault signal is altered to another value instead of the one from the evolution experiment, the evolved controllers may not be working well for the switched fault tolerant control. Moreover, if the beacon position is changed with varied distance to the robot initial position, the evolved controllers are also not working very well. In this case, if different fault signal or different beacon position is required for the switched fault tolerant control, more controllers may need to be evolved based on these new conditions, which is the same as the conclusion in section 4.5.2.2 for future works.

5.3.2.3 Comparison between two fault tolerant control schemes

This section is similar to section 4.5.2.3 where robust and switched fault tolerant control performance will be compared to each other based on the obtained generalisation experiment result of success rate in terms of 10 different trials. The only difference is that this comparison will be conducted based on the NSGA2 evolved controllers instead of MOCGP in section 4.5.2.3, but the aim is still the same to investigate which one performed better for this work, the robust or the switched fault tolerant control.

Table 5.9 lists the comparison result between robust and switched fault tolerant control based on the obtained generalisation rate. This comparison also utilised A-test to estimate the difference of switched fault tolerant control compared to the robust fault tolerant control, which is the same as that in section 4.5.2.3.

Table 5.9: Comparison between the success rate of robust and switched fault tolerant control based on the controllers evolved by NSGA2

Scenario	Initial scenario	Fault signal	Robot condition	Beacon position
A-test score	0.5	0.395062	0.5	0.5

As can be seen from Table 5.9, only the A-test score for new fault signal scenario is different from the other three scenarios. In terms of the new fault signal scenario, the A-test score is 0.395062 which lies in the small effect size interval [0.36, 0.44] and this result indicates that switched fault tolerant control outperforms robust fault tolerant control in the new fault signal scenario, although it is a small effect size. With respect to the other three scenarios, all of them have a same A-test score 0.5, which means both of robust and switched fault tolerant control have the identical performance as listed in the generalisation experiment results.

As can be seen from the comparison result, although switched fault tolerant control only produced a better performance in the new fault signal scenario than the robust fault tolerant control, neither of these two approaches achieved a total 100% success rate among 10 trials as seen from the generalisation result in section 5.3.2.1 and 5.3.2.2. Additionally, although the performance for the new beacon position scenario is the same between robust and switched fault tolerant control, neither of them achieved a 100% success rate for the 10 trials and that means both of robust and switched fault tolerant control are not working well in spite of an identical performance.

In conclusion, the comparison result of the NSGA2 evolved controllers is similar to that in section 4.5.2.3 where switched fault tolerant control also outperforms robust fault tolerant but just in the new fault signal scenario. In terms of the new beacon position scenario, neither of robust and switched fault tolerant control achieved a better performance, which may indicate that NSGA2 is not as effective as MOCGP to obtain better performance controllers as is displayed in the evolution comparison result in section 5.3.1.3.

5.4 Summary

This chapter is actually a comparison work with chapter 4 to investigate the difference between connection weight optimisation and network structure optimisation for NE based fault tolerant control. This comparison is driven by the conclusion in section 2.4.4.3 that network structure optimisation outperforms connection weight optimisation on a series of NE based single objective optimisation benchmarks. In this case, this work utilises NSGA2 instead of MOCGP as the connection weight optimisation approach to investigate whether MOCGP still outperforms NSGA2 for the NE based multi-objective optimisation in terms of fault tolerant control with robot phototaxis task based on the same evolution and generalisation experiment framework as designed in chapter 4.

In terms of the evolution experiment, NSGA2 is able to obtain feasible Pareto sets of controllers for both of fault-free and faulty objectives. As a comparison, MOCGP is also utilised to conduct the same evolution experiment with the same feed forward network type, same weight range and even same neuron transfer function implemented for the homogeneous ANN optimisation as the NSGA2 based evolution experiment. However NSGA2's performance is worse than that of MOCGP in terms of hyper-volume. A possible reason is that MOCGP could obtain a network structure which is not considered by NSGA2 but with more effective performance. However the ANN's structure utilised by NSGA2 just contains one hidden layer but MOCGP could evolve ANN with lots of hidden layers with different hidden neurons. So it is not quite clear whether the performance difference is due to the unusual structure of the evolved ANN by MOCGP or due to the number of hidden layers that may contribute to the ANN's performance. On the other hand, both of NSGA2 and MOCGP utilise 20 individuals as the population size. Although same population size could maintain the consistency of hyper-volume measurement, it is also unknown whether 20 individuals are the most suited choice for each of these two algorithms. In this case, further experiment needs to be conducted to investigate the impact of more than one hidden layer on the ANN's performance for NSGA2 and the impact of different population size for both of NSGA2 and MOCGP, which could be considered as future works. In conclusion, the result in this chapter demonstrates that network structure optimisation still outperforms connection weight optimisation for NE based multi-objective fault tolerant control for robot phototaxis task, which further verifies the benefit of network structure optimisation over just connection weight optimisation.

In terms of the generalisation experiment, the evolved controllers by NSGA2 are capable to achieve both of switched and robust fault tolerant control in generalised scenarios no matter when the fault occurs or the robot initial condition is. However the controllers evolved by NSGA2 are not working very well when new fault signal is injected instead of the original one and when the beacon is moved to a new position in terms of both the switched and robust fault tolerant control. Nevertheless, the comparison result also demonstrates that switched fault tolerant control outperforms robust fault tolerant control for this work despite that the switched fault tolerant control only produced a better performance in the new fault signal scenario. On the other hand, this generalisation experiment result obtained by NSGA2 still demonstrates that a multi-objective optimisation algorithm is essential to achieve the robust fault tolerant control where a single objective optimisation algorithm is not able to achieve such as CGP mentioned in the conclusion of chapter 3. Future work will be the same as section 4.6 that it is required to investigate how the controllers could be evolved to deal with varied fault signals and different beacon positions not only for a more effective fault tolerant control system but also for a more effective robot phototaxis task.

Due to the benefit of network structure optimisation over just connection weight optimisation for NE based multi-objective fault tolerant control, the next chapter will design an extension experiment with more difficult scenarios for fault types and investigate how MOCGPANN will be performing to obtain feasible Pareto sets of controllers for this extension experiment in order to achieve robust and switched fault tolerant control.

Chapter 6 MOCGPANN in extension fault tolerant control

6.1 Introduction

As can be seen from section 5.3.1.3, MOCGP demonstrates competitive performance compared to NSGA2 for NE in terms of the multi-objective fault tolerant control evolution experiment based on the robot phototaxis task. The result in section 5.3.1.3 also indicates that MOCGP based ANN network structure optimization outperforms NSGA2 based ANN connection weight optimization in terms of the controllers' performance based on hyper-volume measurement. In addition, this comparison could be considered as a comprehensive work since the ANN's structure is adjusted firstly in order to maximise its performance. And then, the connection weights are evolved based on the pre-optimised network structure, which not only guarantees the network structure to be optimised, but also ensures optimised connection weight values. Although the decision of hidden node number is not a work achieved by NSGA2, this comparison is still a sensible work which fully demonstrates the performance difference between the network structure optimization and connection weight optimization at least for MOCGP with NSGA2 in terms of multi-objective fault tolerant control.

Due to the capability of MOCGPANN for evolving effective fault tolerant controllers, this chapter will conduct an extension fault tolerant control experiment based on MOCGPANN and investigate how it will be performing for this more difficult fault tolerant control scheme in order to find out whether MOCGPANN could still evolve feasible controllers to achieve both of robust and switched fault tolerant control.

6.2 Experiment setup

The aim of this extension experiment is to investigate how MOCGPANN will be performing to evolve fault tolerant controllers for a more difficult scenario with more sensor failures. The obtained controllers will be tested for robust and switched fault tolerant control respectively but there will be more faulty sensors in the generalised scenarios as well. Therefore, the hypothesis is that MOCGPANN could be also effective to evolve Pareto sets of controllers so as to achieve the extension experiment for both of robust and switched fault tolerant control.

This work is definitely more difficult for the robot to still achieve phototaxis task with just right side sensors to be working. However, it is still interesting to investigate whether MOCGPANN could still obtain a feasible Pareto set of controllers on this extension evolution experiment and whether these evolved controllers will be working in generalised scenarios.

6.2.1 Evolution experiment

The evolution experiment setup is quite similar to section 4.4.1 based on the same robot phototaxis task designed in section 3.2.1. The only difference is that the left 4 light sensors are failed together with zero reading signals as the input to the controller. In other words, the robot would become totally blind to perceive light for the left side when fault occurs, which is more difficult than the original 2 sensor failure scenario designed in section 3.2.2 so as to obtain feasible fault tolerant controllers. In this case, one objective would remain the same for the faultless condition, but the other objective would become a 4 sensor failures condition. The controllers would be evolved for these two objectives respectively in order to create the Pareto optimal set, if the set could be obtained. Except for the objective modification, other aspects would still be the same as the evolution experiment in section 4.4.1 including the calibrated MOCGPANN parameter and convergence measurement parameter values from section 4.6.1.2. Finally, four different crowding fill strategies would also be utilised respectively to assess the performance of MOCGPANN and the comparison among them would be conducted based on the hyper-volume and generation numbers, which still utilised the same procedure mentioned in section 4.5.1.2.

6.2.2 Generalisation experiment

The generalisation experiment is totally the same as that designed in section 4.4.2 except for 4 sensor failures instead of 2 sensor failures. In this case, each generalised scenario would be tested based on 4 sensor failures during the phototaxis task. Additionally, in terms of the new fault signal scenario, each of the left 4 sensors will produce a 0.5 reading signal into the controller instead of 0 reading signal from the evolution experiment. Apart from that, each obtained Pareto set of controllers would be tested with 10 different trials and their success rate was also be used to assess the controller performance in terms of both the robust and switched fault tolerant control in terms of the generalisation tests.

6.3 Result and discussion

6.3.1 Evolution experiment

Figure 6.1 shows the hyper-volume comparison for different crowding fill strategies in terms of the extension evolution experiment. Figure 6.2 shows the generation number comparison for that. Table 6.1 lists the U-test scores for these two comparisons and Table 6.2 lists the corresponding A-test scores.

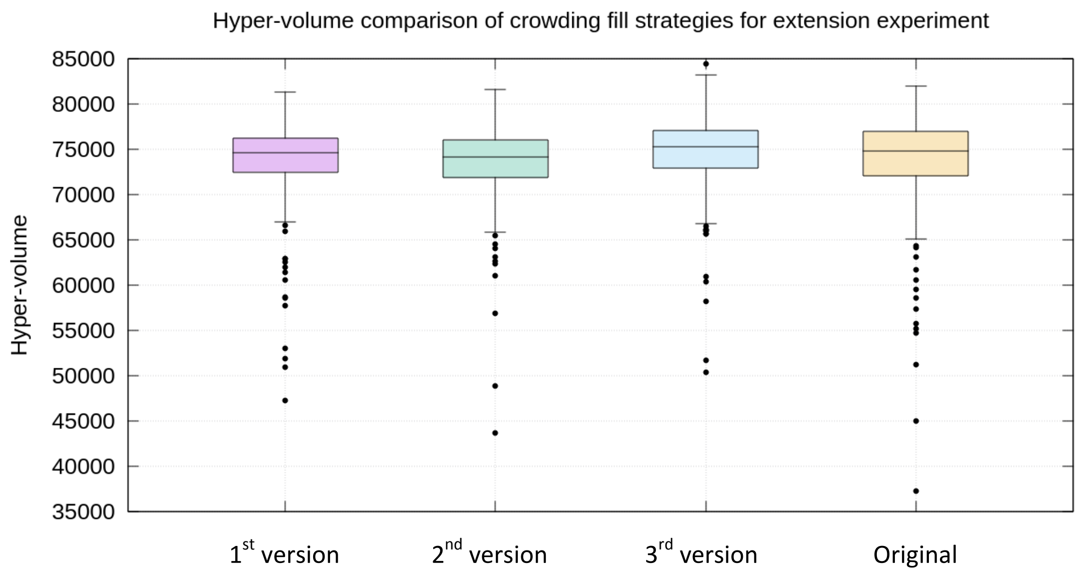

Figure 6.1: Hyper-volume comparison for different crowding fill strategies in terms of extension experiment

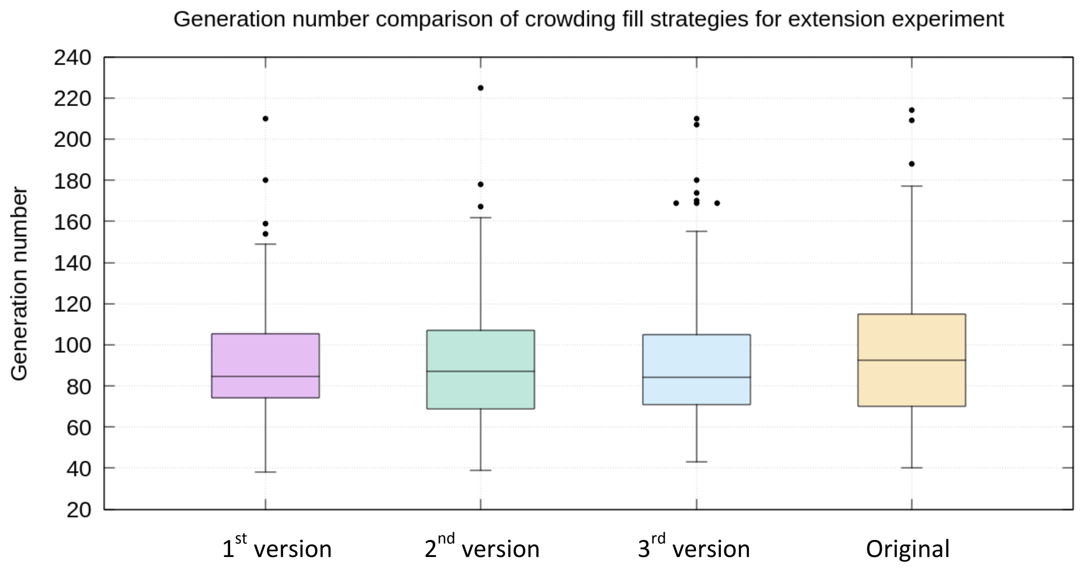

Figure 6.2: Generation number comparison for different crowding fill strategies in terms of extension experiment

Table 6.1: U-test scores for hyper-volume and generation number in terms of extension experiment

	1st version	2nd version	3rd version	Original
HV	1	0.29834	0.0466	0.62414
Gen	1	0.99202	0.72786	0.28462

Table 6.2: A-test scores for hyper-volume and generation number in terms of extension experiment

	1st version	2nd version	3rd version	Original
HV	0.5	0.528356	0.442516	0.485608
Gen	0.5	0.501213	0.511243	0.467661

As can be seen from Table 6.1, the third version is the only one that achieves significant difference compared to the first version in terms of hyper-volume measurement, whose p value is below 0.05. Moreover, as is listed in Table 6.2, the A-test score of the third version in terms of hyper-volume is also below 0.5, which means the hyper-volume is higher than that of the first version where the median value is slightly higher than the first one as well according to Figure 6.1. In addition, the A-test score of the third version in terms of the generation number also achieves a slightly larger value than 0.5, which means the generation number of the third version is slightly less than the first one. In this case, it seems that the third version of crowding fill strategy is the best choice for the extension evolution experiment since it spends fewer generations to obtain better hyper-volume response. However to be more specific, the A-test score of the third version for hyper-volume is still quite a small effect size compared to the first one, where its A-test score is not even located in the small effect range between 0.44 and 0.36. Apart from the third version, the second version spends similar generations to obtain lower hyper-volume response than the first one; the fourth version achieves a higher hyper-volume but the generation number is also larger than that of the first one. However neither second nor fourth version achieves the significant difference compared to the first one, whose U-test scores are both larger than 0.05 in terms of both the hyper-volume and generation number. In conclusion, the third version seems to be the best choice for crowding fill strategy utilised for MOCGPANN for the extension evolution experiment. Although its A-test score for hyper-volume is still a quite small effect compared to the first one, this version of crowding fill strategy still demonstrates the competitive performance among each other for the extension evolution experiment. In this way, the controllers evolved from this version could be a first choice to conduct the generalisation experiment, whose result will be discussed in the next section 6.3.2.

It needs to note that the cumulative mean result is listed in Table 6.3 for the extension evolution experiment. In this table, the first version requires the least number of runs to sufficiently present its performance and the last one obtains the most number of runs. As is mentioned in section 4.5.1.2, the first and second versions achieve much less experiment runs than the other two to obtain a sufficient performance. And in terms of this extension evolution experiment, although second version ranks the third for the cumulative mean result, the fourth version is the last choice for crowding fill strategy in terms of the require number of experiment runs. In a word, preserving population diversity based crowding fill strategy is the second most stable version for the basic evolution experiment and the most stable version for the extension evolution experiment. Original crowding fill strategy is the second most unstable version for the basic experiment and the most unstable version for the extension experiment.

In summary, preserving population diversity based crowding fill strategy is always a relatively better approach and original crowding fill strategy is always not a good choice with the relatively unstable performance for both of the original and the extension evolution experiment. Although there is currently not a best scheme for the improved crowding fill strategy utilised for MOCGPANN, the original crowding fill strategy referred from NSGA2 seems not to be working very well. Therefore, the cumulative mean result for these two evolution experiment implies that the original crowding fill strategy may not suite the MOCGPANN's survival selection. However, which version to be utilised for crowding fill strategy is still an open question and whether preserving genetic drift or population diversity is beneficial to the evolutionary search is also not confirmed. In this sense, further investigation is required to find out whether the crowding fill strategy improvement really has any significant benefit for the survival selection in MOCGPANN as future works.

Table 6.3: Result of number of experiment runs required from cumulative mean approach for the extension experiment

	1st version	2nd version	3rd version	Original
Number of runs	147	173	164	191

6.3.2 Generalisation experiment

As is mentioned in section 6.2.2, generalisation experiment was conducted for robust and switched fault tolerant control respectively with four generalised scenarios based on the success rate for each test. The only difference for this extension experiment is that the left 4 sensors of the robot will be failed during the phototaxis instead of the original experiment designed in section 4.4.2 with only 2 failed sensors. With regard to the controller acquirement, the 3rd version of MOCGPANN was utilised as concluded in section 6.3.1 to obtain 5 different Pareto sets of controllers in order to be implemented for this generalisation experiment. The data of these five Pareto sets' generalisation experiment result can be referred to the Appendix A.3.

6.3.2.1 Robust fault tolerant control

Table 6.4 lists the success rate for 5 obtained Pareto sets of controllers based on 4 different generalised scenarios with 10 different trials for each test in terms of the robust fault tolerant control. As can be seen from Table 6.4, except for the new fault signal scenario, all the initial scenario, the new robot condition scenario and the new beacon position scenario obtained 100% success rate for each test based on the obtained 5 Pareto sets. This result indicates that MOCGPANN is capable to evolve effective controllers that could achieve the robust fault tolerant control even for a more difficult scenario with 4 sensors not working.

Table 6.4: Success rate for extension generalisation experiment results in terms of robust fault tolerant control

	Initial scenario		Fault signal		Robot condition		Beacon position	
	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700
Pareto 1	1	1	0	1	1	1	1	1
Pareto 2	1	1	0	0	1	1	1	1
Pareto 3	1	1	0	1	1	1	1	1
Pareto 4	1	1	1	1	1	1	1	1
Pareto 5	1	1	0	0	1	1	1	1

6.3.2.2 Switched fault tolerant control

Table 6.5 lists the success rate for 5 obtained Pareto sets of controllers based on 4 different generalised scenarios with 10 different trials for each test in terms of the switched fault tolerant control. As can be seen from Table 6.5, the initial scenario and the new robot condition scenarios achieved 100% success rate for each test. However, the evolved 5 Pareto sets of controllers were not working very well in terms of the new fault signal scenario and the new beacon position scenario. Nevertheless, this result still demonstrates that MOCGPANN is capable to obtain effective controllers that could be used to achieve the switched fault tolerant control for this extension experiment.

Table 6.5: Success rate for extension generalisation experiment results in terms of switched fault tolerant control

	Initial scenario		Fault signal		Robot condition		Beacon position	
	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700	Time: 300	Time: 700
Pareto 1	1	1	0	1	1	1	0.7	0.8
Pareto 2	1	1	0	1	1	1	1	1
Pareto 3	1	1	0	0.1	1	1	0.8	0.8
Pareto 4	1	1	1	1	1	1	1	1
Pareto 5	1	1	0	0	1	1	1	1

6.3.2.3 Comparison between two fault tolerant control schemes

As can be seen from section 6.3.2.1 and 6.3.2.2, MOCGPANN is capable to obtain effective controllers that could be implemented to achieve both of robust and switched fault tolerant control for this extension experiment. Based on these generalisation results, Table 6.6 lists the comparison result for these two approaches for fault tolerant control based on the A-test in terms of the success rate for each test.

Table 6.6: Comparison between the success rate of robust and switched fault tolerant control based on the controllers evolved by MOCGPANN for this extension experiment

Scenario	Initial scenario	Fault signal	Robot condition	Beacon position
A-test score	0.5	0.469136	0.5	0.666667

As can be seen from Table 6.6, except for the same performance between robust and switched fault tolerant control for initial and new robot condition scenarios, the new fault signal scenario obtained an A-test score of 0.469136 with nearly no significant difference and the new beacon position scenario obtained an A-test score of 0.666667 which locates in the medium effect size interval. Furthermore, the A-test score of the new beacon position scenario is above 0.5 which indicates that switched fault tolerant control produced a worse performance than robust fault tolerant control in this scenario with a medium effect size. This result is unusual since switched fault tolerant control will normally outperforms robust fault tolerant control as mentioned in section 2.2. In this case, a possible reason is that this extension experiment is quite complicated, so which one will be performing better is not based on their basic evolution experiment result. That is to say, due to the more difficult fault tolerance scenario with 4 sensors failed instead of 2 sensors, those evolved controllers' performance may not be predicted for these generalised scenarios. On the other hand, according to the generalisation results, it is similar to section 4.5.2 where the evolved controllers are not working very well in the new fault signal and new beacon position scenarios. Therefore, it is still required to consider more generalised scenarios like these two into the controller evolution in order to achieve more generalised fault tolerant control as future works.

6.4 Summary

This chapter investigates how MOCGPANN could be performing to obtain fault tolerant controllers for this more difficult extension experiment. This chapter is actually motivated by the conclusion in section 5.3.1.3 where MOCGP outperforms NSGA2 for NE based fault tolerance evolution experiment, so that is why MOCGPANN is tested for this extension work.

Moreover four different crowding fill strategies are also compared between each other for the extension evolution experiment. However the 3rd version is slightly better than the other three versions, so this one would be utilised to obtain controllers for the generalisation experiment. It also needs to note that the 1st version achieved the most stable performance than the other three. The last version, which is the original crowding fill strategy, is the most unstable one. This result is similar to that in section 4.5.1.2 where the 1st version is the most stable one and the 4th version is the second most unstable one. Although there is no obvious significant difference among these four versions in terms of their performance, at least it can be concluded that preserving population diversity based crowding fill strategy could produce relatively more stable performance than the original crowding fill strategy. However, whether a more stable performance is really helpful for the evolutionary search is still not clear at the moment, which needs further investigation as future works.

In terms of the generalisation result, MOCGPANN demonstrates effective performance to obtain controllers so as to achieve both of robust and switched fault tolerant control.

Although switched fault tolerant control performed better than robust fault tolerant control, it can be inferred that the reason is due to the more difficult scenario for the generalisation tests. On the other hand, the obtained controllers were still not working very well for new fault signal and new beacon position scenarios, which is the same as the conclusion in section 4.5.2 for the basic experiment. In this case, more scenarios could be considered during the solution evaluation to obtain more generalised fault tolerant controllers in terms of more generalised robot phototaxis task, which could be investigated as future works.

Chapter 7 Conclusion

7.1 Summary and contributions

A summary for each chapter along with the key contributions will be listed as following:

- Chapter 2 reviewed different fault tolerant control schemes and different structure optimisation based EA approaches along with NE approaches. Moreover, different multi-objective optimisation algorithms were also reviewed along with different crowding measure approaches. Finally, convergence criteria and statistics analysis were also reviewed.

  Contribution: CGPANN was selected as the main approach used to evolve ANN controllers based on the structure optimisation so as to achieve the design of a fault tolerant control system.
- Chapter 3 investigated how CGPANN could be utilised to design a robust robot fault tolerant control system.

  Contribution: The result shows that CGPANN was failed to achieve it since single objective optimisation is not adequate to design a robust fault tolerant controller. For this reason, MOCGPANN was identified as the new approach to achieve the design of a robust as well as a switched fault tolerant control system.
- Chapter 4 presented how MOCGPANN was developed based on the integration of CGPANN and NSGA2 as the main approach for multi-objective controller structure evolution.

  Contribution: During the development of MOCGPANN, preserving population diversity was considered as a solution to solve the problem of the random neighbour pick in the original crowding fill strategy and it also displayed a relatively more stable performance than the original one, although their performance had no significant difference between each other. On the other hand, hyper-volume indicator was successfully used to measure the performance of MOCGPANN so as to assess its convergence without the requirement of a true Pareto optimal front. In terms of the generalisation test, the evolved controllers by MOCGPANN demonstrated effective performance to achieve both of robust and switched fault tolerant control based on the generalised scenarios. Although new fault signal and new beacon position scenarios were not suited to the evolved controllers, those controllers achieved

100% success rate in terms of the initial scenario and the new robot initial condition scenarios, which actually filled the research gap that controller structure optimisation had not been investigated into fault tolerant control.

- Chapter 5 presented how NSGA2 was utilised as the approach for controller parameter optimisation in order to conduct a comparison work with MOCGP based on the ANN controller evolution.

  Contribution: The result shows that NSGA2 performed worse than MOCGP for the controller evolution in spite of a pre-optimised ANN structure for NSGA2, which verified that network structure optimisation outperformed connection weight optimisation even in multi-objective optimisation for fault tolerant control. Nevertheless, the controllers evolved by NSGA2 still achieved the robust and switched fault tolerant control. However its result was similar to MOCGP where the evolved controllers were just working for some of the generalised scenarios.
- Chapter 6 presented how MOCGPANN was utilised further for a more difficult extension experiment where there were more sensors failed during the task.

  Contribution: The result shows that MOCGPANN was still capable to obtain feasible controllers so as to achieve robust and switched fault tolerant control. Furthermore, different crowding fill strategies were also compared for the evolution result. The comparison result also demonstrated that preserving population diversity based crowding fill strategy obtained more stable performance than the original one, which means the original crowding fill strategy is really not suited to MOCGPANN. In terms of the generalisation, it was still similar to the basic experiment where just parts of the generalised scenarios were suited to the evolved controllers. This work further answered the overall research question that MOCGPANN was able to evolve controller even for a more difficult fault tolerant control task, which demonstrates the effective performance of MOCGPANN implemented into fault tolerant control area. In other word, this work also indicates that controller structure optimisation will be an effective solution utilised for evolving fault tolerant robotic controllers.

7.2 Future works

This thesis also left some future works which may need further investigation. The future works can be categorised into two aspects: the optimisation algorithms and the robot test cases, which will be presented as following.

7.2.1 Future works about the optimisation algorithms

- Further benefit of preserving population diversity

During the development of MOCGPANN, preserving population diversity based crowding fill strategy was demonstrated to achieve a relatively more stable performance than the original crowding fill strategy. However they actually had no significant difference between each other in terms of the hyper-volume and generation number. So at the moment, it is not clear whether a more stable performance could result in any further advantages. A possible further investigation is to disable the connection weight genes to create the ANN controller. In this case, all the evolved ANNs' weights will be equal to 1. This modification will further demonstrate whether preserving population diversity or genetic drift will aid the evolutionary search without the contribution of connection weight genes in order to find out the best way of improving crowding fill strategy for MOCGP even in general multi-objective optimisation problems apart from NE.

- Further comparison between MOCGP and NSGA2

This work utilises just a single hidden layer for the basic of the ANN's structure in terms of NSAG2 evolution experiment. Although one hidden layer is considered to solve a majority of problems, it is still unknown whether one hidden layer is adequate for this work. For this reason, more hidden layers should be worth a further investigation for the ANN's structure evolved by NSGA2. On the other hand, this work just utilises one option for the population size. Although the number of this population size is sufficient to create the Pareto optimal front, it is unknown whether this number is the best choice for either NSGA2 or MOCGP. In this case, different options of population size should be considered to conduct the comparison between NSAG2 and MOCGP in order to investigate how the population size impacts the performance of these two algorithms.

- NEAT/HyperNEAT for the same evolution experiment

Although CGPANN produces better performance than NEAT/HyperNEAT for a wide range of application areas, it is actually unknown whether CGPANN still outperforms NEAT/HyperNEAT for this work. On the other hand, NEAT/HyperNEAT also belongs to structure optimisation based NE approach, so NEAT/HyperNEAT is also worth a further investigation in terms of multi-objective optimisation for the same evolution experiment.

7.2.2 Future works about the robotic test case

- Further investigation about the controller performance for generalised scenarios

In terms of all the generalisation experiments, they all had got a same problem that the evolved controllers were not working very well when the fault signal was changed to another value and when the beacon was moved to a new position. A possible explanation is that the current scenario for controller evolution is restricted to only a single type of fault signal, which is 0 in this case. Although the evolved controllers are robust to different fault occurrence time and different robot initial conditions, their performance could be dramatically degraded once the fault signal is changed during the robot task. In this case, the robustness to fault signal value would be the first task for further investigation. An initial solution is to consider evolving controllers without the connections from the faulty sensors. Although this solution will make the controllers capable to tolerate any fault signal values, the performance for the fault-free scenario may be degraded due to the loss of connections from working sensors. For this reason, a more effective solution is to set random fault signals during the controller evolution. This approach may improve the performance when different fault signal values are injected for the online testing experiment as long as sufficient random fault signal values are tested during the solution evaluation for the offline evolution experiment. In summary, further investigation should be conducted for the problem of the robustness to varied fault signals during the controller evolution. The other limitation of this work is that the evolved controllers are not robust to different beacon positions so as to achieve the phototaxis task. A possible solution is to set a series of different distances covering all the possible positions in the arena rather than a fixed distance between the robot initial position and the beacon position, in order to obtain more robust controllers to achieve the phototaxis no matter where the beacon is. If both of these two problems could be solved, this work will be possibly able to achieve not only a more generalised fault tolerant control system design but also a more generalised robot phototaxis task.

- Online controller evolution

All this work is about the offline controller evolution utilised for online robot fault tolerance task. The reason to conduct the offline controller evolution is based on the assumption that the possible upcoming fault types have already be considered into the controller evolution such as complete failures occurred in sensor 1 and 7 in this work, so offline controller evolution is enough in this case. However it cannot guarantee whether all the fault types have been included to evolve controllers especially when there are some unplanned faults occurred during the robot online task such as other sensor faults apart from sensor 1 and 7. In this sense, an online controller evolution needs to be conducted in case that an unplanned fault is occurred during the robot online task. In a word, online controller evolution is worth a further investigation in order to design a complete fault tolerant control system.

- Another robotic test case

This work actually just utilises one test case for evolving the robotic fault tolerant controllers, which is the robot phototaxis task. Actually, phototaxis is not a difficult task for a single robot to complete. So phototaxis may not be sufficient to test the algorithm developed in this work. For this reason, one option is to set some obstacles in the simulation area in order to investigate how the evolved controller could be performing for phototaxis with obstacle. Another option is to set a completely different robot test case such as wall following. Wall following is actually more difficult than phototaxis to evolve feasible controllers. In terms of phototaxis, there is always one or more sensors that could detect the light source as long as the robot is not too far away from the beacon. So it is not quite difficult to evolve feasible controllers to complete the task. However in terms of wall following, the robot could easily get stuck to the wall or just move away from the wall since the proximity sensor has a much shorter detection distance than the light sensor. For this reason, it may be more difficult to obtain feasible controllers to complete the wall following rather than phototaxis. In summary, another more difficult robot test case needs to be considered in order to test the generality of the algorithm developed in this work.

Appendix A

Appendix A lists all the generalisation experiment data throughout the thesis. Each datum in the tables means the time spent for the tested Pareto set of controllers for each trial. If the robot could achieve the phototaxis within 1200 ticks, the result would be in the ticks when the robot finishes the task. However, if the robot could not achieve the phototaxis task within 1200 ticks, a result of “n” would indicate a failed test. It needs to note that there is no time limit for the “Beacon position” test scenario. In this case, as long as the robot could achieve the phototaxis task, a “y” will represent a successful phototaxis task; otherwise an “n” will indicate a failed phtotaxis task.

A.1 Generalisation result based on MOCGP evolved controllers for the basic experiment

- Pareto 1

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	943	943	n	965	908	908	y	y	943	943	941	941	908	908	y	y
	919	919	n	935	911	911	y	y	920	920	919	919	911	911	y	y
	947	947	n	971	926	926	y	y	946	946	944	944	927	927	y	y
	948	948	n	972	919	919	y	y	946	947	945	945	919	919	y	y
	923	922	n	937	937	937	y	y	924	921	1015	987	938	939	y	y
	929	927	n	943	932	930	y	y	929	927	1010	994	932	929	y	y
	929	929	n	947	934	934	y	y	929	929	928	928	935	935	y	y
	916	914	n	927	914	911	y	y	917	914	1007	979	915	912	y	y
	909	909	n	923	915	913	y	y	910	910	909	909	916	913	y	y
	929	929	n	948	925	925	y	y	930	930	929	929	926	926	y	y

- Pareto 2

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	918	918	n	n	909	912	y	y	914	913	n	n	910	910	y	y
	914	917	n	n	912	915	y	y	915	915	n	n	914	914	y	y
	915	915	n	n	928	931	y	y	911	911	n	n	929	929	y	y
	935	938	n	n	920	923	y	y	937	937	n	n	922	922	y	y
	922	925	n	n	939	942	y	y	924	924	n	n	941	940	y	y
	947	947	n	n	933	933	y	y	942	942	n	n	928	928	y	y
	921	924	n	n	935	938	y	y	923	923	n	n	937	937	y	y
	918	918	n	n	915	915	y	y	914	914	n	n	911	911	y	y
	903	906	n	n	917	916	y	y	904	904	n	n	912	912	y	y
	938	938	n	n	927	930	y	y	934	933	n	n	928	928	y	y

- Pareto 3

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	926	926	n	954	910	910	y	y	926	926	n	944	908	908	y	y
	941	941	n	971	913	913	y	y	940	940	n	959	912	912	y	y
	940	940	n	971	929	929	y	n	940	940	n	958	927	927	y	y
	906	906	n	933	921	921	y	n	905	905	n	922	919	919	y	y
	941	940	n	969	940	940	y	y	940	940	n	960	938	938	y	y
	939	939	n	969	927	927	y	y	938	938	n	957	927	927	y	y
	913	912	n	941	936	936	y	y	912	912	n	928	935	935	y	y
	941	941	n	971	910	910	y	y	940	940	n	960	909	909	y	y
	945	945	n	976	911	911	y	n	943	943	n	962	911	911	y	y
	924	923	n	952	928	928	y	n	923	923	n	941	926	926	y	y

- Pareto 4

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	943	943	n	n	952	952	n	n	942	944	960	953	951	952	y*	y*
	904	904	n	n	936	936	y	y	905	905	921	915	934	935	y	y
	924	924	n	n	920	919	y	y	923	924	940	935	918	919	y	y
	937	937	n	n	928	928	y	y	936	937	953	947	926	927	y	y
	922	922	n	n	909	909	y	y	921	922	938	933	909	910	y	y
	916	916	n	n	919	919	n	n	915	916	932	926	918	920	y*	y*
	913	913	n	n	912	912	y	y	913	914	930	924	911	912	y	y
	923	923	n	n	936	936	y	y	922	923	939	934	935	936	y	y
	926	926	n	n	935	934	y	y	926	928	943	937	934	936	y	y
	934	934	n	n	921	921	n	n	932	934	949	943	919	920	y*	y*

Note: * means that in terms of the current test, the robot stays still until the other controller is loaded. However these trials all make the robot reach the beacon finally, so it is actually a successful phototaxis if the criterion is whether the robot achieves the phototaxis eventually or not.

- Pareto 5

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	925	925	926	925	937	937	y	y	925	925	932	925	938	938	y	y
	927	927	928	927	934	934	n	n	928	928	934	928	934	934	y	n
	940	940	942	940	918	918	y	y	940	940	948	940	919	919	y	y
	938	938	940	938	926	926	y	y	938	938	944	938	926	926	y	y
	945	945	945	945	908	908	y	y	941	941	948	941	908	908	y	y
	918	918	918	918	922	922	y	y	915	915	921	915	919	919	y	y
	913	911	913	912	911	911	n	n	909	909	914	909	911	911	n	n
	937	937	939	937	940	940	y	y	937	937	943	937	936	936	y	y
	908	909	908	908	938	938	n	n	905	905	911	905	935	935	y	y
	920	920	921	920	920	920	y	y	917	917	923	917	920	920	y	y

A.2 Generalisation result based on NSGA2 evolved controllers for the basic experiment

- Pareto 1

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	927	925	n	965	987	987	y	y	937	927	n	959	982	977	y	y
	959	951	n	1038	994	995	y	y	965	958	n	1005	976	971	y	y
	929	928	n	964	947	939	y	y	938	927	n	959	956	948	y	y
	930	929	n	964	961	953	y	y	939	927	n	959	966	959	y	y
	959	958	n	1011	930	925	y	y	981	970	n	1019	942	932	y	y
	953	953	n	1001	950	950	y	y	973	961	n	1006	969	957	y	y
	944	935	n	1008	936	928	y	y	953	946	n	987	946	938	y	y
	966	966	n	1026	969	968	y	y	993	984	n	1040	998	990	y	y
	991	991	n	1074	967	967	y	y	978	973	n	1027	995	986	y	y
	943	934	n	1006	949	940	y	y	952	944	n	985	957	950	y	y

- Pareto 2

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	946	947	n	n	972	972	y	y	944	944	n	n	950	951	y	y
	953	954	n	n	1022	1022	y	y	951	951	n	n	983	984	y	y
	939	940	n	n	1002	1003	y	y	938	938	n	n	1005	1004	y	y
	964	965	n	n	1032	1033	n	n	963	962	n	n	1053	1052	n	n
	956	957	n	n	983	984	y	y	955	954	n	n	983	982	y	y
	933	935	n	n	950	950	n	n	926	928	n	n	948	948	n	n
	955	956	n	n	989	989	n	n	940	941	n	n	989	988	n	n
	1012	1013	n	n	927	928	y	y	1019	1018	n	1185	925	926	y	y
	990	991	n	n	928	929	y	y	991	990	n	n	927	927	y	y
	954	955	n	n	1004	1005	n	n	953	952	n	n	1008	1007	n	n

- Pareto 3

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	919	918	n	993	976	972	y	y	917	925	n	972	975	970	y	y
	920	915	n	995	980	976	y	y	918	919	n	968	979	973	y	y
	974	970	n	n	1006	1002	n	n	972	967	n	1054	1004	998	n	n
	1025	1022	n	n	989	985	y	y	1020	1016	n	n	988	982	y	y
	939	935	n	1143	1011	1057	n	n	938	933	n	993	n	n	n	n
	920	922	n	1000	936	933	n	n	921	930	n	977	936	931	n	n
	955	951	n	n	1067	1065	y	y	953	948	n	1018	1052	1050	y	y
	946	943	n	n	957	953	n	n	945	940	n	1004	956	950	n	n
	970	967	n	n	955	952	y	y	969	964	n	1047	954	949	y	y
	932	941	n	1111	1002	999	n	n	939	950	n	1003	1000	995	n	n

- Pareto 4

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	928	991	n	n	1054	1050	y	y	925	990	n	n	1029	1025	y	y
	986	985	n	n	1028	1026	y	y	978	977	n	n	1011	1009	y	y
	954	952	n	n	993	992	n	n	947	946	n	n	985	984	n	n
	980	979	n	n	1004	1003	n	n	972	971	n	n	995	994	n	n
	950	948	n	n	982	981	n	n	944	943	n	n	974	973	n	n
	985	984	n	n	954	952	y	y	977	976	n	n	947	947	y	y
	1008	1007	n	n	986	984	n	n	998	997	n	n	977	977	n	n
	982	981	n	n	925	991	y	y	974	973	n	n	924	989	y	y
	1024	1022	n	n	927	994	y	y	1009	1007	n	n	925	990	y	y
	990	989	n	n	995	994	n	n	982	981	n	n	986	986	n	n

- Pareto 5

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	975	976	n	1003	978	979	y	y	956	963	965	964	959	966	y	y
	961	963	n	987	973	974	y	y	943	951	952	952	954	961	y	y
	966	967	n	993	952	954	y	y	948	955	957	957	935	943	y	y
	932	935	n	954	962	964	y	y	921	930	931	931	944	952	y	y
	936	934	n	953	932	934	y	y	928	932	938	935	921	929	y	y
	960	961	n	985	985	984	y	y	941	949	951	951	957	960	y	y
	955	957	n	981	940	943	y	y	938	946	947	947	926	934	y	y
	979	978	n	1004	1074	1070	y	y	954	957	966	960	1018	1022	y	y
	967	968	n	993	1096	1090	y	y	948	956	957	957	1029	1033	y	y
	943	945	n	967	954	956	y	y	927	936	937	937	936	944	y	y

A.3 Generalisation result based on MOCGP evolved controllers for the extension experiment

- Pareto 1

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	937	937	n	968	942	942	y	y	934	934	n	1016	1124	1123	n	y
	949	949	n	981	939	939	y	y	945	945	n	1035	937	937	y	n
	907	907	n	928	924	924	y	y	904	904	n	964	921	921	y	n
	933	933	n	963	931	931	y	y	931	931	n	1009	929	929	n	y
	931	931	n	959	913	913	y	y	928	928	n	1005	910	910	y	y
	947	947	n	978	920	920	y	y	947	947	n	1039	918	918	y	y
	917	917	n	944	916	916	y	y	915	915	n	984	913	913	y	y
	950	950	n	982	938	938	y	y	944	944	n	1033	935	935	y	y
	924	924	n	949	936	936	y	y	921	921	n	993	934	934	y	y
	917	917	n	943	925	925	y	y	915	915	n	982	922	922	n	y

- Pareto 2

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	936	942	n	n	941	948	y	y	934	939	n	1153	941	946	y	y
	937	943	n	n	938	945	y	y	936	941	n	1163	937	943	y	y
	938	944	n	n	927	929	y	y	937	942	n	1164	922	927	y	y
	921	922	n	n	933	937	y	y	914	920	n	1109	930	935	y	y
	932	935	n	n	917	918	y	y	927	933	n	1139	910	916	y	y
	913	913	n	n	925	927	y	y	906	911	n	1089	920	925	y	y
	933	936	n	n	920	922	y	y	928	934	n	1142	914	920	y	y
	936	942	n	n	938	944	y	y	934	939	n	1157	937	942	y	y
	947	954	n	n	937	943	y	y	947	952	n	1189	936	941	y	y
	920	922	n	n	928	930	y	y	914	920	n	1110	923	928	y	y

- Pareto 3

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	938	939	n	981	939	940	y	y	936	936	n	n	938	938	n	n
	931	932	n	972	937	936	y	y	929	929	n	n	934	934	y	y
	908	909	n	941	920	921	y	y	906	906	n	906	919	918	y	y
	928	929	n	967	928	928	y	y	926	926	n	n	926	926	y	y
	903	904	n	936	909	911	y	y	902	902	n	n	908	908	y	y
	904	905	n	936	920	921	y	y	902	902	n	n	918	918	y	y
	914	915	n	947	913	914	y	y	912	912	n	n	911	911	y	y
	937	938	n	979	937	938	y	y	936	936	n	n	935	935	y	y
	939	940	n	981	936	938	y	y	938	938	n	n	934	934	n	n
	938	940	n	980	921	922	y	y	936	936	n	n	920	920	y	y

• Pareto 4

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	946	946	1035	961	914	916	y	y	942	938	994	938	908	908	y	y
	951	951	1042	966	919	919	y	y	947	951	994	973	911	911	y	y
	911	911	986	924	927	935	y	y	907	904	955	904	927	928	y	y
	927	927	1009	941	927	927	y	y	922	919	974	919	919	920	y	y
	902	912	985	917	938	947	y	y	902	905	945	927	938	938	y	y
	936	947	1030	951	928	939	y	y	934	936	980	960	927	929	y	y
	938	947	1036	959	942	942	y	y	938	938	999	948	934	934	y	y
	914	925	1003	927	910	921	y	y	913	914	958	941	910	910	y	y
	916	916	993	929	912	923	y	y	908	908	953	908	911	915	y	y
	936	944	1031	957	926	934	y	y	936	939	983	965	926	928	y	y

- Pareto 5

Fault tolerant control type	Robust fault tolerant control								Switched fault tolerant control							
Test scenario	Initial scenario		Fault signal		Robot condition		Beacon position		Initial scenario		Fault signal		Robot condition		Beacon position	
Fault occurrence time (tick)	300	700	300	700	300	700	300	700	300	700	300	700	300	700	300	700
Result of the tested Pareto set of controllers (tick) or (yes/no)	914	913	n	n	911	910	y	y	914	914	n	n	907	907	y	y
	945	944	n	n	910	910	y	y	944	944	n	n	910	910	y	y
	919	919	n	n	932	932	y	y	919	919	n	n	926	926	y	y
	903	902	n	n	918	918	y	y	903	903	n	n	918	918	y	y
	942	941	n	n	945	945	y	y	941	941	n	n	937	937	y	y
	945	944	n	n	969	968	y	y	944	944	n	n	927	927	y	y
	949	948	n	n	934	933	y	y	940	940	n	n	933	933	y	y
	913	912	n	n	979	979	y	y	912	912	n	n	909	909	y	y
	954	953	n	n	989	990	y	y	935	935	n	n	910	911	y	y
	933	933	n	n	933	935	y	y	933	933	n	n	925	925	y	y

Appendix B

B.1 Sensitivity analysis for MOCGPANN parameters

This section shows how these parameters for MOCGPANN mentioned in section 4.5.1.1 are adjusted to their optimal values, which will be shown as the following.

- Nconv

In terms of the convergence criteria parameters, Figure B.1 shows the comparison of Nconv for hyper-volume and Figure B.2 shows that for generation number. Both of these 2 comparisons perturbed the parameter Nconv with value 20, 30 and 40. Figure B.3 displays the Vargha-Delaney A-test score for parameter Nconv pertubation from the Parameter Robustness technique developed in Spartan.

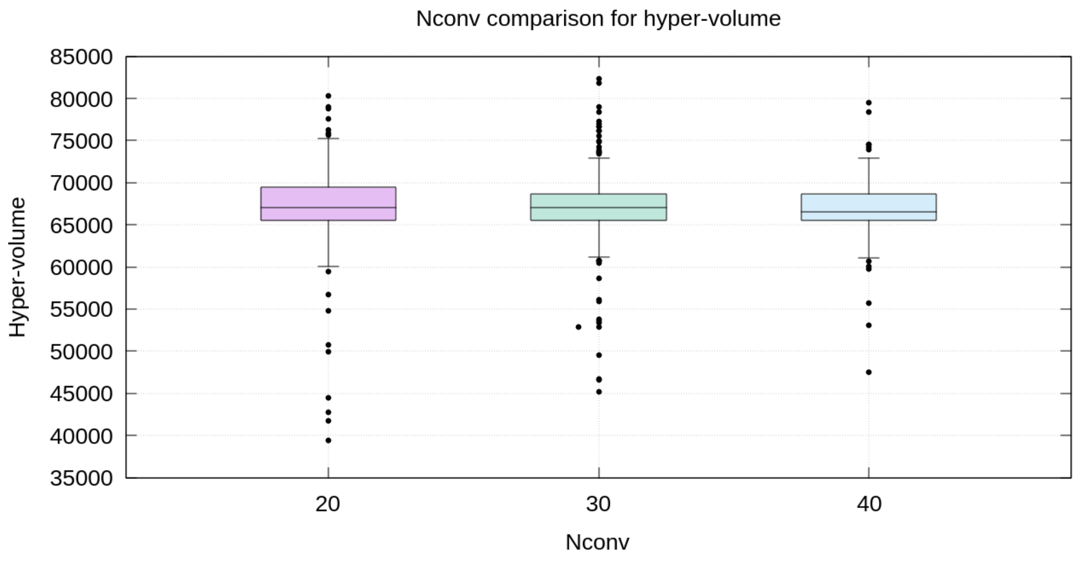

Figure B.1: Nconv comparison for hyper-volume

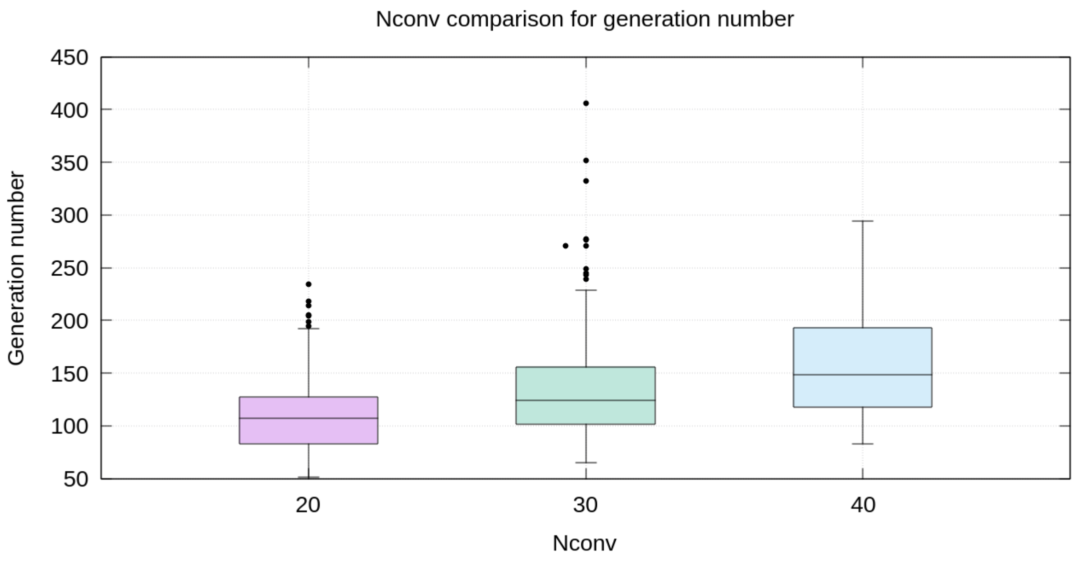

Figure B.2: Nconv comparison for generation number

As can be seen from Figure B.3, hyper-volume has nearly no difference among 20 to 40 and number of generations has more impact but still no large difference. Figure B.1 demonstrates a slight decrease of hyper-volume from 20 to 40 of Nconv, which can be ignored and Figure B.2 presents a more significant increase of number of generations from 20 to 40 of Nconv but still no large difference. In this sense, Nconv has no influence on hyper-volume among 20 to 40 but it has more or less impact on the number of generations in spite of no large difference. A possible explanation is that more Nconv could result in more generations to make the evolution converged. However there is still nearly no difference for hyper-volume, so it is worth trying smaller Nconv values and seeing whether it could make any changes for the experiment responses.

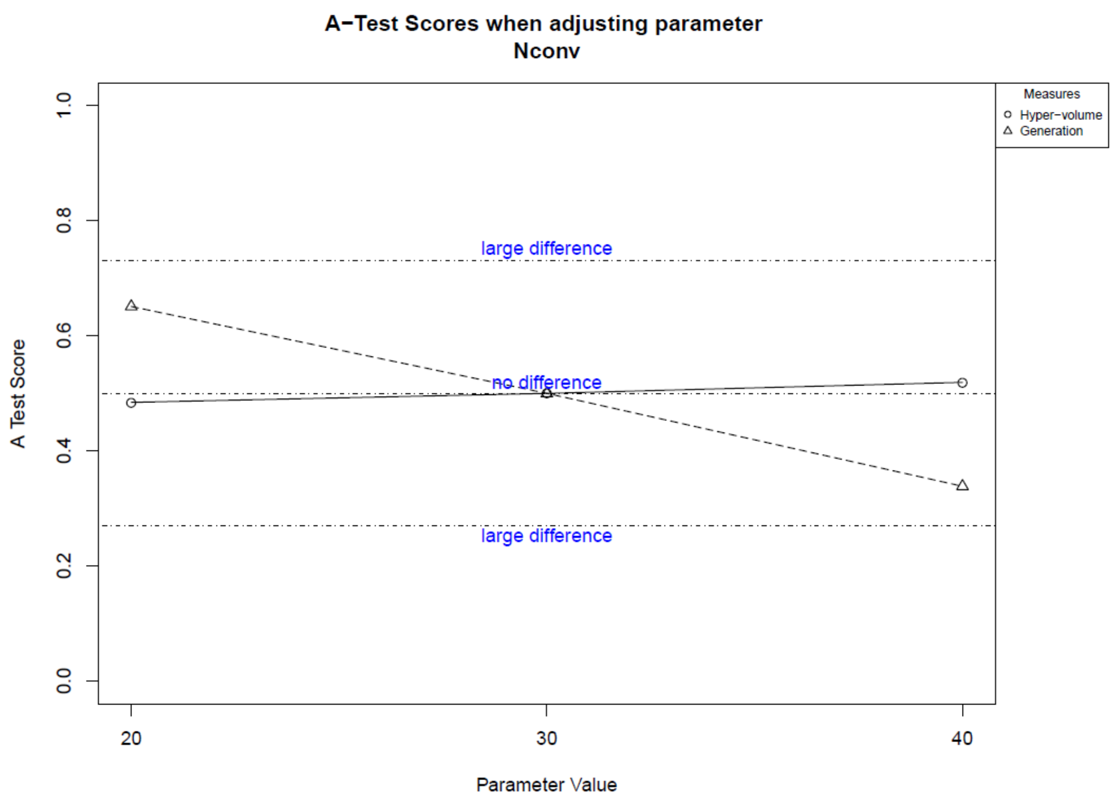

Figure B.3: A-test score for Nconv comparison

- Rconv

Figure B.4 shows the Rconv comparison for hyper-volume, Figure B.5 shows the Rconv comparison for generation number. Both of these 2 comparisons perturbed Rconv with 1.01, 1.001 and 1.0001. Figure B.6 displays the Vargha-Delaney A-test score for parameter Rconv.

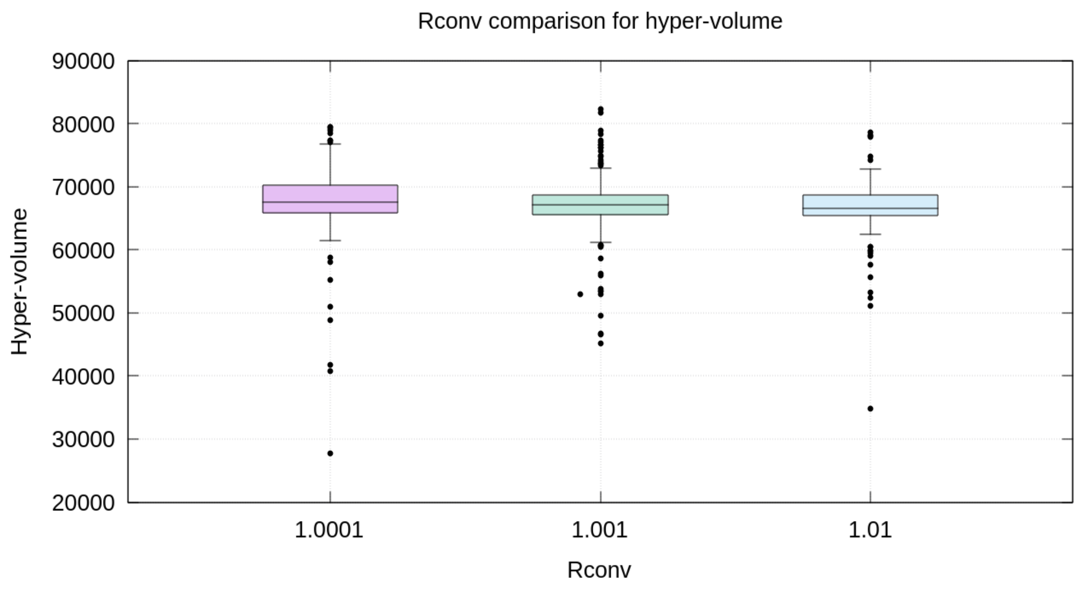

Figure B.4: Rconv comparison for hyper-volume

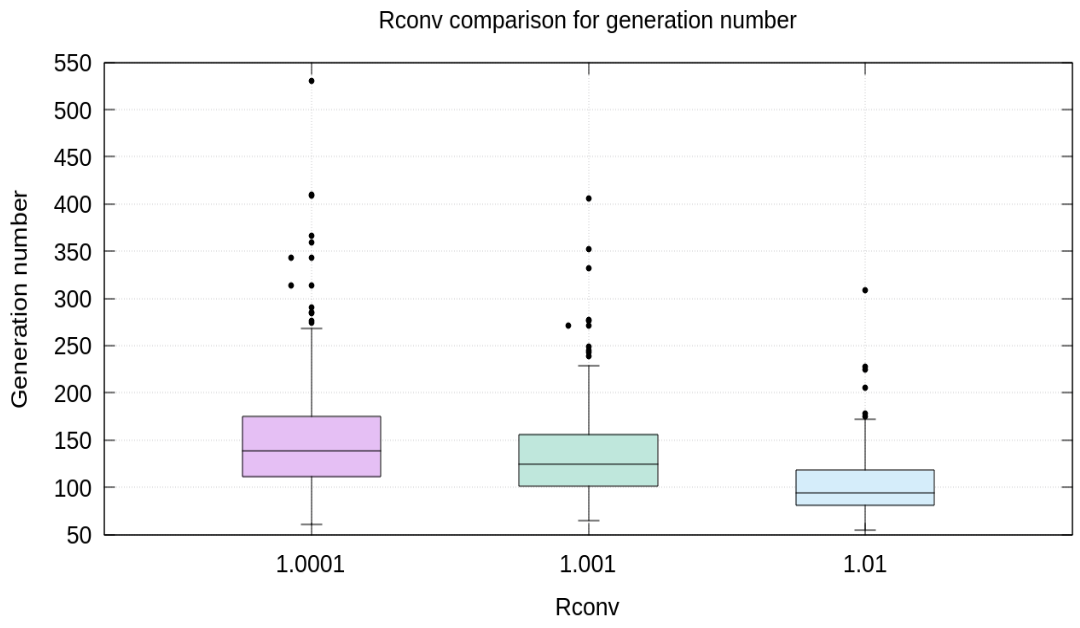

Figure B.5: Rconv comparison for generation number

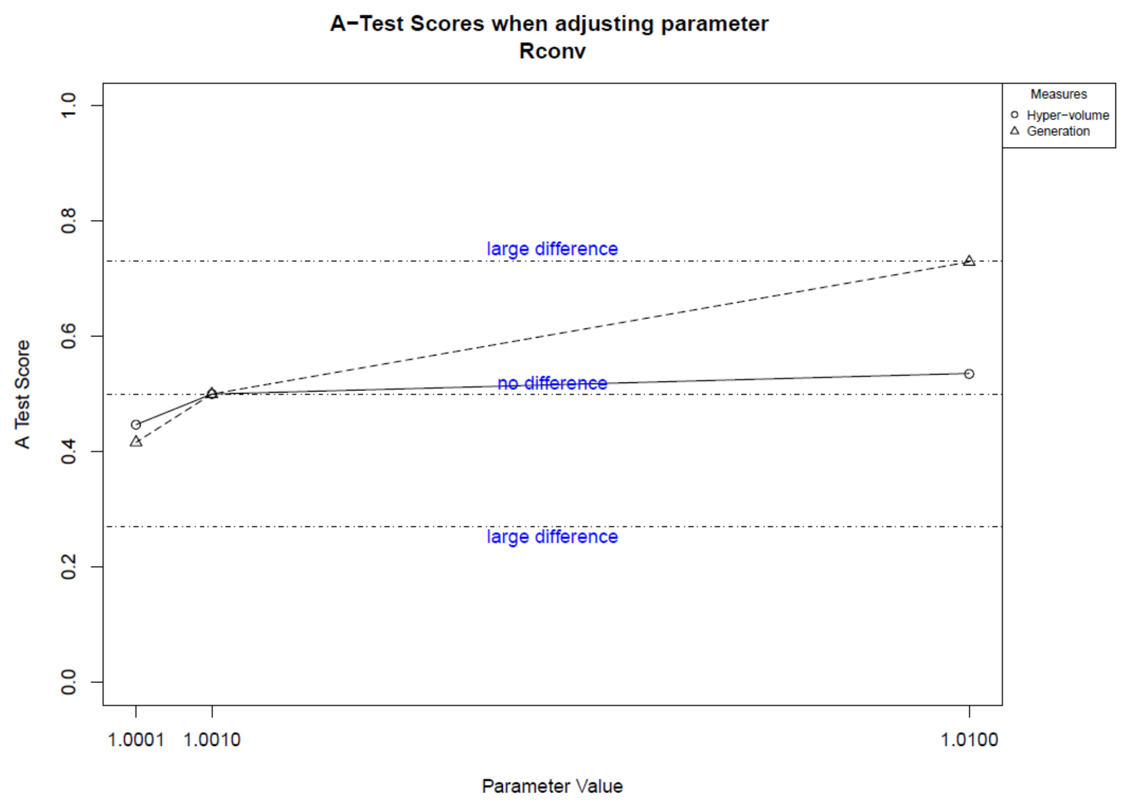

Figure B.6: A-test score for Rconv comparison

As can be seen from these results for Rconv, different parameter values actually lead to no significant different for hyper-volume. However larger Rconv values result in smaller responses for generation number especially when Rconv is 1.01, which has already caused large difference from Vargha-Delaney A-test. In this sense, it seems that larger Rconv value could result in better response for convergence. A possible explanation is that larger Rconv could make evolution converged earlier, but there is nearly no difference for hyper-volume response. So it is still worth trying to set larger values for Rconv and see whether there will be any improvement for the experiment responses.

- NumNode

In terms of the optimization algorithm parameters, Figure B.7 shows the hyper-volume comparison results for NumNode and Figure B.8 shows the generation number comparison results for it. Both of these 2 comparisons perturbed NumNode with 10, 20 and 100. And Figure B.9 indicates the Vargha-Delaney A-test score for those comparisons.

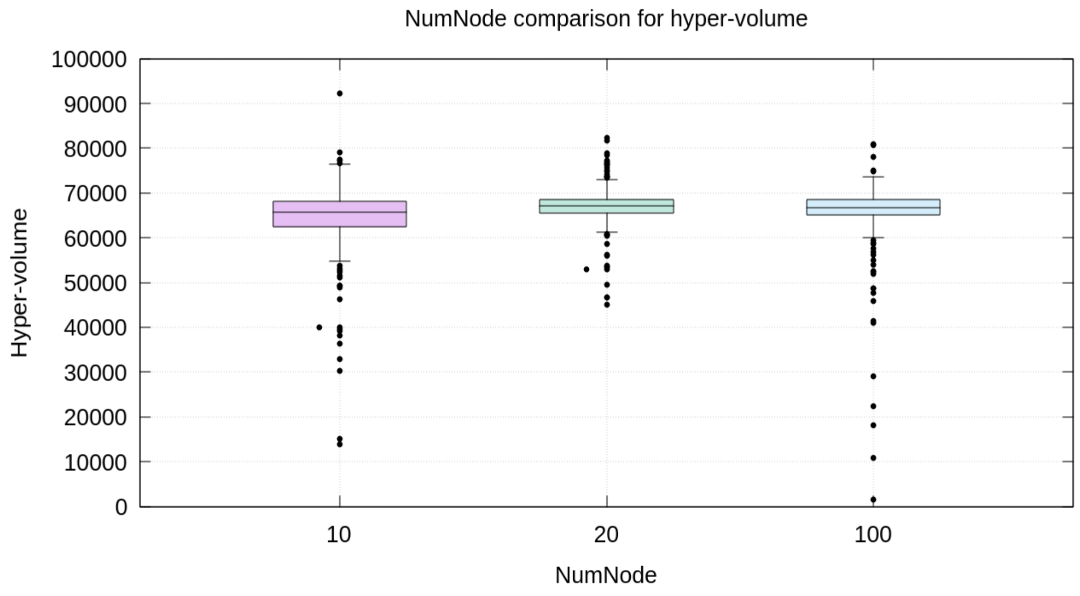

Figure B.7: NumNode comparison for hyper-volume

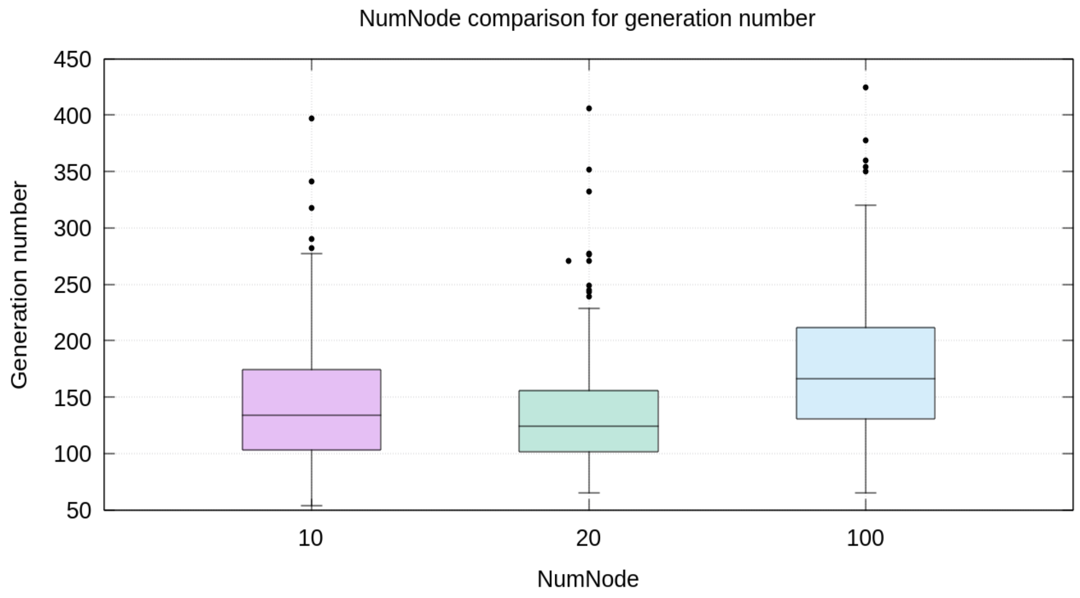

Figure B.8: NumNode comparison for generation number

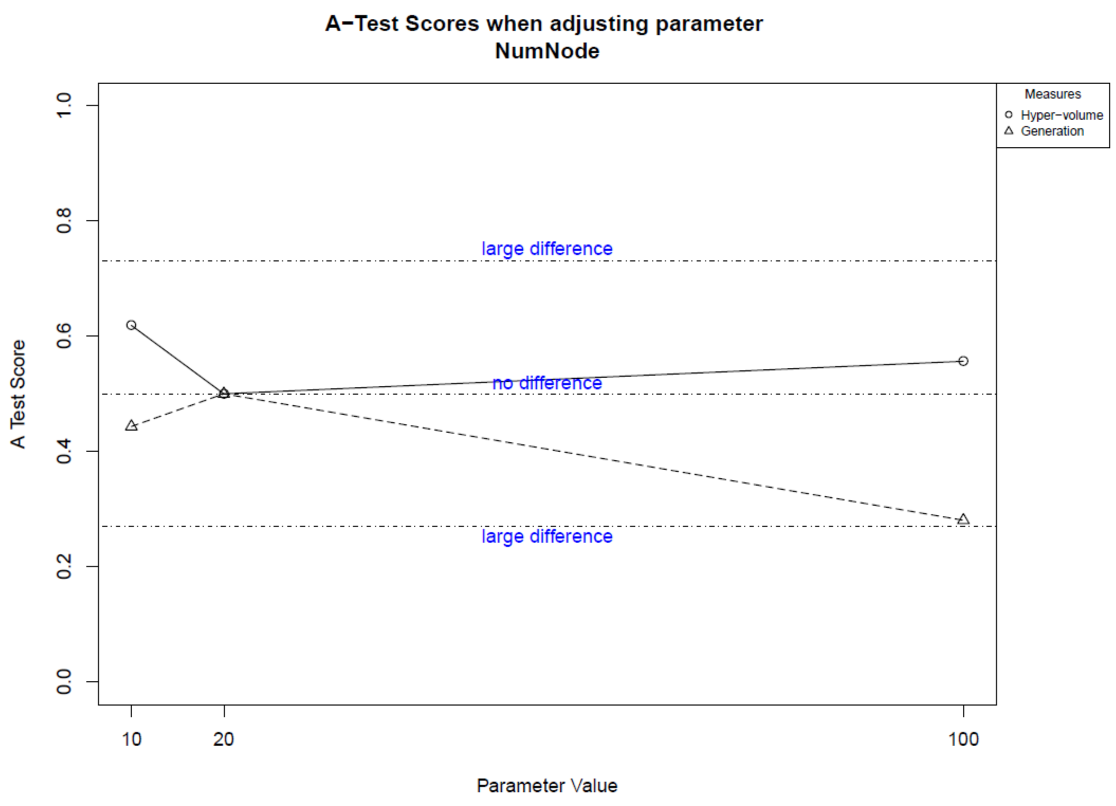

Figure B.9: A-test score for NumNode comparison

As can be seen from the comparison results, hyper-volumes of different NumNode values show no large difference. Even so, the hyper-volume of the baseline value 20 still achieves the best performance with the largest response. On the other hand, the generation number of value 100 nearly reaches the large difference criteria with a worst performance. Although there is no large difference for value 10, the generation number of baseline value 20 still obtains the least number of generations for convergence, which still performs the best. In conclusion, the baseline value 20 of NumNode achieves the best performance compared with 10 and 100 in terms of both hyper-volume and generation number responses. So there is no need to change the value of NumNode either for increase or decrease.

- NodeArity

Figure B.10 shows the hyper-volume comparison results for different NodeArity. Figure B.11 shows the generation number comparison result for them. Both of them perturbed NodeArity values with 2, 5 and 10. And Figure B.12 demonstrates the Vargha-Delaney A-test scores for those comparisons.

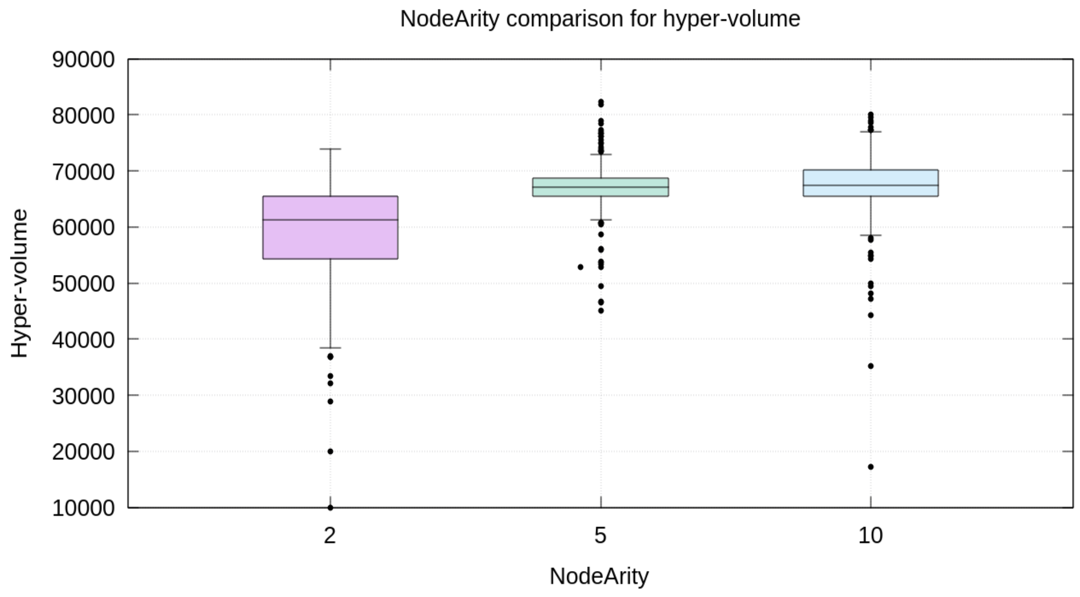

Figure B.10: NodeArity comparison for hyper-volume

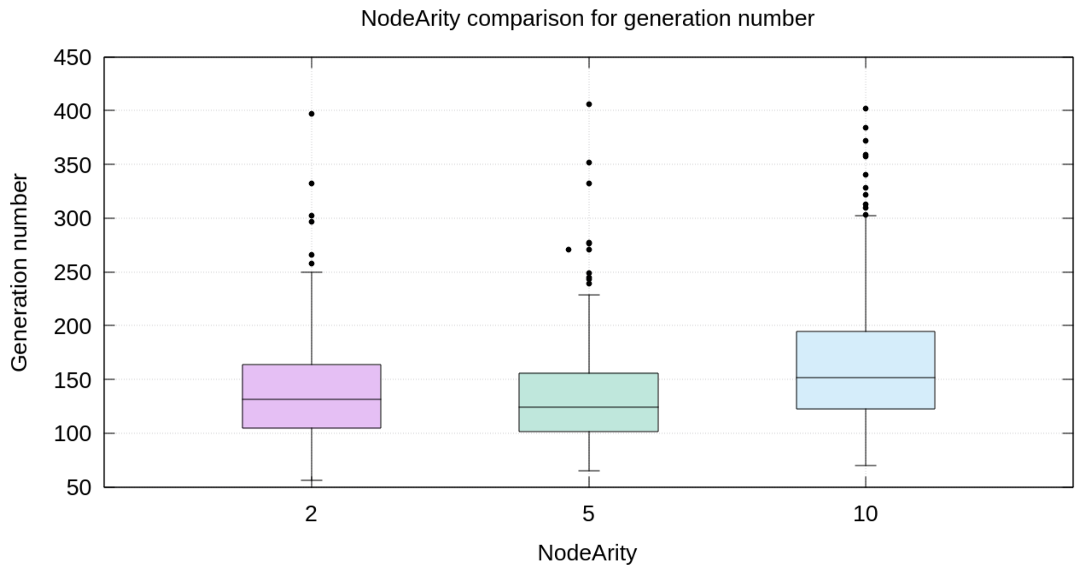

Figure B.11: NodeArity comparison for generation number

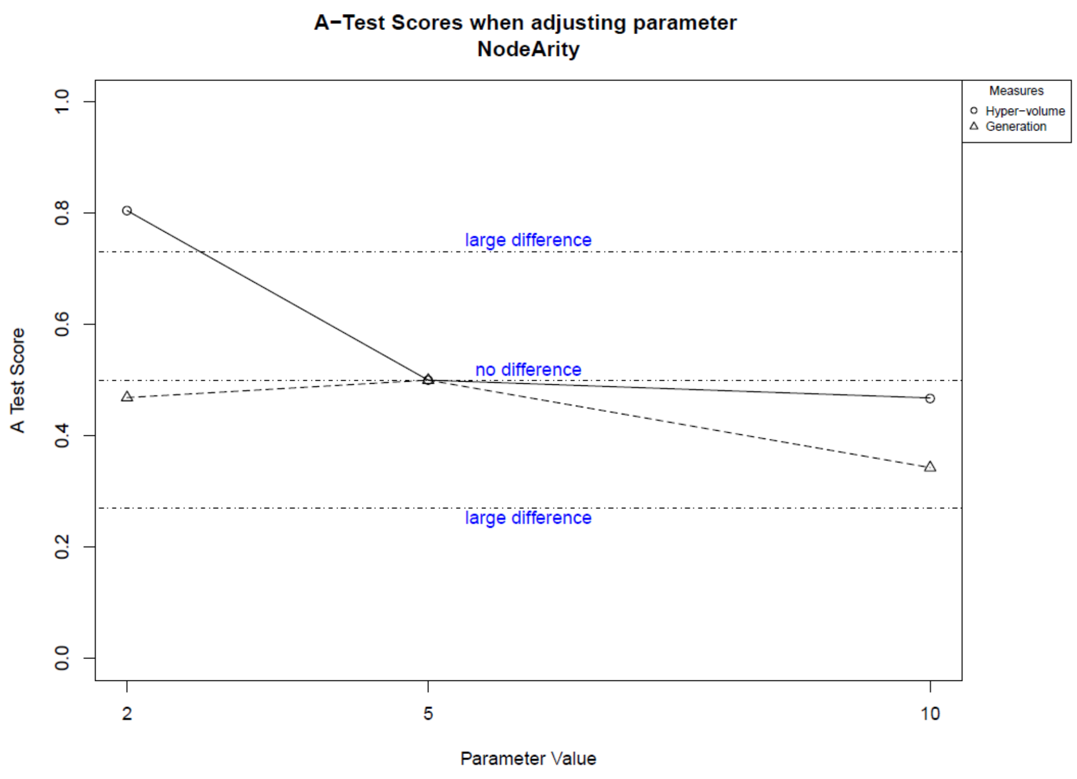

Figure B.12: A-test score for NodeArity comparison

As can be seen from the NodeArity comparison results, the value of 2 performs the worst with the lowest hyper-volume in large difference and slightly higher generation number compared to baseline value. The value of 10 also doesn’t achieve the best performance. Although the hyper-volume is just slightly higher than that from baseline value, it needs more generations than baseline value to make the evolution converged. And the significant difference of generation number is larger than that of hyper-volume, although they are both in the area of no large difference. A possible explanation is that more NodeArity may lead to a better response of hyper-volume, but it needs much more effort to make the evolution converged, which is not beneficial to the evolution. In conclusion, although there is no large difference between value of 10 and the baseline value, the results still indicate that the baseline value is the best option for NodeArity. So there is no need to further increase the baseline value.

- WeightRange

Figure B.13 shows the WeightRange comparison for hyper-volume and Figure B.14 shows that for generation number. WeightRange was perturbed with +/3, +/-5 and +/-10. And Figure B.15 illustrates the Vargha-Delaney A-test scores for those comparisons.

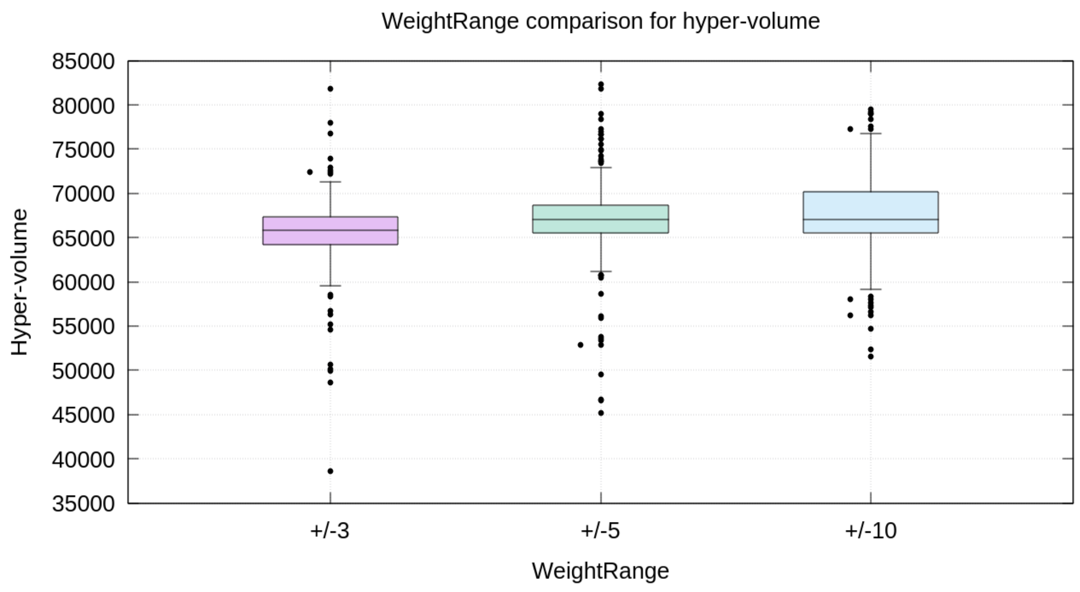

Figure B.13: WeightRange comparison for hyper-volume

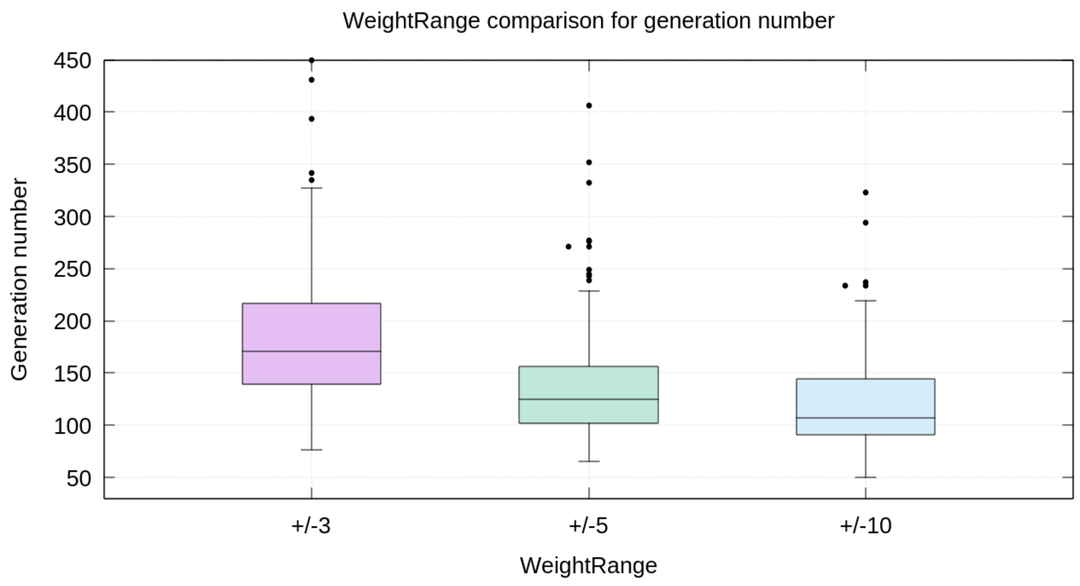

Figure B.14: WeightRange comparison for generation number

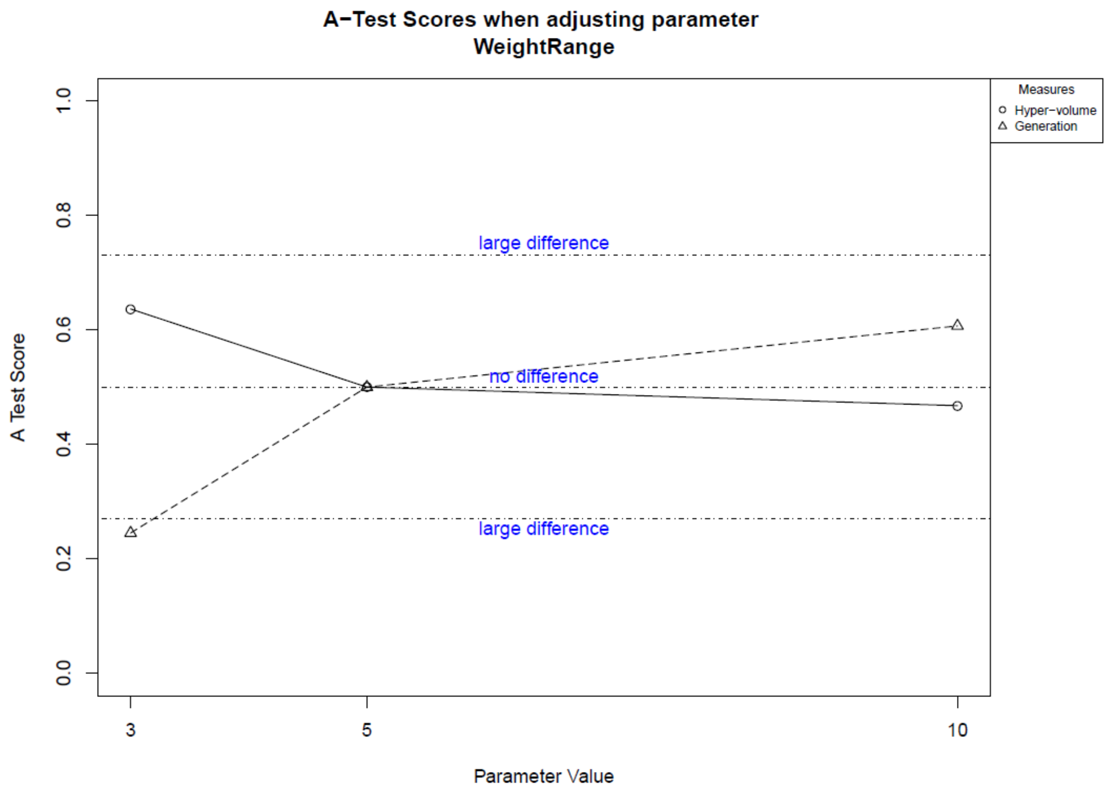

Figure B.15: A-test score for WeightRange comparison

According to the comparison results for WeightRange, the value of +/-3 obtains the worst performance with the lowest hyper-volume and largest number of generations to make evolution converged which has already caused the large difference compared to baseline value. However the value of +/-10 achieves a better performance than that of the baseline value of +/-5. The value of +/-10 has a slightly larger hyper-volume but with a much less generation number. Although there is no large difference for both of these 2 responses, the value of +/-10 still achieves larger hyper-volume with less generations compared to baseline value responses, which indicates a promising way to get better performance with larger WeightRange. In conclusion, it is worth trying to set a larger WeightRange value than +/-10 and see whether there will be any further improvement on the experiment performance.

- MutationRate

Figure B.16 shows the MutationRate comparison for hyper-volume and Figure B.17 shows that for generation number. MutationRate was perturbed with value of 0.01, 0.05 and 0.1. Figure B.18 illustrates the Vargha-Delaney A-test scores for the comparisons.

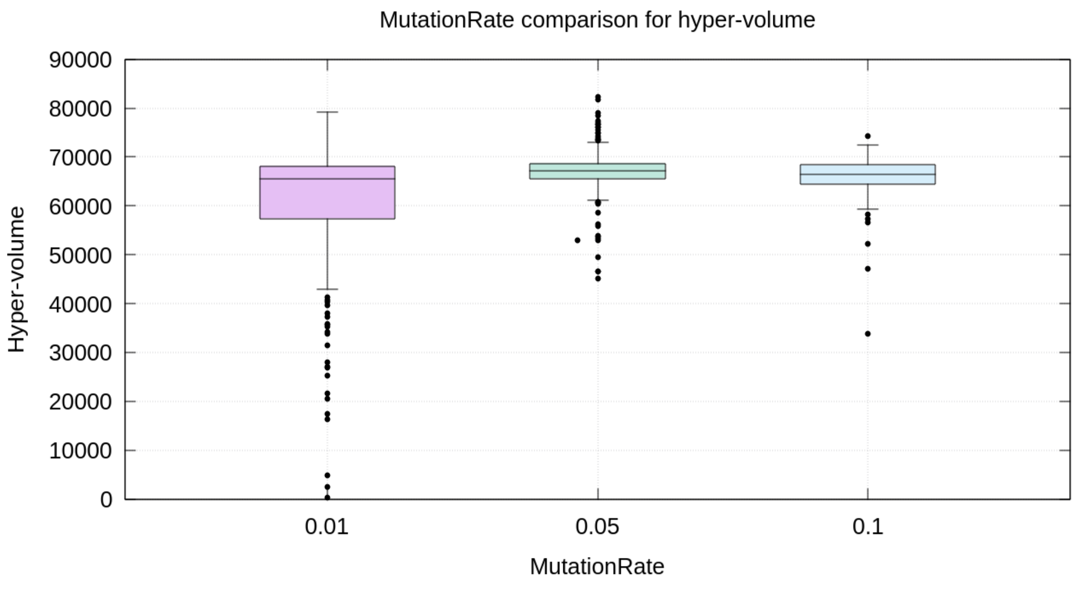

Figure B.16: MutationRate comparison for hyper-volume

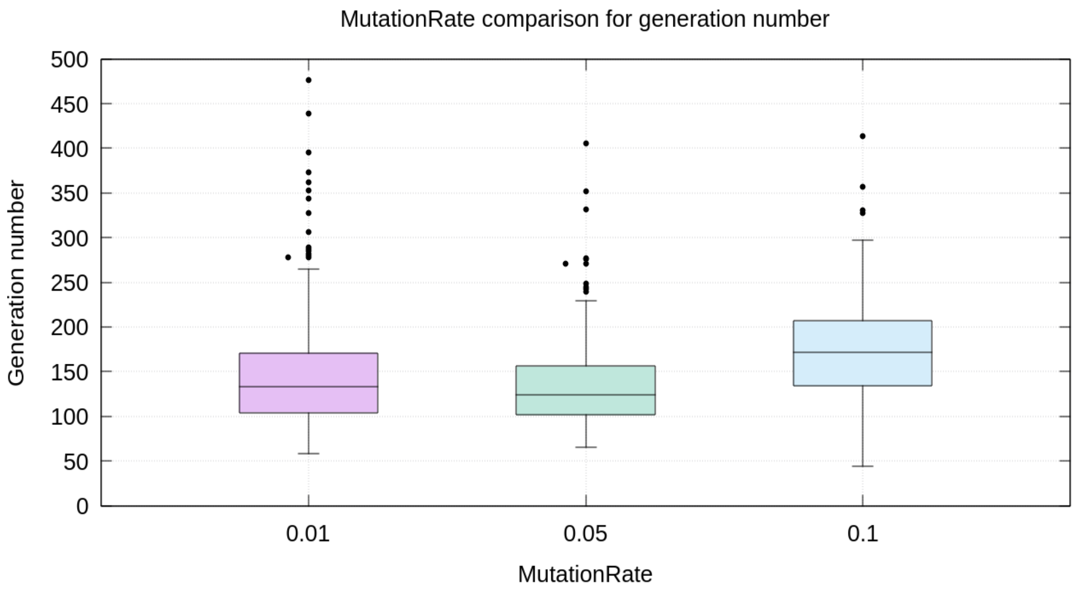

Figure B.17: MutationRate comparison for generation number

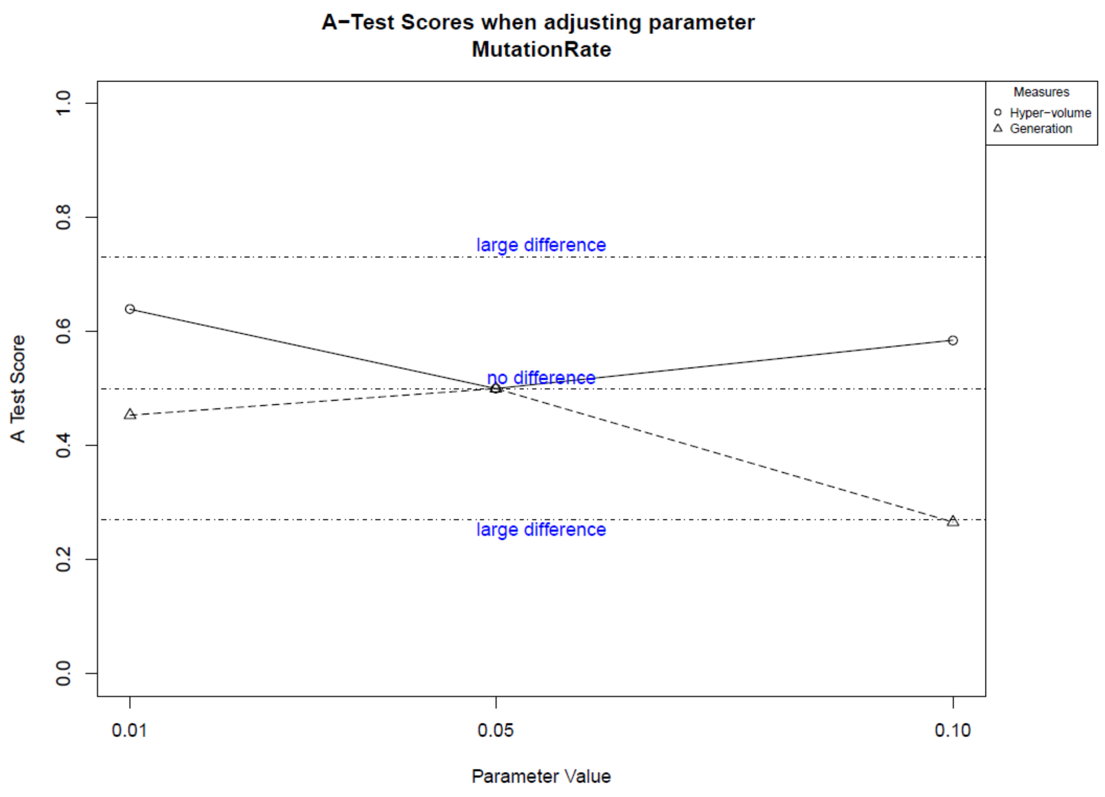

Figure B.18: A-test score for MutationRate comparison

As are shown in these MutationRate comparison results, the baseline value of 0.05 achieves the best experiment performance compared with the others. The value of 0.01 and 0.1 both result in lower hyper-volumes with more generations to make evolution converged compared to the performance of baseline value. In addition, the value of 0.1 also reaches the large difference area in terms of generation number, which means it costs much more generations to obtain a lower hyper-volume compared to baseline value. Although the value of 0.01 has no large difference in terms of both 2 responses compared with baseline value, the Vargha-Delaney A-test still demonstrates that the baseline value is currently the best option for MutationRate. In conclusion, the baseline value achieves better experiment performance than the other options. So there is no need to change the current value of MutationRate.

- RecurrentConnectionProbability

Finally, Figure B.19 shows the RecurrentConnectionProbability comparison for hyper-volume and Figure B.20 shows that for generation number. Both of 2 comparisons perturbed the value of RecurrentConnectionProbability with 0, 0.1, 0.3 and 0.5. It needs to note that the value of 0 actually doesn't trigger any mutation to create recurrent connections in the controller. In this sense, this neutral network controller becomes a feed forward controller without any recurrent connections. In other words, this RecurrentConnectionProbability comparison can be considered as a comparison between feed forward controller and recurrent controllers with different recurrent connection probabilities. Figure B.21 illustrates the final Vargha-Delaney A-test comparison for the perturbed values.

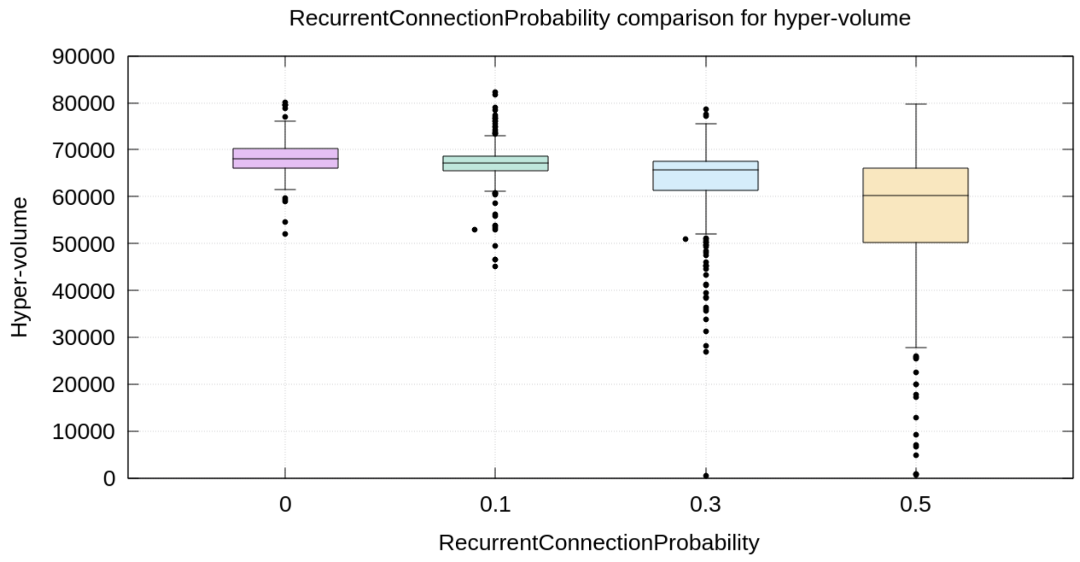

Figure B.19: RecurrentConnectionProbability comparison for hyper-volume

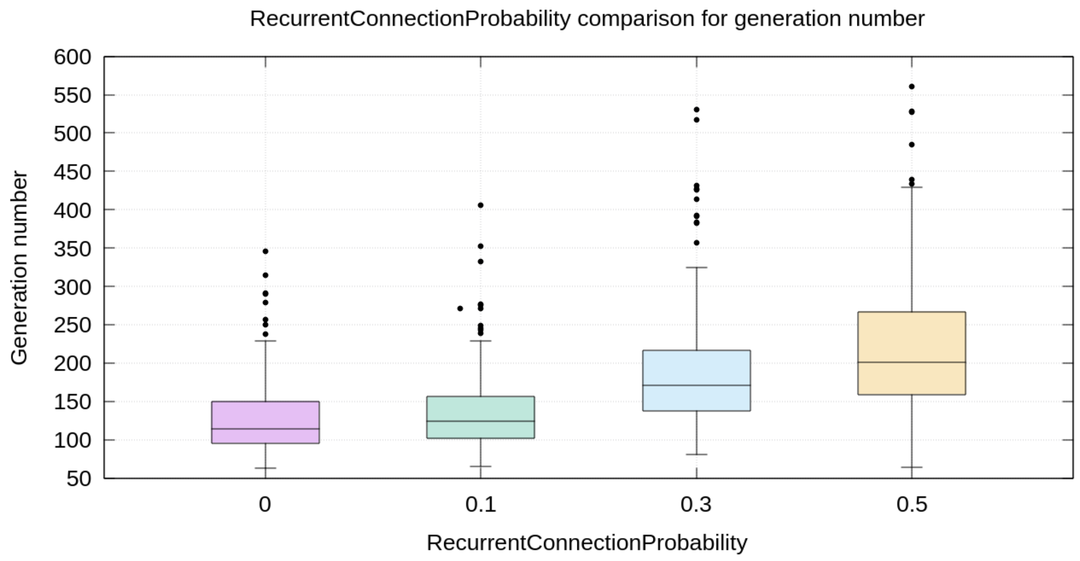

Figure B.20: RecurrentConnectionProbability comparison for generation number

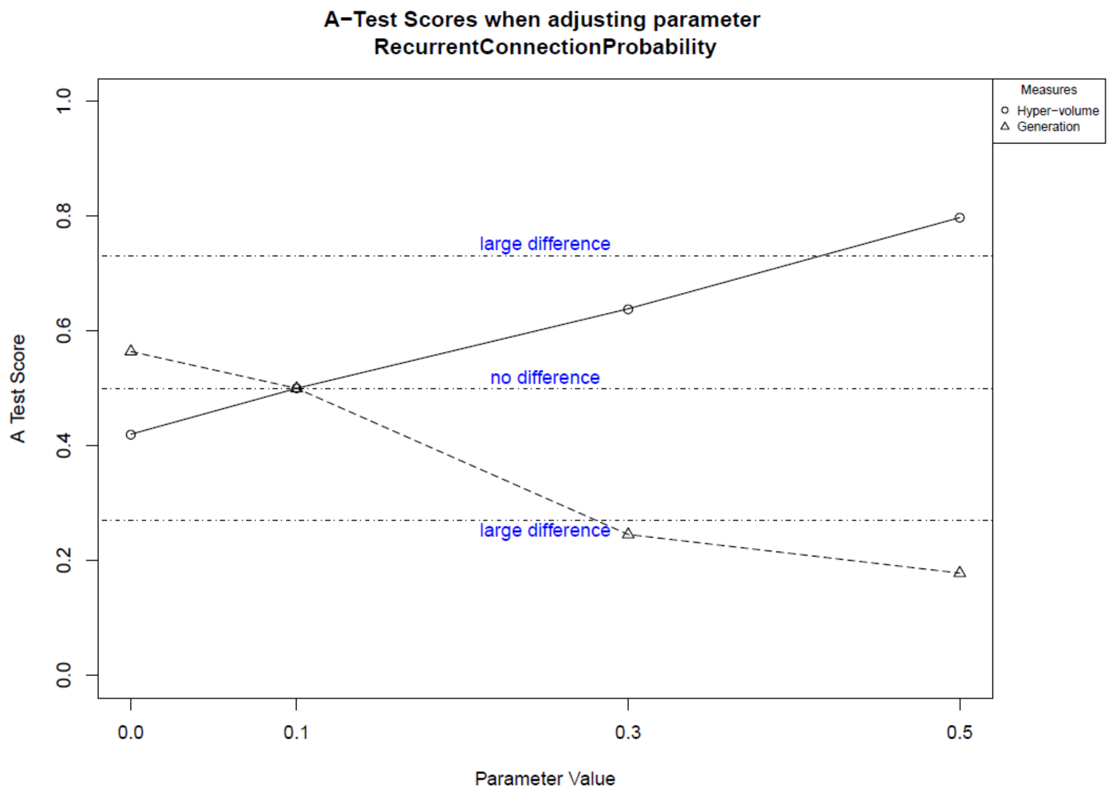

Figure B.21: A-test score for RecurrentConnectionProbability comparison

These comparisons of RecurrentConnectionProbability indicate an unexpected result for the parameter value selection. From the value 0 to 0.5, the hyper-volumes drop continually with a large difference of 0.5 finally. At the same time, the generation numbers also keep increasing from value 0 to 0.5 and the large difference already occurs from 0.3. Generally speaking, the experiment performance is declining with the increase of RecurrentConnectionProbability. Although there is no large difference between the performance of 0 and 0.1, those results still indicate that the value of 0 achieves a better experiment performance rather than that of 0.1. From the analysis, it seems that a feed forward neural network controller performs better than any other recurrent neural network controllers with different recurrent connection probabilities. Although the recurrent connection probability demonstrates advantages to evolve recurrent neural networks in [30], this work doesn't benefit any more from the recurrent neural network controllers. In conclusion, the value of RecurrentConnectionProbability needs to be set 0 instead of 0.1 for a better experiment performance. That is to say, a feed forward neural network controller is currently best suited to this work.

In summary, Nconv and Rconv both have impact on the experiment performance compared with the baseline values. On the other hand, WeightRange is the only parameter which could also result in different experiment performance compared with its baseline value. In addition to WeightRange, RecurrentConnectionProbability also affects the experiment performance. However the best value of it, which is 0, has already been found based on the sensitivity analysis of the currently selected parameter values. In this sense, NumNode, NodeArity and MutationRate will keep the baseline values. RecurrentConnectionProbability will be the new value found based on the sensitivity analysis. Apart from that, a further investigation needs to be conducted to see what values of Nconv, Rconv and WeightRange will be most suited to the experiment performance, which will be discussed in the next section B.2.

B.2 Further investigation on the sensitivity analysis of MOCGPANN parameters

This section describes a further investigation based on the previous sensitivity analysis results by the Parameter Robustness technique in Spartan. As mentioned in the previous section, Nconv, Rconv and WeightRange all have more or less influence on the experiment performance rather than their baseline values. In this case, more parameter value options were selected to do the evolution experiment again for each of these 3 parameters respectively and the Parameter Robustness technique was also utilised to check whether the new selected values were more suited to the experiment performance.

- Nconv

Figure B.22 shows a further Nconv comparison for hyper-volume and Figure B.23 shows that for generation number. The value of Nconv was further perturbed with smaller values including 5, 10 and 15 in addition to 20, 30 and 40. Figure B.24 demonstrates the Vargha-Delaney A-test comparison including those further perturbed values.

As can be seen from the results, the hyper-volume decreases with the decline of Nconv from 20 and the large difference appears when the value reaches 5. On the other hand, generation number drops straight from 40 to 5 with the large difference appeared around 15. In this sense, the most suited value would be 20 for Nconv since the hyper-volumes of 20, 30 and 40 are quite similar but the generation number of 20 achieves the least. Another option is to select 15. Although its hyper-volume is lower than that of 20, 30 and 40, it is still not far from the no difference criteria but with a much less generation number, which has already reached the large difference area. In conclusion, the value of Nconv could be set 20 rather than the baseline value 30. Although the value of 15 may be another option, the value of 20 may guarantee a better performance due to 5 more generations for the convergence observation.

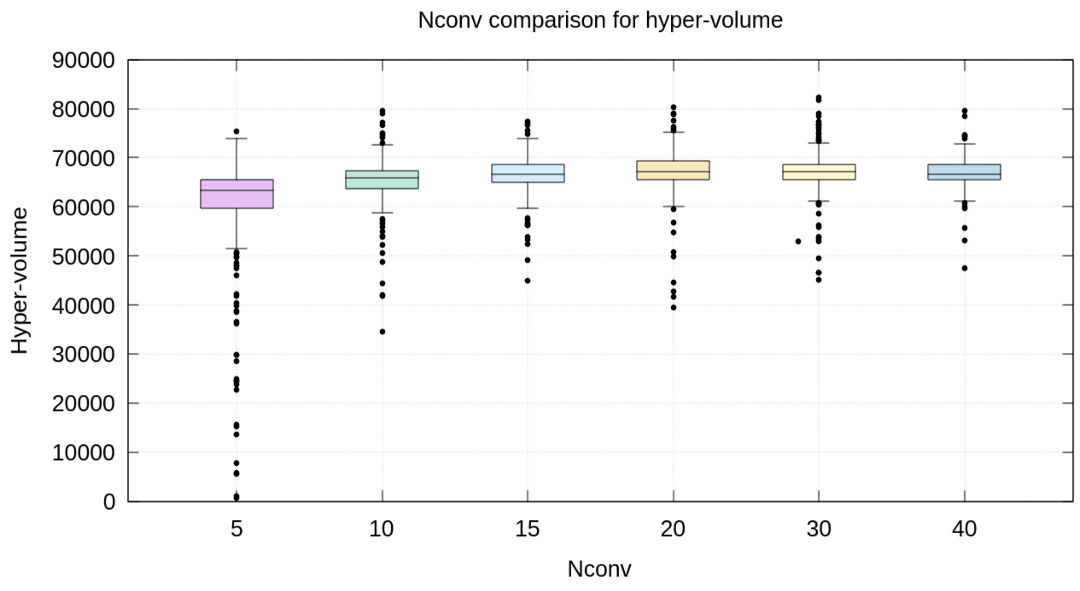

Figure B.22: Nconv further comparison for hyper-volume

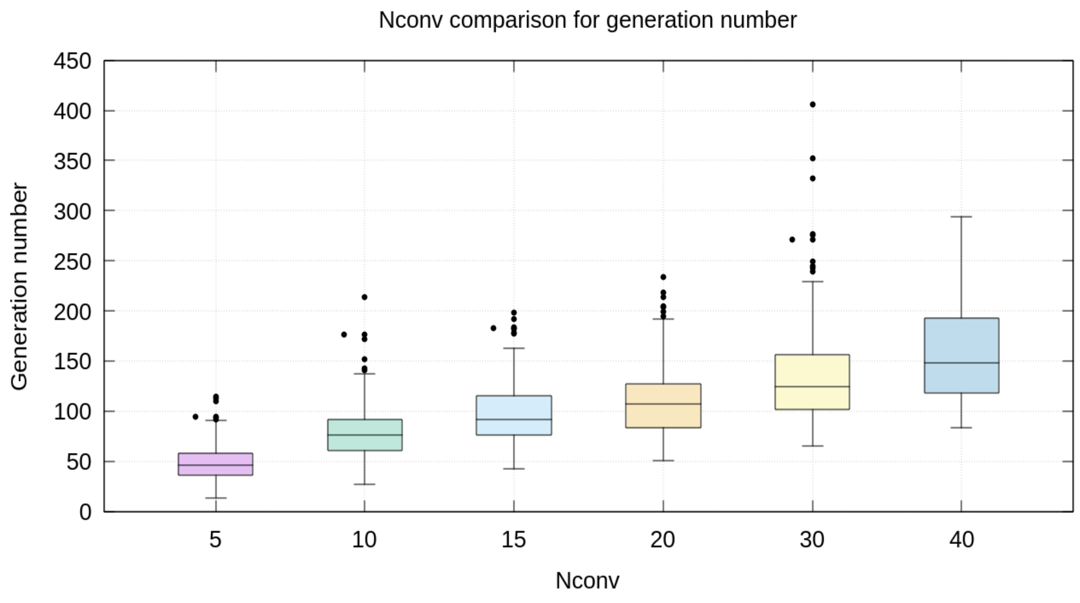

Figure B.23: Nconv further comparison for generation number

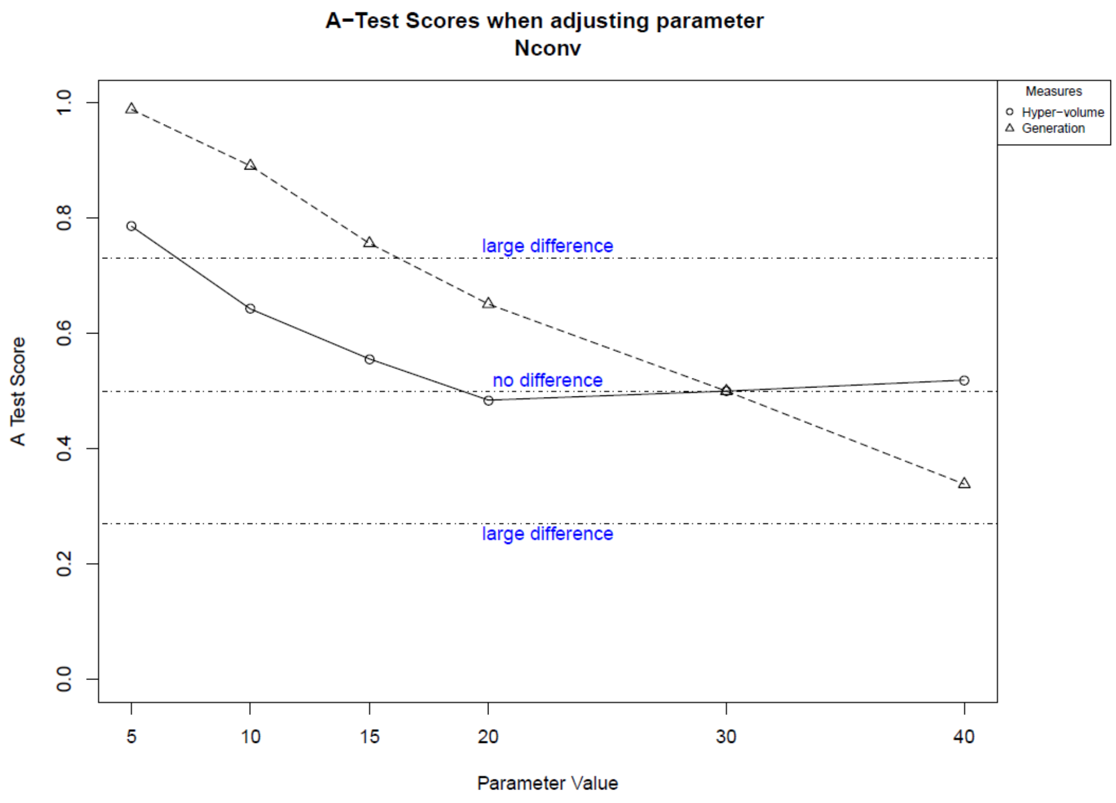

Figure B.24: A-test score for Nconv further comparison

- Rconv

Figure B.25 shows the Rconv further comparison for hyper-volume and Figure B.26 shows that for generation number. Both of these two further comparisons set slightly larger values for Rconv, which were 1.05 and 1.1 in addition to the previously perturbed values 1.0001, 1.001 and 1.01. Figure B.27 indicates the Vargha-Delaney A-test comparison for this further comparison.

According to the further comparison results for Rconv, the hyper-volumes decrease all the way from 1.0001 to 1.1 with a straight decline of generation number at the same time. The large difference appears around 1.1 for hyper-volume but it appears around 1.01 for generation number. Generally speaking, it is common that the hyper-volume will decrease with the decline of generation number when Rconv becomes larger since larger Rconv could result it earlier convergence with degraded performance. However 1.01 is still the most suited value for this further comparison since the hyper-volume of 1.01 has no obvious difference with that from baseline value but the generation number drops significantly. As a contrary, 1.05 has less generation number than 1.01

but the hyper-volume is far from the no difference criteria to some extent, which is not beneficial to experiment performance. In conclusion, the most suited value of Rconv could be 1.01 instead of 1.001.

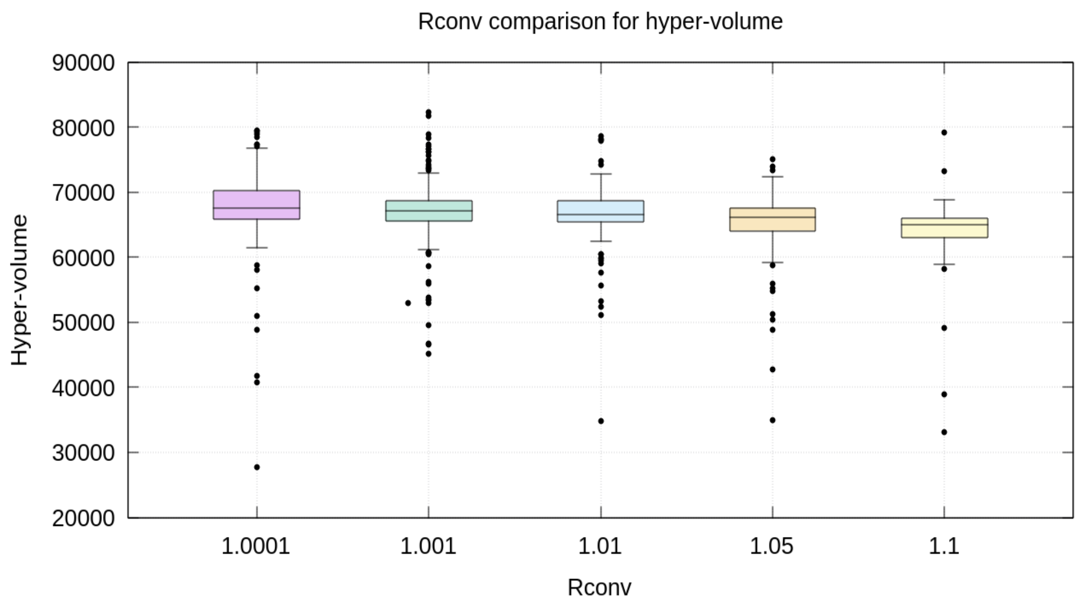

Figure B.25: Rconv further comparison for hyper-volume

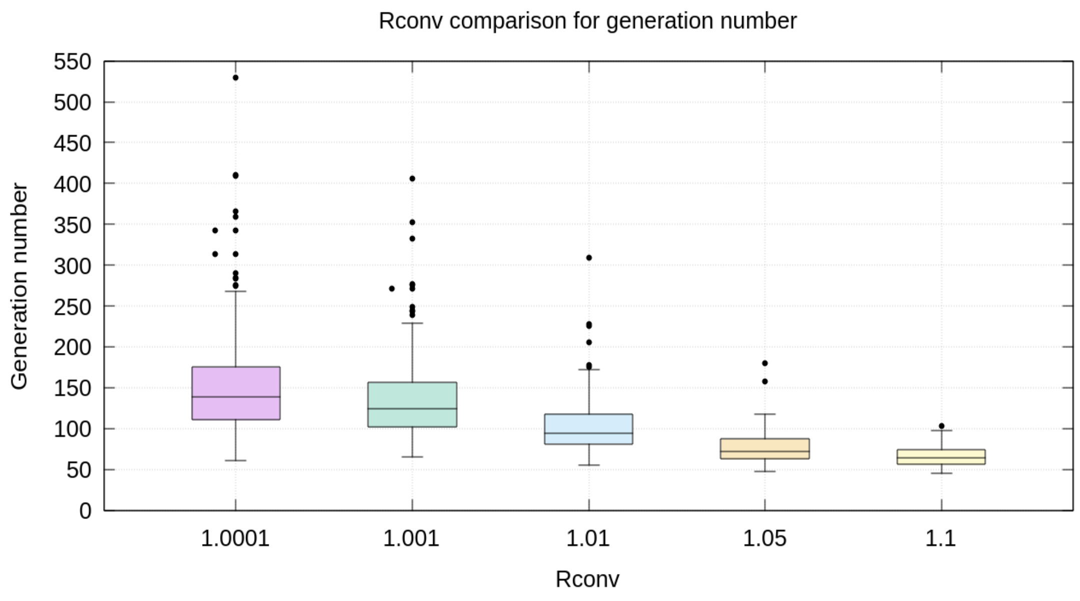

Figure B.26: Rconv further comparison for generation number

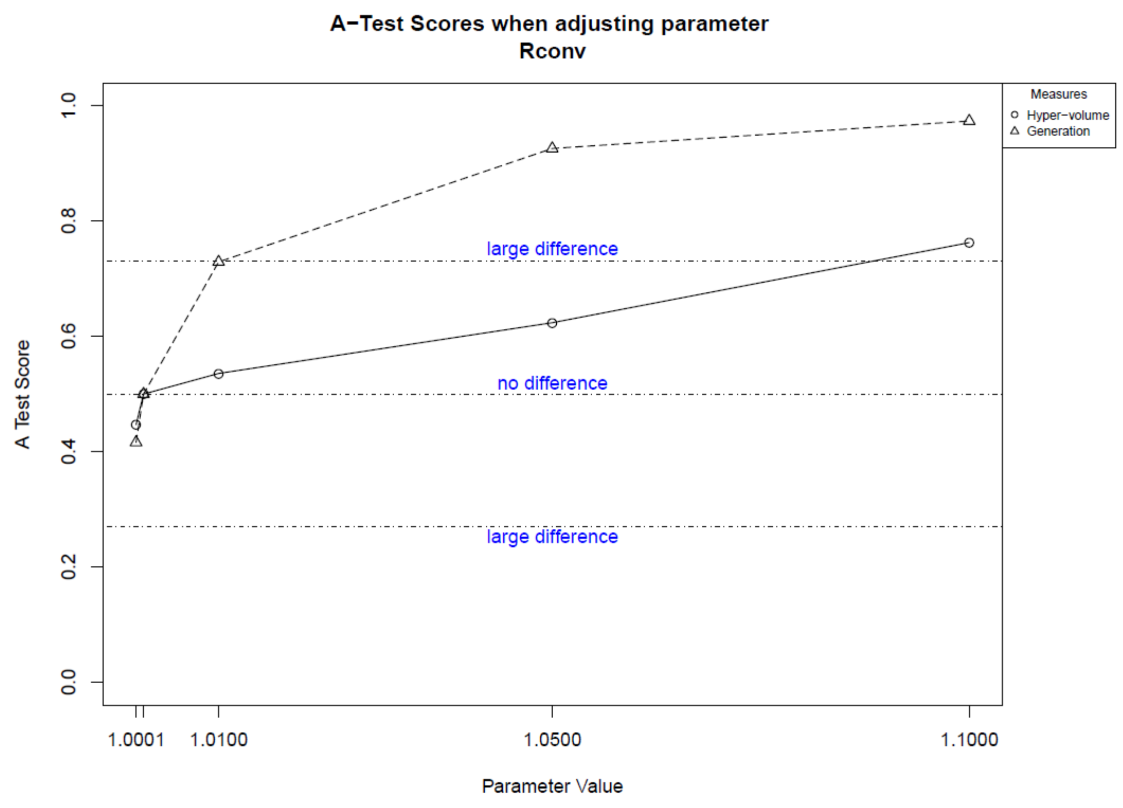

Figure B.27: A-test score for Rconv further comparison

- WeightRange

In terms of the WeightRange, Figure B.28 shows the further comparison result for hyper-volume and Figure B.29 shows that for generation number. In this further comparison, WeightRange was set more extensive values like +/-15 and +/20 besides the previously set values +/3, +/5 and +/-10. Finally, Figure B.30 illustrates the A-test score for the WeightRange further comparisons.

As can be seen from the WeightRange further comparison results, the hyper-volumes of +/-10, +/-15 and +/-20 have actually no obvious difference compared to that of the baseline value +/-5. However all of the generation numbers of +/-10, +/-15 and +/-20 are less than that of the baseline value with some difference to some extent. An interesting point is that the experiment performances of +/-10, +/-15 and +/-20 are actually quite similar not only in hyper-volume but also in generation number. A possible explanation is that the WeightRange of +/-10 has already made the experiment performance saturated, so there is nearly no improvement when setting even larger values. In conclusion, +/-10 can be utilised as the currently best suited value for WeightRange instead of the baseline value +/-5.

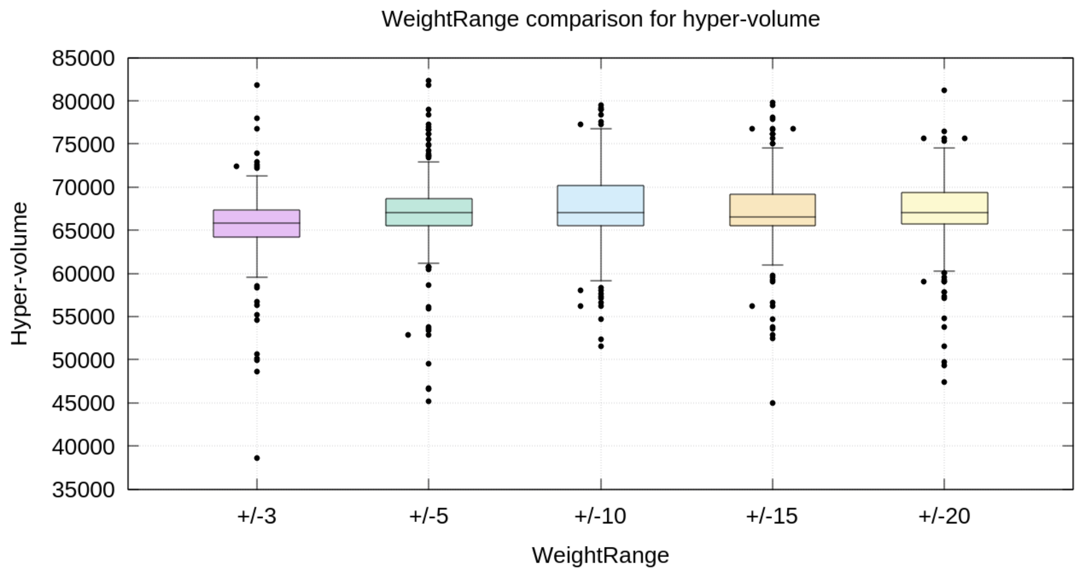

Figure B.28: WeightRange further comparison for hyper-volume

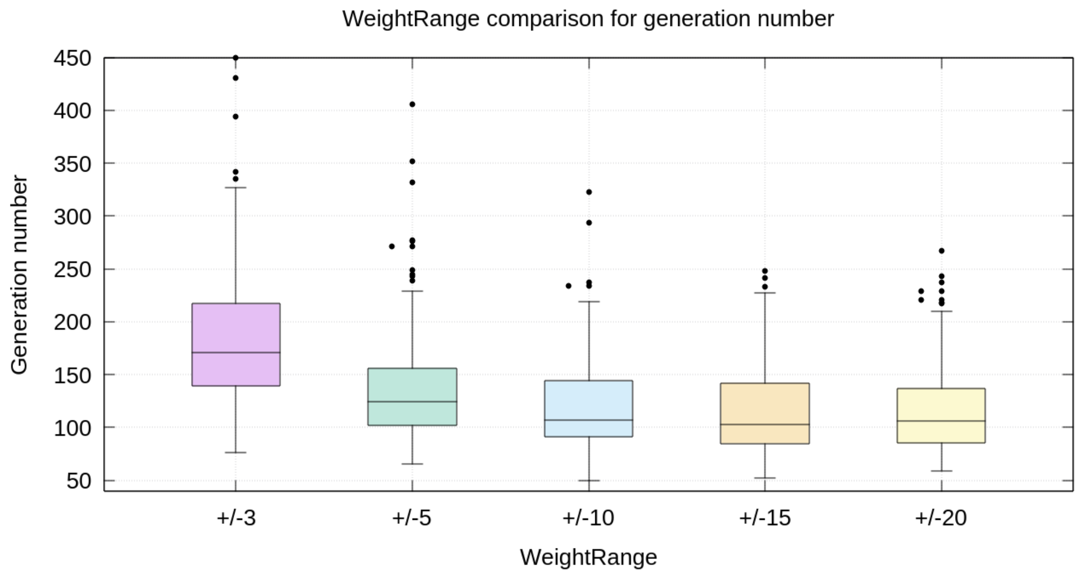

Figure B.29: WeightRange further comparison for generation number

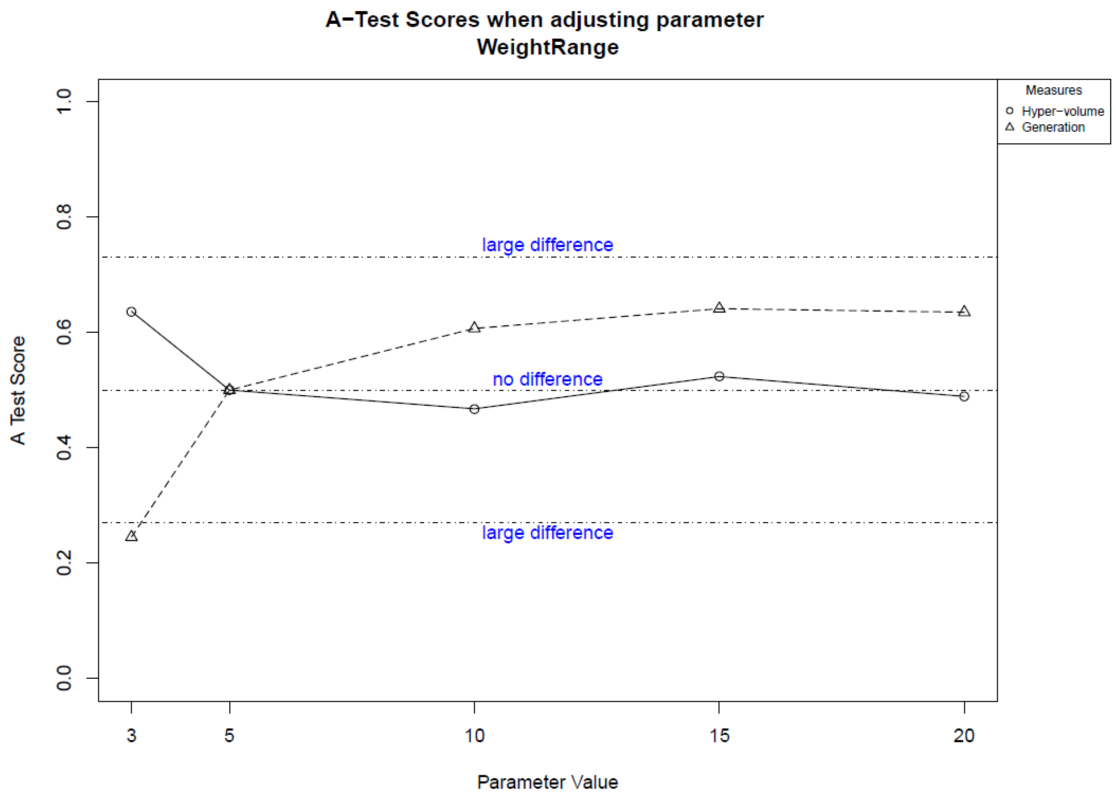

Figure B.30: A-test score for WeightRange further comparison

In summary, the new value of Nconv can be set 20 with 1.01 as the new value for Rconv. +/-10 can be used as a new WeightRange value. In addition, RecurrentConnectionProbability will be set 0 instead based on the sensitivity analysis in the previous section. However it needs to note that this kind of Parameter Robustness technique utilises a one at a time approach to tune these parameters. So each parameter is tuned with the same values remained for other parameters. In this case, the best parameter value may be just suited to the situation where the other parameters still keep their baseline values. So it is still not sure whether 2 or more parameters with the new values still perform better than that with baseline parameter values. To investigate this problem, the parameters could be tuned in two groups which are convergence parameters and CGP parameters. The convergence parameters refer to Nconv and Rconv, which are used to set a suited convergence criterion for observing the performance. And CGP parameters are related to optimization algorithm itself which include WeightRange and RecurrentConnectionProbability to be investigated based on the Parameter Robustness technique results. In this case, convergence parameters will be perturbed firstly with the calibration CGP parameter values from the above results of the further sensitivity analysis. And then CGP parameters will be perturbed with the convergence parameter calibration values. Both of these two parameter perturbations will be compared with the baseline parameter values which were set initially.

- Nconv with Rconv

In terms of Nconv and Rconv, the parameter values will be perturbed by 20 with 1.01; 20 with 1.001; 30 with 1.01 and 30 with 1.001. WeightRange and RecurrentConnectionProbability are kept for their calibration values which are +/-10 and 0. All the other CGP parameters including NumNode, NodeArity and MutationRate are fixed with the baseline values. Finally they will be compared with the baseline parameter values, which were set at the beginning of the experiment, in terms of hyper-volume and generation number. However the Parameter Robustness technique in Spartan is not available for printing the graph based on the A-test score for this kind of group parameter comparison. So the A-test score will only be listed in table rather than graph and the Mann-Whitney U-test score will also be utilised in addition to the A-test for a more meaningful comparison. Figure B.31 shows the hyper-volume comparisons for these four perturbed Nconv and Rconv combinations with the baseline values. Figure B.32 shows the generation number comparisons for those Table B.1 displays the U-test scores of the comparisons for the perturbed convergence parameters and Table B.2 displays the A-test scores for them.

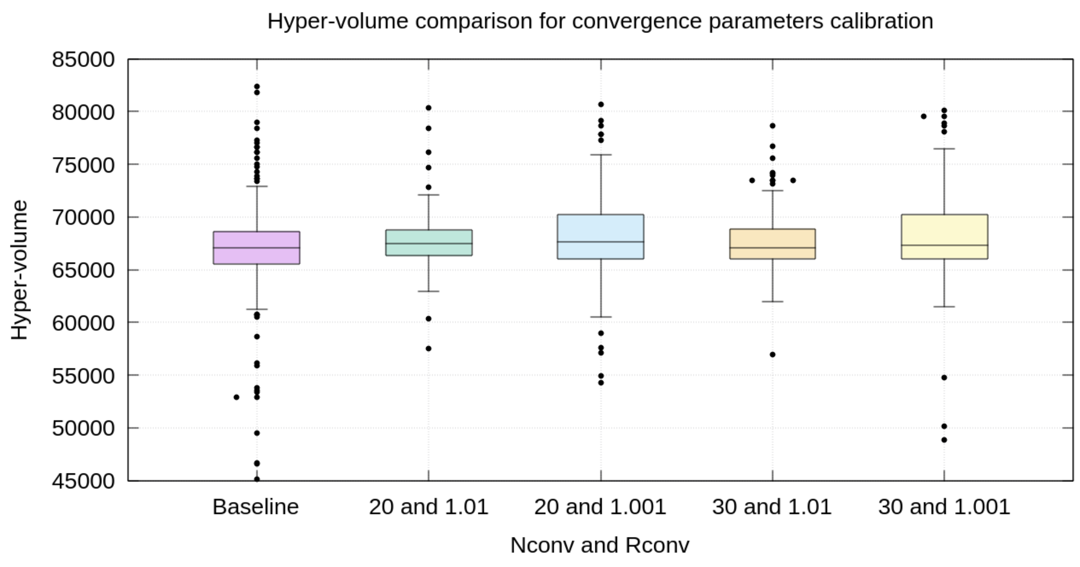

Figure B.31: Hyper-volume comparison for convergence parameter calibration

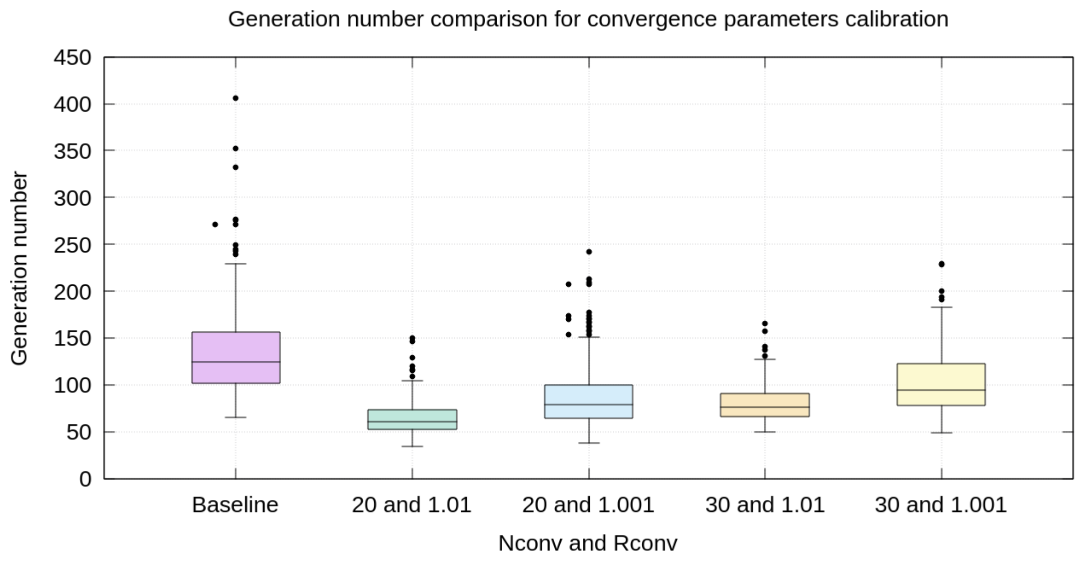

Figure B.32: Generation number comparison for convergence parameter calibration

Table B.1: U-test scores of the comparisons for convergence parameter calibration

	Baseline	20 & 1.01	20 & 1.001	30 & 1.01	30 & 1.001
Hyper-volume	1	0.11876	0.00288	0.4965	0.03156
Generation number	1	<0.00001	<0.00001	<0.00001	<0.00001

Table B.2: A-test scores of the comparisons for convergence parameter calibration

	Baseline	20 & 1.01	20 & 1.001	30 & 1.01	30 & 1.001
Hyper-volume	0.5	0.455692	0.421474	0.481269	0.437491
Generation number	0.5	0.954808	0.832945	0.895212	0.719441

As can be seen from Table B.1, 20 with 1.001 and 30 with 1.001 demonstrate significant difference compared to the baseline value response in terms of hyper-volume, where the p values are below 0.05. However the generation number responses for all of the four combinations achieve significant difference, where the p values are all below 0.00001. In this case, it is necessary to conduct an A-test to further investigate whether they have different effect sizes compared to the baseline value response and that's why A-test based on effect size is required. As can be seen from Table B.2, all of the parameter combinations have similar but better hyper-volume responses compared to the baseline value, where the A-test scores are all below 0.5. However they all obtain small or even no effect sizes between 0 and 0.14, where the corresponding scores are between 0.5 and 0.36. This consequence indicates that although 20 with 1.001 and 30 with 1.001 have significant difference compared to the baseline value response, they both actually have small effect. However in terms of generation number, all of them present more or less effects compared to the baseline value. Among them, the combination of 20 with 1.01 achieves the least generation number to make evolution converged, whose A-test score is the largest. Another option is 30 with 1.01 which achieves the second least generation number. However to be more specific, 20 with 1.01 has slightly larger hyper-volume response than that of 30 with 1.01 but the generation number of 20 with 1.01 is also less than 30 with 1.01. In other words, 20 with 1.01 spend fewer generations to get a better performance compared to 30 with 1.01. The third place in generation number response is 20 with 1.001. It achieves a better hyper-volume performance but with a larger generation number. This phenomenon also implies that a higher resolution Rconv value could result in more generations but with better performance. So it depends on which response is more important from the perspective of experiment designer: a faster convergence with a worse performance or a better performance with a slower convergence. 30 with 1.001 ranks the final place in generation number but still obtains a similar hyper-volume compared to 20 with 1.001, which could be eliminated. As a consequence, 20 with 1.01 could be currently considered as the first choice for Nconv and Rconv. In addition, 20 with 1.001 would be also an alternative choice since it achieves a better hyper-volume response than 20 with 1.01 but a worse generation number response. In conclusion, 20 with 1.01 would be currently a first combination for Nconv and Rconv with 20 with 1.001 as an alternative choice depending on which response is more relevant to the experiment designer. Both of them were investigated coupled with the calibration values for WeightRange and RecurrentConnectionProbability, which are +/-10 and 0 respectively.

- WeightRange with RecurrentConnectionProbability

In terms of the CGP parameters, WeightRange and RecurrentConnectionProbability are perturbed with +/-10 with 0; +/-10 with 0.1; +/-5 with 0 and +/-5 with 0.1. All the other CGP parameters including NumNode, NodeArity and MutationRate are fixed with the baseline values. Moreover, Nconv and Rconv are kept with 20 and 1.01 as calibration values for this comparison. Figure B.33 shows the hyper-volume comparison result for perturbed WeightRange and RecurrentConnectionProbability. Figure B.34 shows that for generation number. Table B.3 displays the U-test scores for these comparisons and Table B.4 displays the A-test for them.

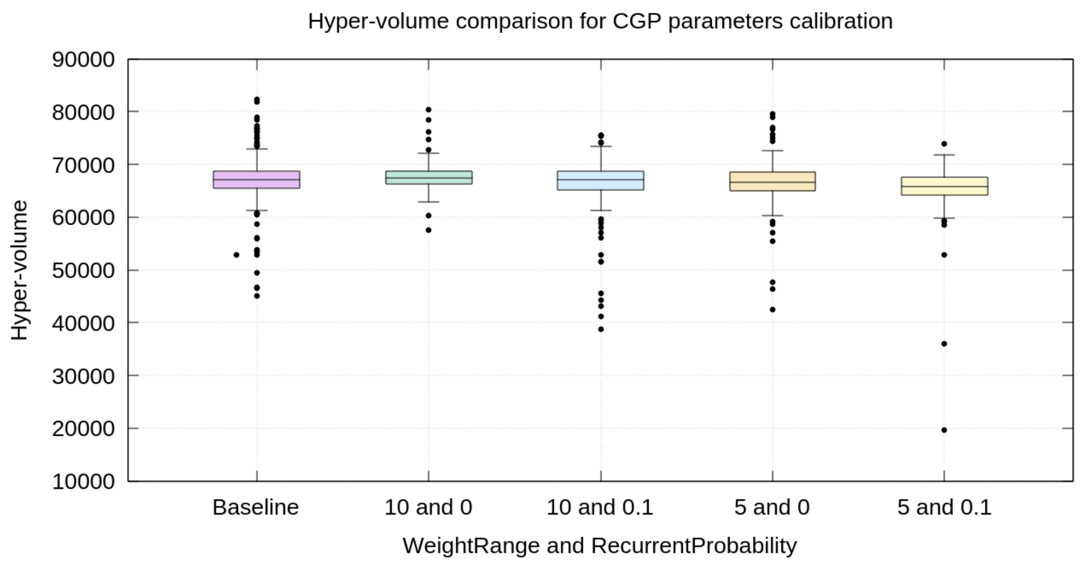

Figure B.33: Hyper-volume comparison for CGP parameters calibration

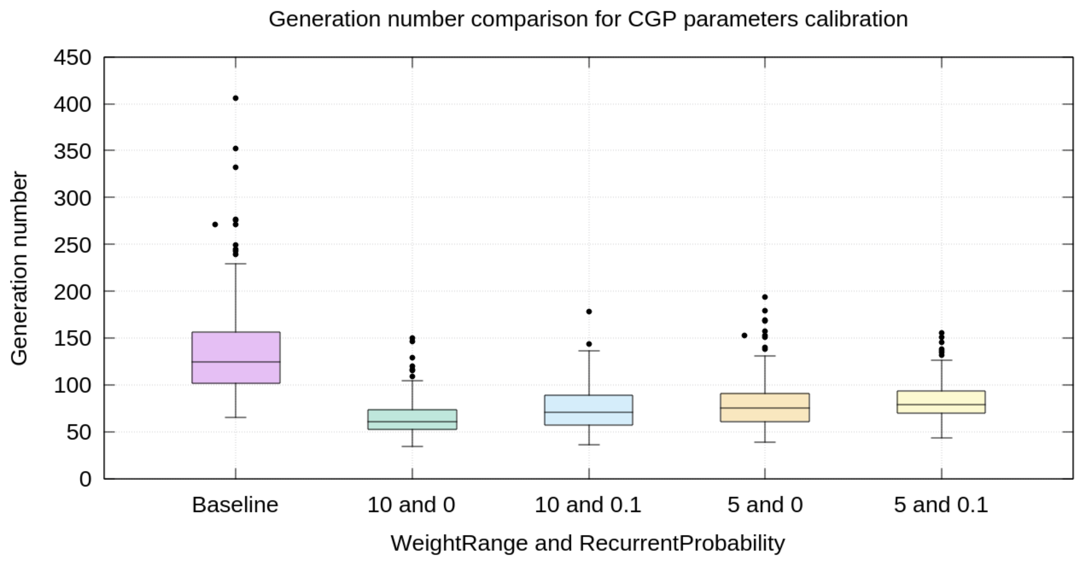

Figure B.34: Generation number comparison for CGP parameters calibration

Table B.3: U-test scores of the comparisons for CGP parameter calibration

	Baseline	+/-10 & 0	+/-10 & 0.1	+/-5 & 0	+/5 & 0.1
Hyper-volume	1	0 .11876	0 .18352	0 .04036	< 0.00001
Generation number	1	< 0.00001	< 0.00001	< 0.00001	< 0.00001

Table B.4: A-test scores of the comparisons for CGP parameter calibration

	Baseline	+/-10 & 0	+/-10 & 0.1	+/-5 & 0	+/5 & 0.1
Hyper-volume	0.5	0.455692	0.5391032	0.556328	0.6432423
Generation number	0.5	0.954808	0.9088182	0.8892483	0.8800448

As can be seen from Table B.3 and Table B.4, the hyper-volume responses for the perturbed parameter values also have no large effect compared to the baseline values. Although +/-5 with 0 and +/-5 with 0.1 both achieve significant difference from the U-test compared to the baseline value response, they actually obtain worse performance than that of the baseline value, where the A-test scores are both above 0.5. However the value of +/-10 with 0 is the only one that outperforms the baseline value in terms of hyper-volume with an A-test score below 0.5 in spite of quite small effect size. In terms of the generation number, although all of them have similar A-test scores with the large effect size, +/-10 with 0 still achieves the least generation number to make evolution converged. That is to say, +/-10 with 0 has got the relatively best experiment performance with the least generation number. In conclusion, +/-10 with 0 is currently the best parameter combination for WeightRange and RecurrentConnectionProbability with Nconv and Rconv remained at 20 and 1.01, which are the calibration values.

Appendix C

C.1 Number of hidden nodes selection

The results are obtained based on five different numbers of hidden nodes including 6, 8, 10, 12 and 14 for the ANN in terms of hyper-volume and generation number responses in order to explore how many numbers of hidden nodes are sufficient. Figure C.1 shows the hyper-volume comparison for different number of hidden nodes and Figure C.2 shows that for generation number comparison. Table C.1 lists the U-test scores for the comparison and Table C.2 lists the corresponding A-test scores.

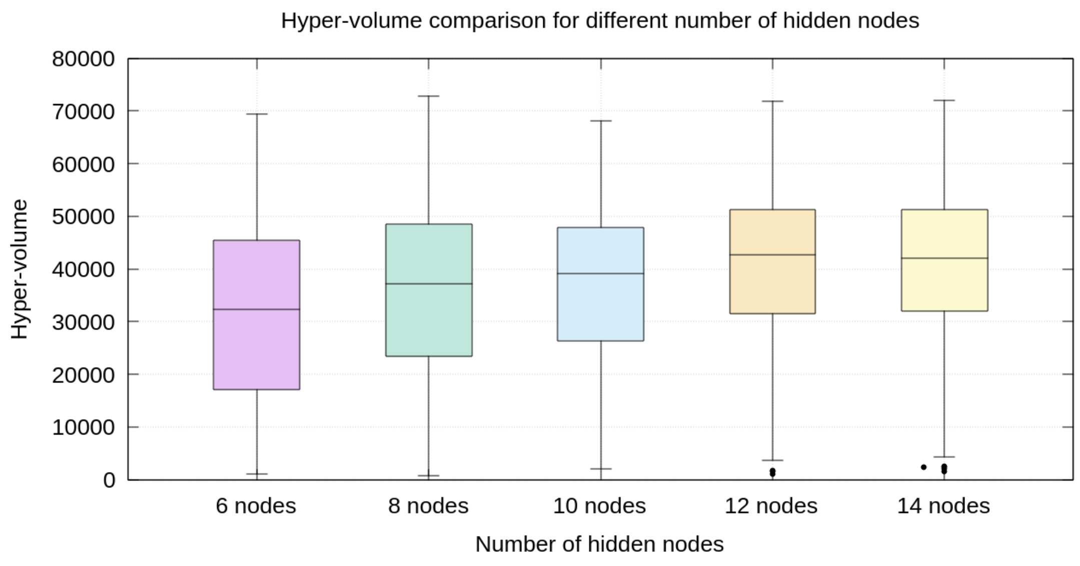

Figure C.1: Hyper-volume comparison for different number of hidden nodes optimized by NSGA2

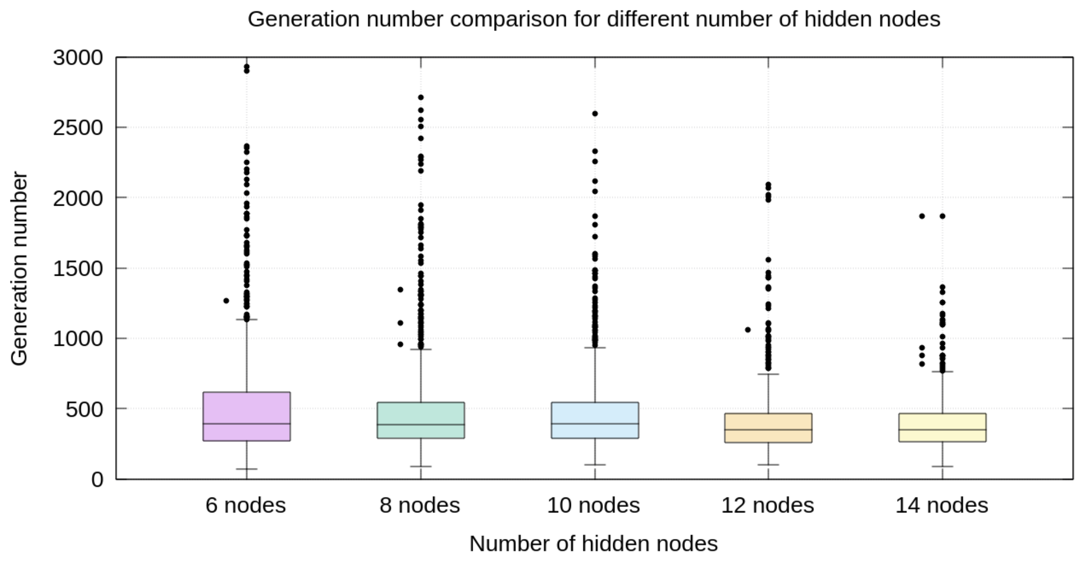

Figure C.2: Generation number comparison for different number of hidden nodes optimized by NSGA2

Table C.1: U-test scores for different number of hidden nodes comparison based on NSGA2

	6 Nodes	8 Nodes	10 Nodes	12 Nodes	14 Nodes
HV	1	< 0.00001	< 0.00001	< 0.00001	< 0.00001
Gen	1	0.55662	0.885292	< 0.00001	< 0.00001

Table C.2: A-test scores for different number of hidden nodes comparison based on NSGA2

	6 Nodes	8 Nodes	10 Nodes	12 Nodes	14 Nodes
HV	0.5	0.42731	0.412712	0.350605	0.348853
Gen	0.5	0.5082	0.501999	0.573866	0.5743

The statistics analysis uses the responses of 6 nodes as the standard data and other responses will be compared with it based on U-test and A-test respectively. As can be seen from Table C.1, apart from the generation number responses for 8 nodes and 10 nodes, other responses all achieve the significant difference compared to 6 nodes with p values < 0.00001%. To assess how much the difference is, Table C.2 lists the corresponding effect size scores. According to Table C.2, the generation number responses for 8 nodes and 10 nodes are both quite close to 0.5, which correspond to the U-test scores in Table C.1 without any significant difference. However the hyper-volume of 8 nodes and 10 nodes are both below 0.5 with scores around 0.42, which lie in the small effect size interval [0.36, 0.44]. That is to say, the performance of 8 nodes and 10 nodes achieve higher hyper-volume with small effect sizes but the generation numbers are quite similar compared to 6 nodes. In addition, the performance of 12 nodes and 14 nodes are both much better than that of 8 nodes and 10 nodes. In terms of the hyper-volume, 12 nodes and 14 nodes achieve much higher responses than that of 8 and 10 nodes with A-test scores around 0.35, which means their effect sizes are already located in the medium interval [0.29, 0.36]. On the other side, their generation number responses are both much less than 8 and 10 nodes with scores around 0.57, which lie in the small effect size interval [0.56, 0.64]. That is to say, 12 and 14 nodes realise higher hyper-volume responses with much less generation numbers than 8 and 10 nodes. In this sense, 12 or 14 nodes in the hidden layer seem a suitable choice for the NSGA2 based ANN's structure. Although more nodes could be tested for the ANN's structure, 14 nodes seem to be already saturated for the hidden layer with quite similar responses with 12 nodes. In this case, 12 nodes or 14 nodes could be both considered as the optimal ANN's structure for connection weight evolution. Therefore, 12 nodes are selected for the ANN's hidden layer and this type of structure will be utilised throughout this chapter for the evolution fault tolerant experiment based on connection weight optimization with NSGA2.

C.2 Sensitivity analysis for NSGA2 parameters

This section shows how these four parameters for NSGA2 are tuned to their optimal values including the crossover probability (PCrossover), mutation probability (PMutation), distribution index for crossover (DICrossover) and mutation (DIMutation) respectively. All the comparison results will be displayed in boxplot and their corresponding A-test scores will also be presented in graphs created by Parameter Robustness technique as following.

- PCrossover

Figure C.3 shows the PCrossover comparison results for hyper-volume and Figure C.4 shows that for generation number. Figure C.5 illustrates the A-test scores in the graph created by Parameter Robustness technique in Spartan.

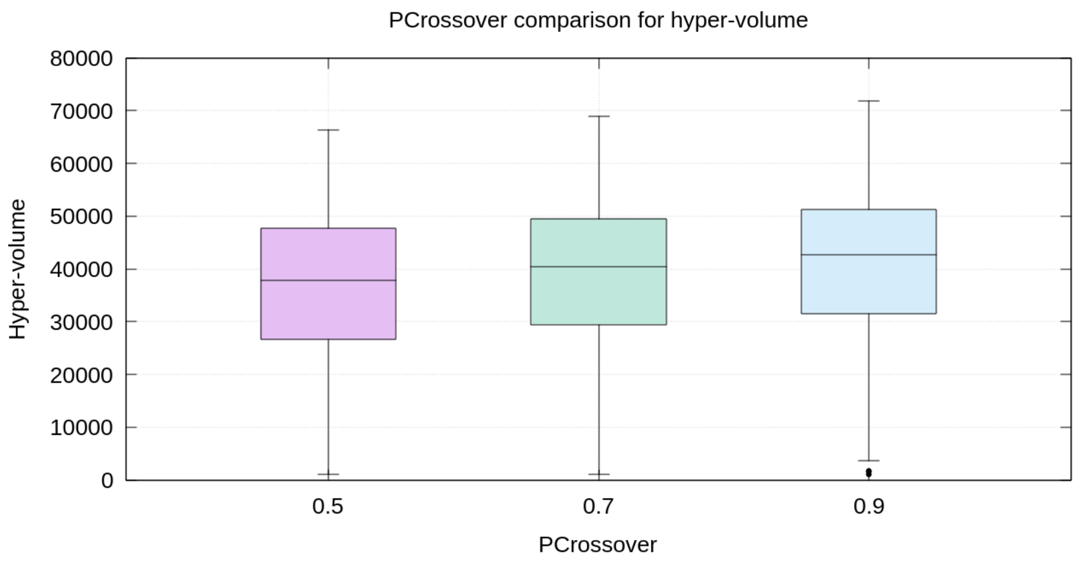

Figure C.3: PCrossover comparison for hyper-volume

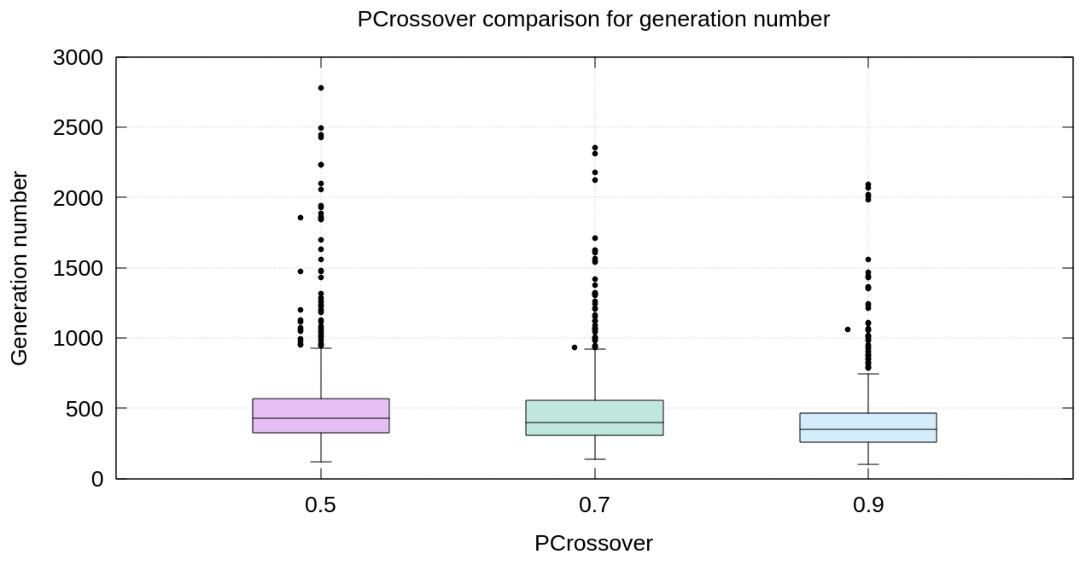

Figure C.4: PCrossover comparison for generation number

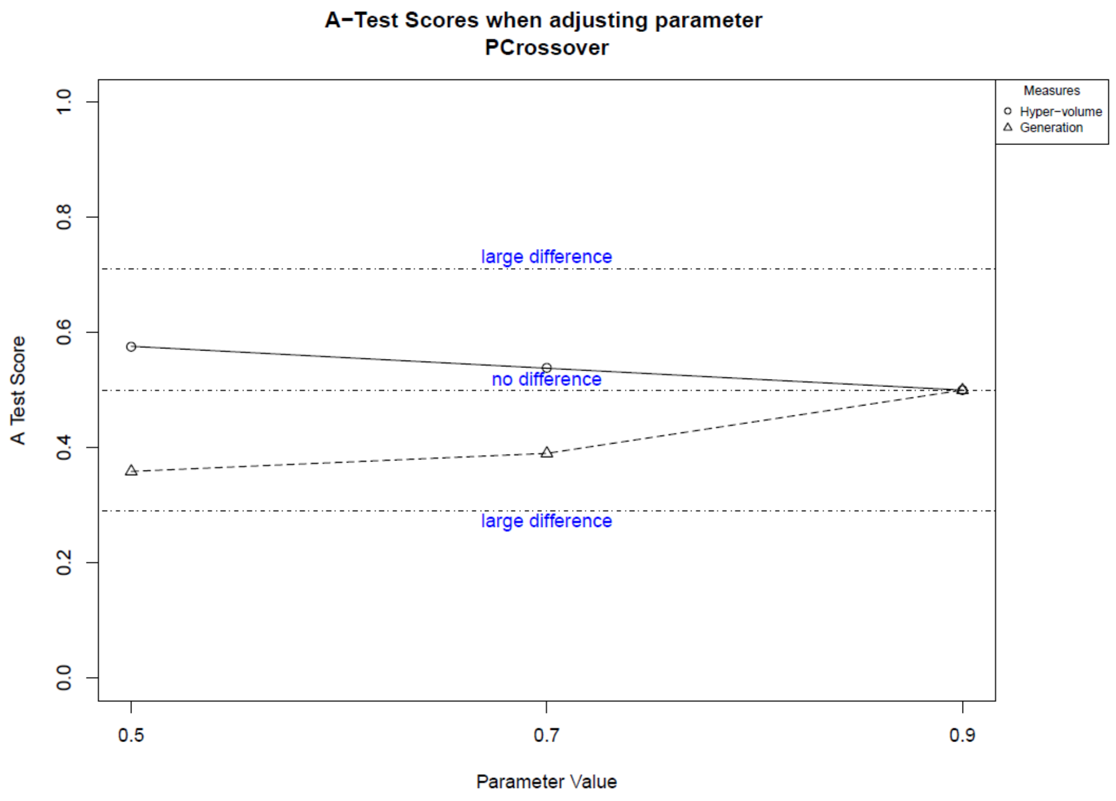

Figure C.5: A-test score for PCrossover comparison

As can be seen Figure C.3 and Figure C.4, the hyper-volume is increasing but the generation number is decreasing with the growth of PCrossover, which indicates that the PCrossover baseline value of 0.9 achieves the best responses with the largest hyper-volume in the least generation number. Figure C.5 also demonstrates the same result in the A-test scores. The A-test score for hyper-volume is decreasing to 0.5 of the baseline value response and the score for generation number is increasing to 0.5 of the baseline value response. That is to say, the hyper-volume is rising and the generation number is declining until they reach the baseline value responses. In conclusion, the baseline PCrossover value of 0.9 achieves the best responses for both of hyper-volume and generation number compared to 0.5 and 0.7.

- PMutation

Figure C.6 shows the PMutation comparison for hyper-volume and Figure C.7 shows that for generation number. Figure C.8 illustrates the A-test scores for these two comparisons.

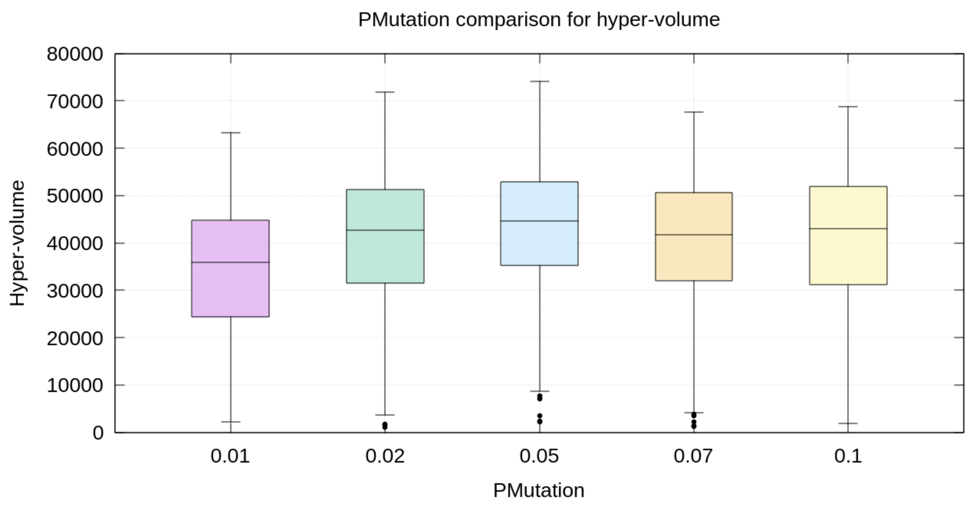

Figure C.6: PMutation comparison for hyper-volume

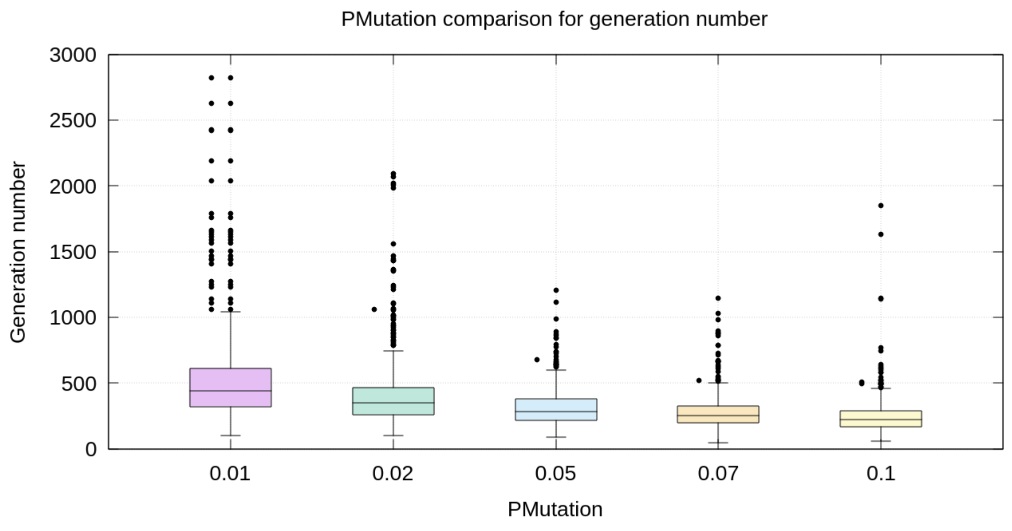

Figure C.7: PMutation comparison for generation number

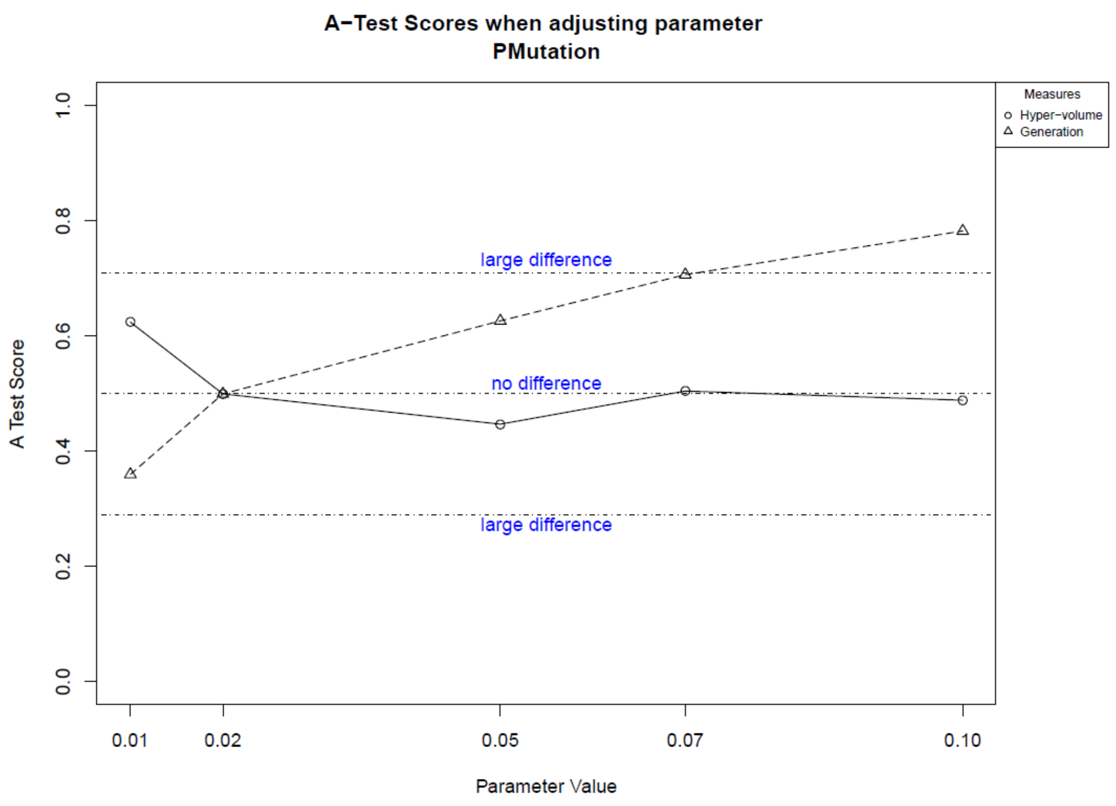

Figure C.8: A-test scores for PMutation comparison

As can be seen from Figure C.6 and Figure C.7, the PMutation value of 0.05 achieves the highest hyper-volume and the value of 0.1 obtains the least generation number. According to Figure C.8, it displays the same result where 0.05 has the lowest A-test score for hyper-volume and 0.1 has the highest A-test score for generation number with large difference compared to the baseline value. To be more specific, only 0.05 achieves the highest hyper-volume although its generation number is not the lowest. 0.1 achieves the lowest generation number but its hyper-volume is quite similar to the baseline response without any obvious difference. In this case, it depends on which aspect the designer is more focused: the hyper-volume or the generation number. If hyper-volume is more relevant for the given task, 0.05 is the most suited value. But if generation number is more relevant, 0.1 would be the most suited one.

However it needs to note that the aim of the NSGA2 based evolution experiment is to compare its performance with MOCGP based results. Due to the huge difference of these two optimisation algorithms, it is meaningless to consider the generation number into the comparison between each other. So only the performance, which is the hyper-volume response, will be taken into account for the comparison between NSGA2 and

MOCGP for the evolution experiment. In this case, the hyper-volume is more focused for this task, so the value of 0.05 will be considered as the most suited value instead of the baseline value of 0.02 for parameter PMutation.

- DICrossover

Figure C.9 shows the DICrossover comparison for hyper-volume and Figure C.10 shows that for generation number. And Figure C.11 illustrates the A-test scores for these two comparisons.

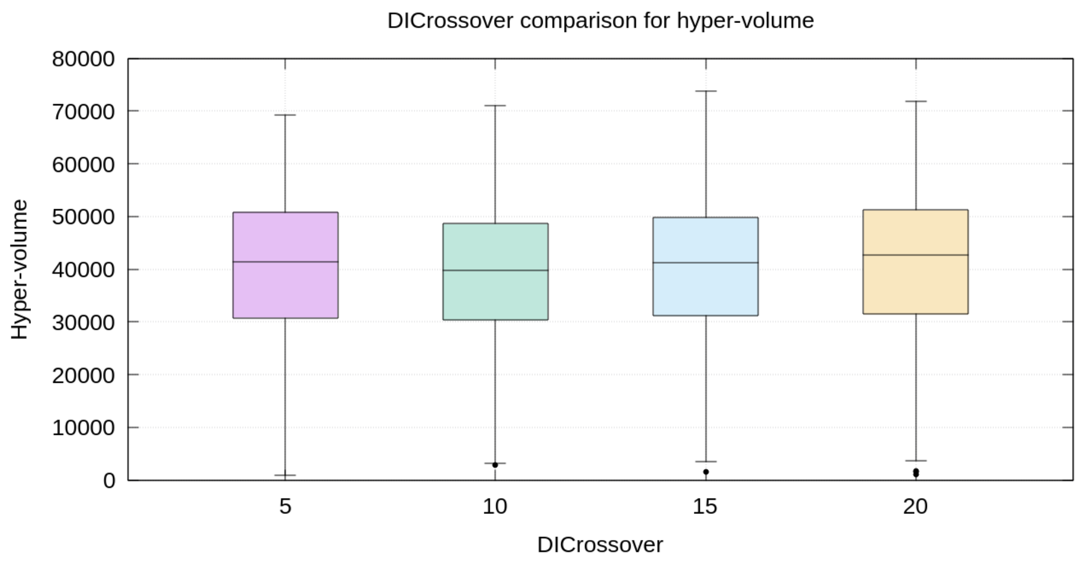

Figure C.9: DICrossover comparison for hyper-volume

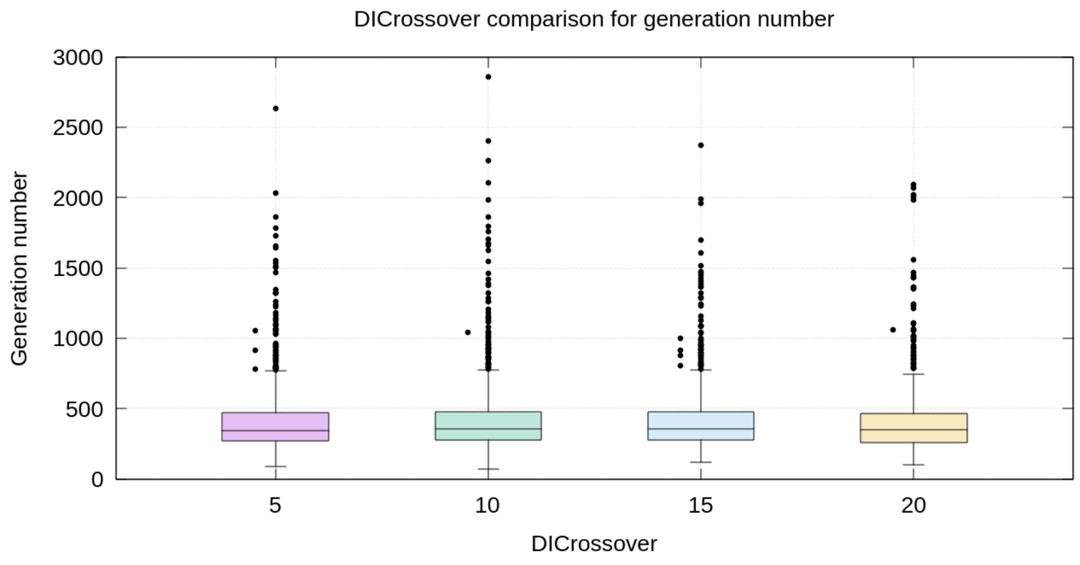

Figure C.10: DICrossover comparison for generation number

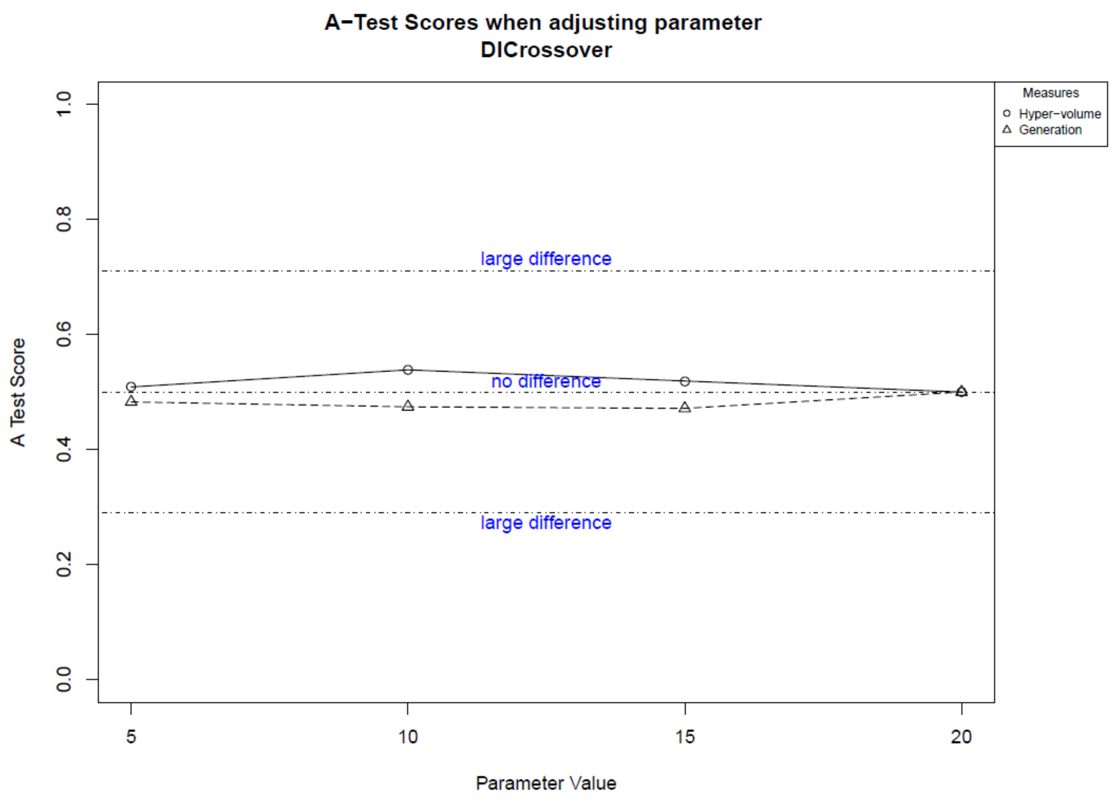

Figure C.11: A-test score for DICrossover comparison

As can be seen from Figure C.9 and Figure C.10, all the four values for DICrossover are quite similar in terms of hyper-volume and generation number responses. Figure C.11 also presents the same result with A-test score that all the other three values actually achieve the similar A-test score with quite small difference compared to the baseline value. To be more specific, the value of 10 obtains the relatively most obvious difference than the others. However its response for hyper-volume is the lowest with the highest value for generation number, which is the worst choice. In conclusion, no best value has been found for parameter DICrossover since all the four options have got quite similar responses for both the hyper-volume and generation number. In this case, the baseline value of 20 could be still utilised for the parameter DICrossover.

- DIMutation

Figure C.12 shows the DIMutation comparison for hyper-volume and Figure C.13 shows that for generation number. Finally Figure C.14 displays the corresponding A-test scores for them.

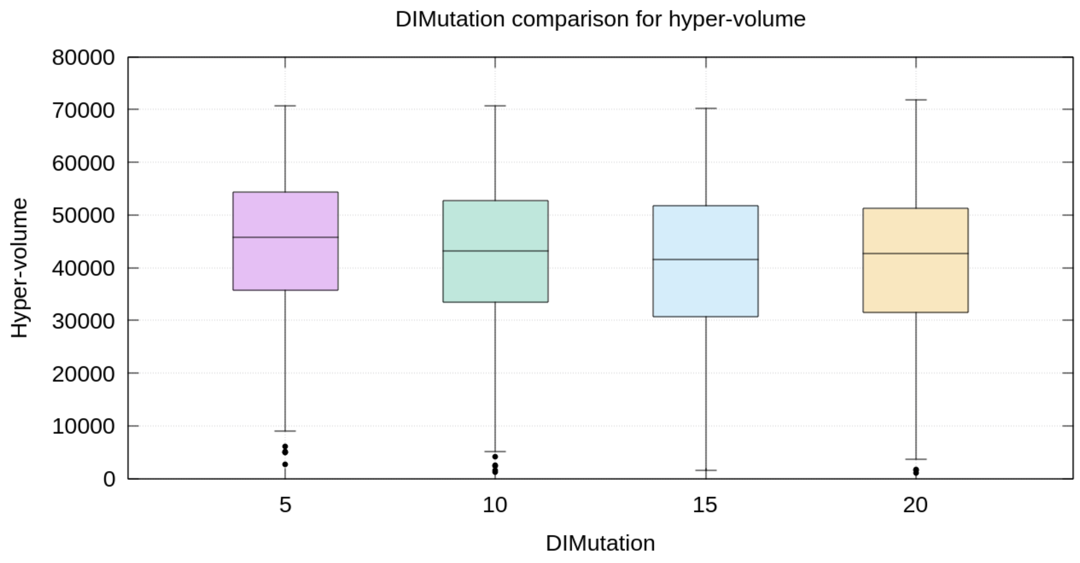

Figure C.12: DIMutation comparison for hyper-volume

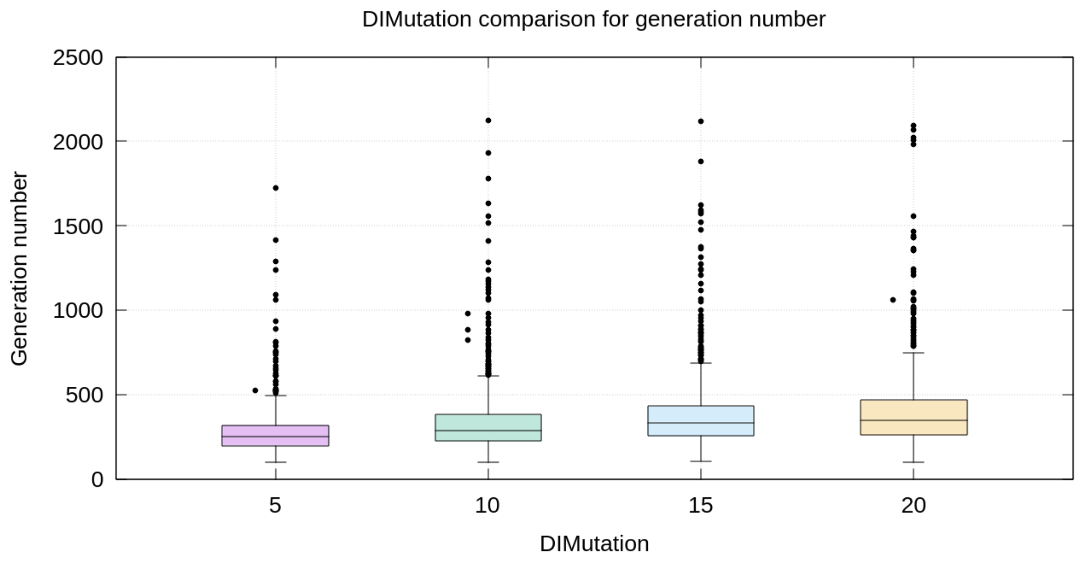

Figure C.13: DIMutation comparison for generation number

As is shown in Figure C.12 and Figure C.13, the hyper-volume response for DIMitation is declining gradually until it reaches the baseline point of 20. On the other hand, the generation number response is growing all the way to the baseline point. This result indicates that the first value of 5 is the most suited value rather than baseline value of 20. According to Figure C.14, the A-test scores also demonstrate the same result that the value of 5 achieves the lowest score for hyper-volume and the highest score for generation number. That is to say, its hyper-volume is the highest and its generation number is the lowest among all the options. In conclusion, the value of 5 is the most suited value for parameter DIMutation instead of the baseline value of 20.

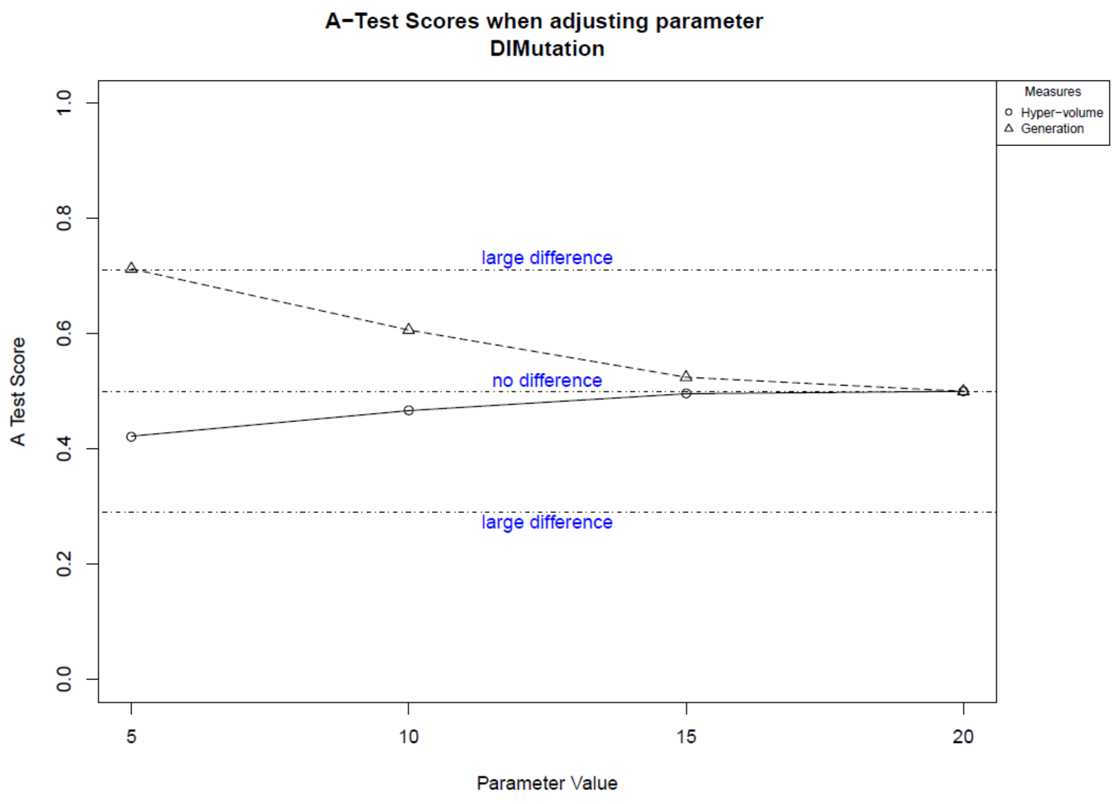

Figure C.14: A-test score for DIMutation comparison

Bibliography

[1] M. Blanke, "What is fault-tolerant control?," *Safeprocess*, 2000.

[2] L. E. G. Castanon and A. V. Martinez, "Artificial Intelligence Methods in Fault Tolerant Control," 2009.

[3] J. Eterno, J. Weiss, D. Looze, and A. Willsky, "Design issues for fault tolerant-restructurable aircraft control," in *1985 24th IEEE Conference on Decision and Control*, 1985, no. December, pp. 900–905.

[4] M. Blanke, M. Staroswiecki, and N. E. Wu, "Concepts and methods in fault-tolerant control," *Proc. 2001 Am. Control Conf. Cat No01CH37148*, vol. 4, no. June, pp. 2606–2620, 2001.

[5] M. Blanke, J. Lunze, M. Kinnaert, M. Staroswiecki, and J. Schröder, *Diagnosis and fault-tolerant control*. 2006.

[6] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annu. Rev. Control*, vol. 32, pp. 229–252, 2008.

[7] J. Lunze and J. Richter, "Control Reconfiguration : Survey of Methods and Open Problems," 2006.

[8] S. Skogestad and I. Postlethwaite, *Multivariable feedback control: analysis and design*, vol. 21. 2005.

[9] A. Numsomran, K. Witheephanich, V. Tipsuwanporn, and N. Klinsmitth, "Robust controller design for plant uncertainty," *2006 SICE-ICASE Int. Jt. Conf.*, pp. 109–113, 2006.

[10] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. 1996.

[11] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, vol. 12, no. 2. 2003.

[12] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2013.

[13] P. Schroder, a. J. Chipperfield, P. J. Fleming, and N. Grum, “Fault tolerant control of active magnetic bearings,” *IEEE Int. Symp. Ind. Electron. Proceedings. ISIE’98 (Cat. No.98TH8357)*, vol. 2, pp. 573–578, 1998.

[14] M. Blanke, S. A. B, and C. P. Lunau, “Fault-tolerant control systems-- A holistic view,” *Control Eng. Pract.*, vol. 5, no. 5, pp. 693–702, 1997.

[15] J. R. Koza, M. a. Keane, J. Yu, F. H. Bennett, and W. Mydlowec, “Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming,” *Genet. Program. Evolvable Mach.*, vol. 1, no. 1, pp. 121–64, 2000.

[16] J. Koza, “Genetic programming: on the programming of computers by natural selection.,” *Cambridge,MA:MITPress*, 1992.

[17] I. Sekaj and J. Perkacz, “Genetic programming - based controller design,” *2007 IEEE Congr. Evol. Comput.*, pp. 1339–1343, 2007.

[18] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, “Genetic programming III: darwinian invention and problem solving,” *IEEE Trans. Evol. Comput.*, vol. 3, no. 3, 1999.

[19] D. Searson, M. Willis, and G. Montague, “Chemical Process Controller Design Using Genetic Programming,” in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, 1998, pp. 359–364.

[20] B. McKay, M. Willis, and G. Barton, “Steady-state Modelling of Chemical Process System using Genetic Programming,” *Comput. Chem. Eng.*, vol. 21, pp. 981–996, 1997.

[21] J. R. Koza, J. Yu, M. A. Keane, and W. Mydlowec, “Evolution of a controller with a free variable using genetic programming,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2000, vol. 1802, pp. 91–105.

[22] T. Astrom, K. & Hagglund, “PID Controllers: Theory, Design, and Tuning.” 1995.

[23] J. Imae, S. Nakatani, and J. Takahashi, “GP based flight control in the windshear,” *IEEE SMC’99 Conf. Proceedings. 1999 IEEE Int. Conf. Syst. Man, Cybern. (Cat. No.99CH37028)*, vol. 2, no. c, pp. 650–653, 1999.

[24] K. L. Ng and R. Johansson, “Evolving Programs and Solutions Using Genetic Programming with Application to Learning and Adaptive Control,” *J. Intell. Robot. Syst.*, vol. 35, no. 3, pp. 289–307, 2002.

[25] K. J. Åström and B. Wittenmark, *Adaptive Control*, vol. 32. 1989.

[26] M. Ebner, “Evolution of a control architecture for a mobile robot,” in *Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES 98)*, 1998, vol. 1478, pp. 303–310.

[27] C. Lazarus and H. Hu, “Using genetic programming to evolve robot behaviours,” in *3rd British Conference on Autonomous Mobile Robotics & Autonomous Systems*, 2001.

[28] W.-P. Lee and J. Hallam, “Evolving reliable and robust controllers for real robots by genetic programming,” *Soft Comput. -- A Fusion Found. Methodol. Appl.*, vol. 3, no. 2, pp. 63–75, 1999.

[29] J. F. Miller, *Cartesian Genetic Programming*. 2011.

[30] A. J. Turner, “Evolving Artificial Neural Networks using Cartesian Genetic Programming,” 2015.

[31] B. Kadlic, I. Sekaj, and D. Pernecký, “Design of continuous-time controllers using cartesian genetic programming,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2014, vol. 19, no. 2007, pp. 6982–6987.

[32] B. Kadlic and I. Sekaj, “Controller Design Based on Cartesian Genetic Programming in MATALB,” *System*, vol. 3, 2007.

[33] S. Harding and J. Miller, “Evolution of robot controller using cartesian genetic programming,” *Genet. Program.*, pp. 62–73, 2005.

[34] Y. Hirayama, T. Clarke, and J. F. Miller, “Fault tolerant control using cartesian genetic programming,” *GECCO’08 Proc. 10th Annu. Conf. Genet. Evol. Comput. 2008*, pp. 1523–1530, 2008.

[35] P. Kundur, *Power System Stability and Control*, vol. 23. 2006.

[36] M. O'Neill and C. Ryan, "Grammatical Evolution," vol. 5, pp. 349–358, 2001.

[37] D. Pernecký and I. Sekaj, "Grammatical evolution based controller design," *19th Int. Conf. Soft Comput. Brno. Czech Repub.*, pp. 1–6, 2013.

[38] R. Burbidge, J. H. Walker, and M. S. Wilson, "Grammatical evolution of a robot controller," *2009 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 357–362, 2009.

[39] I. Sekaj, "Robust Parallel Genetic Algorithms with Re-initialisation," in *Parallel Problem Solving from Nature - PPSN VIII*, vol. 3242, 2004, pp. 411–419.

[40] W. Duch and N. Jankowski, "Survey of neural transfer functions," *Neural Comput. Surv.*, vol. 2, pp. 163–212, 1999.

[41] G. Cybenko, "Approximations by superpositions of sigmoidal functions," *Approx. Theory its Appl.*, vol. 9, no. 3, pp. 17–28, 1989.

[42] K. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1993.

[43] M. Caudill and C. Butler, *Understanding Neural Networks: Computer Explorations, Vols. 1 and 2*. 1992.

[44] R. J. Patton and F. J. Uppal, "Artificial Intelligence Approaches To Fault Diagnosis," *Intell. Syst. Eng.*, 1999.

[45] B. Widrow and M. A. Lehr, "30 Years of Adaptive Neural Networks : Perceptron , Madaline , and Backpropagation," vol. 78, no. 9, pp. 1415–1442, 1990.

[46] J. Li, J. Cheng, J. Shi, and F. Huang, "Brief Introduction of Back Propagation (BP) Neural Description of BP Algorithm in Mathematics," *Adv. Comput. Sci. Inf. Eng.*, vol. 2, pp. 553–558, 2012.

[47] M. Konomi and G. M. Sacha, "Influence of the learning method in the performance of feedforward neural networks when the activity of neurons is modified," pp. 1–11.

[48] H. Wang and Y. Wang, "Neural-network-based fault-tolerant control of

unknown nonlinear systems," *IEE Proceedings - Control Theory and Applications*, vol. 146, no. 5. p. 389, 1999.

[49] A. A. Pashilkar, N. Sundararajan, and P. Saratchandran, "A fault-tolerant neural aided controller for aircraft auto-landing," *Aerosp. Sci. Technol.*, vol. 10, pp. 49–61, 2006.

[50] E. S. E. Sugawara, M. F. M. Fukushi, and S. H. S. Horiguchi, "Fault tolerant multi-layer neural networks with GA training," *Proceedings. 16th IEEE Symp. Comput. Arith.*, 2003.

[51] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: From architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1. pp. 47–62, 2008.

[52] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[53] R. K. Belew, J. McInerney, and N. N. Schraudolph, "Evolving Networks: Using the Genetic Algorithm with Connectionist Learning," *Artif. Life II*, vol. 10, pp. 511–547, 1992.

[54] A. P. Wieland, "Evolving neural network controllers for unstable systems," *IJCNN-91-Seattle Int. Jt. Conf. Neural Networks*, vol. ii, pp. 667–673, 1991.

[55] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Mach. Learn.*, vol. 22, no. 1–3, pp. 11–32, 1996.

[56] K. O. Stanley and R. Miikkulainen, "Efficient Evolution of Neural Network Topologies," *Evol. Comput. 2002. CEC'02. Proc. 2002 Congr. Evol. Comput.*, no. figure 1, pp. 1757–1762, 2002.

[57] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks.," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 54–65, 1994.

[58] S. Luke and L. Panait, "A Comparison of Bloat Control Methods for Genetic Programming," *Evol. Comput.*, vol. 14, no. 3, pp. 309–344, 2006.

[59] S. Silva and E. Costa, "Dynamic limits for bloat control in genetic programming

and a review of past and current bloat theories," *Genet. Program. Evolvable Mach.*, vol. 10, no. 2, pp. 141–179, 2009.

[60] L. Trujillo, L. Munoz, E. Naredo, and Y. Martinez, "{NEAT}, There's No Bloat," *17th Eur. Conf. Genet. Program.*, vol. 8599, pp. 174–185, 2014.

[61] W. Duch and N. Jankowski, "Transfer functions: hidden possibilities for better neural networks," *9th Eur. Symp. Artif. Neural Networks*, pp. 81–94, 2001.

[62] K. O. Stanley, "Efficient Evolution of Neural Networks through Complexification," 2004.

[63] K. O. Stanley, D. D. Ambrosio, and J. Gauci, "A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks," vol. 15, no. 2, pp. 1–39, 2009.

[64] K. O. Stanley and B. D. Bryant, "Real-Time Neuroevolution in the NERO Video Game," no. Thurrott 2002, pp. 1–41, 2005.

[65] F. Silva, P. Urbano, S. Oliveira, and A. L. Christensen, "odNEAT: An Algorithm for Distributed Online, Onboard Evolution of Robot Behaviours," *Artif. Life 13*, no. July 2012, pp. 251–258, 2012.

[66] F. Silva, L. Correia, and A. L. Christensen, "Evolutionary online behaviour learning and adaptation in real robots," *R. Soc. Open Sci.*, vol. 4, no. 7, 2017.

[67] M. D'Angelo, B. Weel, and A. E. Eiben, "HyperNEAT versus RL PoWER for online gait learning in modular robots," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8602, pp. 777–788, 2014.

[68] S. Lee, J. Yosinski, K. Glette, H. Lipson, and J. Clune, "Evolving gaits for physical robots with the HyperNEAT generative encoding: The benefits of simulation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 7835 LNCS, pp. 540–549.

[69] M. Kingsley-Jones, "Evolving Robot Gaits in Hardware: the HyperNEAT Generative Encoding Vs. Parameter Optimization," *Aviat. Week Sp. Technol. (New York)*, vol. 172, no. 38, p. 47, 2010.

[70] S. Risi and K. O. Stanley, “Enhancing es-hyperneat to evolve more complex regular neural networks,” in *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO ’11*, 2011, p. 1539.

[71] E. Haasdijk, A. A. Rusu, and A. E. Eiben, “HyperNEAT for locomotion control in modular robots,” *Int. Conf. Evolvable Syst. Springer, Berlin Heidelb.*, pp. 169–180, 2010.

[72] M. M. Khan, G. M. Khan, and J. F. Miller, “Evolution of neural networks using Cartesian Genetic Programming,” *IEEE Congr. Evol. Comput.*, pp. 1–8, 2010.

[73] A. J. Turner and J. F. Miller, “Cartesian genetic programming encoded artificial neural networks,” *Proceeding fifteenth Annu. Conf. Genet. Evol. Comput. Conf. - GECCO ’13*, p. 1005, 2013.

[74] J. F. Miller and S. L. Smith, “Redundancy and computational efficiency in cartesian genetic programming,” *IEEE Trans. Evol. Comput.*, vol. 10, no. 2, pp. 167–174, 2006.

[75] J. Miller, “What Bloat? Cartesian Genetic Programming on {Boolean} Problems,” *2001 Genet. Evol. Comput. Conf. Late Break. Pap.*, pp. 295–302, 2001.

[76] S. Silva and E. Costa, “Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories,” *Genet. Program. Evolvable Mach.*, 2009.

[77] L. Vanneschi, M. Castelli, and S. Silva, “Measuring bloat, overfitting and functional complexity in genetic programming,” *… 12Th Annu. Conf. …*, pp. 877–884, 2010.

[78] C. Igel, “Neuroevolution for reinforcement learning using evolution strategies,” in *The 2003 Congress on Evolutionary Computation, 2003. CEC ’03.*, 2003, vol. 4, pp. 2588–2595.

[79] E. Galván-López, R. Poli, A. Kattan, M. O’Neill, and A. Brabazon, “Neutrality in evolutionary algorithms... What do we know?,” *Evolving Systems*, vol. 2, no. 3. pp. 145–163, 2011.

[80] M. Kimura, “Evolutionary rate at the molecular level,” *Nature*, vol. 217. pp. 624–

626, 1968.

[81] V. K. Vassilev and J. F. Miller, “The Advantages of Landscape Neutrality in Digital Circuit Evolution,” in *Ices’00*, 2000, pp. 252–263.

[82] T. Yu and J. Miller, “Neutrality and the Evolvability of Boolean Function Landscape,” in *Genetic programming*, 2001.

[83] T. Blickle and L. Thiele, “Genetic Programming and Redundancy,” *Genet. Algorithms within Framew. Evol. Comput. (KI-94 Work.*, pp. 33–38, 1994.

[84] M. M. Khan, G. M. Khan, and J. F. Miller, “Evolution of optimal ANNs for non-linear control problems using Cartesian Genetic Programming,” *Proc. 2010 Int. Conf. Artif. Intell. ICAI 2010*, vol. 1, pp. 339–346, 2010.

[85] M. Mahsal Khan, A. Masood Ahmad, G. Muhammad Khan, and J. F. Miller, “Fast learning neural networks using Cartesian genetic programming,” *Neurocomputing*, vol. 121, pp. 274–289, 2013.

[86] F. Zafari, G. M. Khan, M. Rehman, and S. Ali Mahmud, “Evolving Recurrent Neural Network using Cartesian Genetic Programming to Predict The Trend in Foreign Currency Exchange Rates,” *Appl. Artif. Intell.*, vol. 28, no. 6, pp. 597–628, 2014.

[87] J. W. Grizzle and A. Isidori, “Approximation by Superpositions of a Sigmoidal Function*,” *Math. Control. Signals, Syst.*, vol. 2, no. 4, pp. 315–341, 1989.

[88] J. Park and I. W. Sandberg, “Universal Approximation Using Radial-Basis-Function Networks,” *Neural Comput.*, 1991.

[89] K. Funahashi and Y. Nakamura, “Approximation of dynamical systems by continuous time recurrent neural networks,” *Neural Networks*, 1993.

[90] A. M. SCHÄFER and H.-G. ZIMMERMANN, “RECURRENT NEURAL NETWORKS ARE UNIVERSAL APPROXIMATORS,” *Int. J. Neural Syst.*, 2007.

[91] A. Garg and K. Tai, “A Hybrid Genetic Programming – Artificial Neural Network Approach For Modeling of Vibratory Finishing Process,” *Computing*, vol. 18, pp. 14–19, 2011.

[92] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[93] C. M. Fonseca and P. J. Fleming, “Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization,” *Icga*, vol. 93, pp. 416–423, 1993.

[94] J. Horn, N. Nafpliotis, and D. E. Goldberg, “A niched Pareto genetic algorithm for multiobjective optimization,” *Evol. Comput. 1994. IEEE World Congr. Comput. Intell. Proc. First IEEE Conf.*, vol. 1, pp. 82–87, 1994.

[95] N. Srinivas and K. Deb, “Muiltiobj ective Optimization Using Nondominated Sorting in Genetic Algorithms,” *Evol. Comput.*, vol. 2, no. 3, pp. 221--248, 1995.

[96] K. Deb, “Multi-Objective Optimization Using Evolutionary Algorithms,” *John Wiley & sons, LTD*. p. 497, 2001.

[97] J. Knowles and D. Corne, “The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation,” in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 1999, vol. 1, pp. 98–105.

[98] R. K. U. Evalife, “Diversity-Guided Evolutionary Algorithms.”

[99] T. Ray, K. Tai, and K. C. Seow, “Multiobjective Design Optimization by an Evolutionary Algorithm,” *Eng. Optim.*, 2001.

[100] F. Sarro, F. Ferrucci, and C. Gravino, “Single and Multi Objective Genetic Programming for software development effort estimation,” *Proc. 27th Annu. ACM Symp. Appl. Comput. - SAC ’12*, p. 1221, 2012.

[101] Y. Liang, M. Zhang, and W. N. Browne, “Multi-objective genetic programming for figure-ground image segmentation,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2016.

[102] U. Bhowan, M. Zhang, and M. Johnston, “Multi-Objective Genetic Programming for Classification with Unbalanced Data,” *Proc. 22nd Australas. Jt. Conf. Artif. Intell.*, 2009.

[103] C. K. Oh and G. J. Barlow, “Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming,” *Proc. 2004 Congr. Evol. Comput. (IEEE Cat. No.04TH8753)*, vol. 2, pp. 1538–1545, 2004.

[104] Z. Vasicek and L. Sekanina, “Circuit approximation using single- and multi-objective Cartesian GP,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9025, pp. 217–229, 2015.

[105] J. Hilder, J. A. Walker, and A. Tyrrell, “Use of a multi-objective fitness function to improve cartesian genetic programming circuits,” *2010 NASA/ESA Conf. Adapt. Hardw. Syst. AHS 2010*, pp. 179–185, 2010.

[106] G. Corriveau, R. Guilbault, A. Tahan, and R. Sabourin, “Review and study of genotypic diversity measures for real-coded representations,” *IEEE Trans. Evol. Comput.*, vol. 16, no. 5, pp. 695–710, 2012.

[107] M. Nei, “GENETIC DISTANCE BETWEEN POPULATIONS,” *Am. Nat.*, vol. 95, no. 949, p. 261, 1961.

[108] H. Abbass and K. Deb, “Searching under Multi-evolutionary Pressures,” in *Proceedings of the Fourth Conference on Evolutionary Multi- Criterion Optimization*, 2003, pp. 391–404.

[109] R. W. Morrison and K. A. De Jong, “Measurement of population diversity,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2002, vol. 2310, pp. 31–41.

[110] F. Herrera and M. Lozano, “Adaptation of Genetic Algorithm Parameters Based on Fuzzy Logic Controllers,” *Genet. Algorithms Soft Comput.*, pp. 95–125, 1996.

[111] O. Olorunda and A. P. Engelbrecht, “Measuring exploration/exploitation in particle swarms using swarm diversity,” in *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, 2008, pp. 1128–1134.

[112] a B. Doeschl-Wilson, D. Vagenas, I. Kyriazakis, and S. C. Bishop, “Exploring the assumptions underlying genetic variation in host,” *Genet. Sel. Evol.*, vol. 40, no. June 2002, pp. 241–264, 2008.

[113] R. W. Hamming, “Error Detecting and Error Correcting Codes,” *Bell Syst. Tech. J.*,

vol. 29, no. 2, pp. 147–160, 1950.

[114] D. J. S. Robinson, *An Introduction to Abstract Algebra*. 2003.

[115] M. Tang, Y. Yu, W. G. Aref, Q. M. Malluhi, and M. Ouzzani, “Efficient Processing of Hamming-Distance-Based Similarity-Search Queries Over MapReduce ∗,” *Edbt*, pp. 361–372, 2015.

[116] J. Koljonen, “On Fitness Distance Distributions and Correlations, GA Performance, and Population Size of Fitness Functions with Translated Optima,” *Proc. 9th Scand. Conf. Artif. Intell.*, pp. 68–74, 2006.

[117] L. Altenberg, “Fitness distance correlation: an instructive counterexample,” *Seventh Int. Conf. Genet. Algorithms*, pp. 57–64, 1997.

[118] A. L. Nelson, G. J. Barlow, and L. Doitsidis, “Fitness functions in evolutionary robotics: A survey and analysis,” *Rob. Auton. Syst.*, 2009.

[119] V. Trianni, “Advantages of Multi-Objective Optimisation in Evolutionary Robotics : Survey and Case Studies,” 2014.

[120] D. Greenhalgh and S. Marshall, “Convergence criteria for genetic algorithms,” vol. 30, no. 1, pp. 269–282, 2000.

[121] K. A. De Jong, “An Analysis of the Behavior of a Class of Genetic Adaptive Systems,” 1975.

[122] Z. Michalewicz, “Genetic Algorithms + Data Structures = Evolution Programs,” *Computational Statistics & Data Analysis*, vol. 24, no. 3. pp. 372–373, 1996.

[123] M. Safe, J. Carballido, I. Ponzoni, and N. Brignole, “On Stopping Criteria for Genetic Algorithms,” *Adv. Artif. Intell. – SBIA 2004*, pp. 405–413, 2004.

[124] H. L. Liu, L. Chen, K. Deb, and E. Goodman, “Investigating the effect of imbalance between convergence and diversity in evolutionary multi-objective algorithms,” *IEEE Trans. Evol. Comput.*, vol. PP, no. 99, pp. 408–425, 2016.

[125] P. Chakraborty, S. Das, G. G. Roy, and A. Abraham, “On convergence of the multi-objective particle swarm optimizers,” *Inf. Sci. (Ny).*, vol. 181, no. 8, pp.

1411–1425, 2011.

[126] Y. Cao, B. J. Smucker, and T. J. Robinson, "On using the hypervolume indicator to compare Pareto fronts: Applications to multi-criteria optimal experimental design," *J. Stat. Plan. Inference*, vol. 160, pp. 60–74, 2015.

[127] T. Goel and N. Stander, "A study on the convergence of multiobjective evolutionary algorithms," *Prepr. Submitt. to 13th AIAA/ISSMO …*, pp. 1–18, 2010.

[128] D. Brockhoff, T. Friedrich, and F. Neumann, "Analyzing hypervolume indicator based algorithms," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5199 LNCS, pp. 651–660.

[129] R. Berghammer, T. Friedrich, and F. Neumann, "Convergence of set-based multi-objective optimization, indicators and deteriorative cycles," *Theor. Comput. Sci.*, vol. 456, pp. 2–17, 2012.

[130] E. Zitzler and K. Simon, "Indicator-Based Selection in Multiobjective Search," *8th Int. Conf. Parallel Probl. Solving from Nat. (PPSN VIII)*, vol. 3242, no. i, pp. 832–842, 2004.

[131] E. Zitzler and L. Thiele, "Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study," *Proc. Int. Conf. Parallel Probl. Solving from Nat.*, no. September, pp. 292–304, 1998.

[132] E. Zitzler, D. Brockhoff, and L. Thiele, "The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration," *Evol. Multi-Criterion Optim.*, vol. 4403, pp. 862–876, 2007.

[133] L. Lu and C. M. Anderson-Cook, "Adapting the hypervolume quality indicator to quantify trade-offs and search efficiency for multiple criteria decision making using pareto fronts," *Qual. Reliab. Eng. Int.*, vol. 29, no. 8, pp. 1117–1133, 2013.

[134] E. Zitzler, J. Knowles, and L. Thiele, "Quality assessment of pareto set approximations," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5252 LNCS, pp. 373–404.

[135] F. Sambo, M. Borrotti, and K. Mylona, “A coordinate-exchange two-phase local search algorithm for the D- and I-optimal designs of split-plot experiments,” *Comput. Stat. Data Anal.*, vol. 71, pp. 1193–1207, 2014.

[136] N. Beume, B. Naujoks, and M. Emmerich, “SMS-EMOA: Multiobjective selection based on dominated hypervolume,” *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1653–1669, 2007.

[137] M. Emmerich, N. Beume, and B. Naujoks, “An EMO algorithm using the hypervolume measure as selection criterion,” in *Evolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005*, 2005, pp. 62–76.

[138] J. Bader and E. Zitzler, “HypE: an algorithm for fast hypervolume-based many-objective optimization.,” *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, 2011.

[139] P. E. McKnight and J. Najab, “Mann-Whitney U Test,” in *The Corsini Encyclopedia of Psychology*, 2010.

[140] A. Vargha and H. D. Delaney, “A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong,” *J. Educ. Behav. Stat.*, vol. 25, no. 2, pp. 101–132, 2000.

[141] K. Alden, M. Read, P. S. Andrews, J. Timmis, and M. Coles, “Applying spartan to Understand Parameter Uncertainty in Simulations,” *R J.*, vol. 6, no. 2, pp. 63–80, 2014.

[142] S. Robinson, *Simulation: The Practice of Model Development and Use*, vol. 67. 2004.

[143] M. T. Alexander, D. C. Montgomery, and G. Runger, “Applied Statistics and Probability for Engineers,” *Technometrics*, vol. 37, no. 4, p. 455, 1995.

[144] M. Read, P. S. Andrews, J. Timmis, and V. Kumar, “Techniques for grounding agent-based simulations in the real domain: a case study in experimental autoimmune encephalomyelitis,” *Math. Comput. Model. Dyn. Syst.*, vol. 18, no. 1, pp. 67–86, 2012.

[145] C. Pinciroli, V. Trianni, R. O’Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, T. S. Stirling, Á. Gutiérrez, L. M. Gambardella,

and M. Dorigo, “ARGoS: A pluggable, multi-physics engine simulator for heterogeneous swarm robotics,” *IRIDIA – Tech. Rep. Ser.*, pp. 1–22, 2011.

[146] Y. Chen, X. Zou, and W. Xie, “Convergence of multi-objective evolutionary algorithms to a uniformly distributed representation of the Pareto front,” *Inf. Sci. (Ny).*, vol. 181, no. 16, pp. 3336–3355, 2011.

[147] S. Hosangadi, “Distance Measures for Sequences,” *Arxiv*, p. 16, 2012.

[148] C. D. Pilcher, J. K. Wong, and S. K. Pillai, “Inferring HIV transmission dynamics from phylogenetic sequence relationships,” *PLoS Medicine*, vol. 5, no. 3. pp. 0350–0352, 2008.

[149] J. Heaton, “Programming neural networks in Java,” *Bttp//Www. Heatonresearch. Com*, 2004.

[150] K. Deb and R. B. Agrawal, “Simulated Binary Crossover for Continuous Search Space,” *Complex Syst.*, vol. 9, pp. 1–34, 1994.