Multi Objective Genetic Programming Dynamic Flexible Job Shop Scheduling

by

Gaofeng Shi

A report of COMP589 research project
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Master of Science
in Computer Science.

Victoria University of Wellington
2022

Abstract

Job shop scheduling (JSS) is the challenging combinatorial optimisation problem which has been studied for many years. It is to schedule a number of jobs on a number of machines where the scheduling is the set of processing sequences for each of the machines. The objective of JSS problem is minimize the make-span, total flowtime, or total weighted tardiness of job shop. Extension of JSS is the flexible (FJSS) that allows a job operation which can be processed by any machine if the candidate machines are idle. In addition, FJSS problems are not only for searching the sequencing of operations, but also need to find the routing assignment of operations to the suitable machines. However, in the real world the situation of Job shop scheduling might not always occur under static conditions such as stochastic job arrival, uncertain processing time, and unexpected machines breakdown. In particular, this thesis focuses on the $dynamic$ flexible job shop scheduling (DFJSS) problem which is more complicated under the uncertain complex environment.

As DFJSS problems don't have the prior information for scheduling jobs in advance, dispatching rule becomes the best solution for this problem. The dispatching rule makes the decision which job is assigned to a machine when the machine become idle. It can react to unexpected change quickly, particularly for solving the dynamic and stochastic problem.

Hyper-Heuristic is searching for heuristics in search space instead of directly searching solutions of the problem. It becomes the best approach for designing DFJSS dispatching rules. In particular, many genetic programming hyper-heuristic (GPHH) approaches have been proposed to generate the effective dispatching rules for DFJSS problems. Dispatching rule includes Sequencing rule and Routing rule which represented by

GPHH tree. However, the GPHH generated dispatching rule is not easy understanding by human as the best performance heuristic often contains huge GP rule-size. What is the relationship between performance and rule-size of solution generated by GPHH that is interesting topic to be studied. It is worthwhile to study the two conflicted objectives by Multi-objective Genetic Programming (MOGP) to have better interpretability and understanding of GP dispatching rule.

The goal of this report is to study and search better effectiveness MOGP algorithm for multi-objective performance and rule-size of DFJSS problem.

Firstly to study the background of the evolutionary computation of GP and the problem of DFJSS. Then, it will learn some recently related research of GPHH for DFJSS and related multi-objective genetic programming MOGP studies.

This report will introduce two new approaches of MOGP for DFJSS. Firstly, a new proposed α dominance and archive MOGP algorithm αMOGP-n which has obtained significantly better effectiveness than the original algorithm. αMOGP-n apply the adjustment and normalization on αMOGP-a method [45] that improved the algorithm magnificently. The adjustment by setting $\alpha = 0$ to get the reference Pareto front for α self-adaptive algorithm and extract the individuals for updating archive. Normalization makes the α dominance on non-dominate sorting process are more stable in variety DFJSS scenarios.

Then, it will demonstrate a novel algorithm α dominance multi population MOGP algorithm αMOGP-mp that achieved even more significant better effectiveness than the new algorithm αMOGP-n. αMOGP-mp implements two MOGP populations which potentially decomposed to two different tasks, one applying α dominance non-dominate sorting and other is traditional non-dominate sorting. The two population not only searching multi direction of MOGP rule-size but also sharing knowledge by transfer learning from their rule-size intersection part.

Acknowledgments

I would like to thank my supervisors, Dr. Yi Mei and Dr. Fangfang Zhang for providing me with constant support and encouragement during my Master's research. The feedback they provided on my research has been invaluable for my learning experiences as a Master's student and helped me improve various skills such as my learning, critical thinking, and analytical skills.

Contents

1 Introduction **1**

1.1 Problem Statement 1
1.2 Research Goals 3
1.3 Major Contributions 4
1.4 Organisation 5

2 Literature Review **7**

2.1 Basic Concepts 7
2.1.1 Scheduling 7
2.1.2 Machine Learning 8
2.1.3 Hyper-Heuristic 10
2.1.4 Evolutionary Computation 11
2.2 JSS Problem 12
2.2.1 Job Shop Scheduling 12
2.2.2 Flexible job shop scheduling 13
2.2.3 Dynamic flexible job shop scheduling 14
2.3 Genetic Programming 15
2.3.1 Representations 15
2.3.2 Fitness Function 17
2.3.3 Genetic Programming Hyper-Heuristic for DFJSS . . 18
2.4 Related Work of GP for DFJSS 18
2.4.1 Multi-objective Genetic Programming 19
2.4.2 Cooperation Coevolution GP 20

2.4.3 Surrogate Genetic Programming 21
2.4.4 Multitask Genetic Programming 25
2.4.5 Feature Selection 28
2.4.6 Other related work for Multi-Objective GP 32
2.5 Issues and Challenges 33

3 α Dominance and Archive MOGP for DFJSS 35
3.1 α Dominance for Unbalance Objectives MOGP 35
3.1.1 Unbalanced Objectives 36
3.1.2 α Dominance 36
3.1.3 α Dominance and Archive 38
3.2 α Dominance and Archive MOGP for DFJSS 38
3.2.1 Overall Framework 39
3.2.2 α Self-adaptive Scheme 41
3.2.3 Archive Algorithm 42
3.2.4 $\alpha = 0$ reference Pareto front for Update α and Archive 44
3.2.5 Normalization of Objective Value 45
3.3 Experiment Design 46
3.3.1 DFJSS Simulation Model and Scenarios 47
3.3.2 Parameter Settings 48
3.3.3 Experiment Settings 48
3.4 Results and Discussion 50
3.4.1 Results Comparison 50
3.4.2 Results Discussion 55
3.5 Summary 57

4 Multi Population MOGP Approach 59
4.1 Introduction 59
4.2 Applying Multi Population αMOGP-mp 60
4.2.1 Overall Framework 60
4.2.2 Multipopulation Multi-Objective GP 62
4.2.3 Transfer Learning between Two Populations 62

4.2.4 Share Knowledge Crossover-Pipeline Algorithm . . . 63
4.3 Experiment Design . . . 64
4.3.1 Simulation Model and Scenarios . . . 64
4.3.2 Parameter Settings . . . 64
4.3.3 Experiment Setting . . . 66
4.4 Results and Discussion . . . 67
4.4.1 Results Comparison . . . 68
4.4.2 Results Discussion . . . 69
4.5 Summary . . . 71

5 Conclusions **75**

Chapter 1

Introduction

1.1 Problem Statement

Job shop Scheduling (JSS) are NP-hard problems [16]. JSS problems are set of jobs and machines denoted by N and M respectively. Each job consists of a sequence of operations P that each operation has the release time and due date and has to be assigned to one of the eligible machines. The constraints of JSS are that all of the job's operations must be in order that one operation only can be processed on one machine, and one machine can only process one operation at a time and can't be interrupted. The scheduling objective of the job shop is to minimize the make-span, the total flowtime, and the total weighted tardiness [25][26].

The extension version of JSS can be categorised to flexible job shop scheduling (FJSS) and dynamic flexible job shop scheduling (DFJSS). The FJSS problem allows the job operation to be processed by any machine if the candidate machines are idle. It makes the problem more complicated comparing with JSS. The scheduling of JSS only requires a sequencing of operations on suitable machines, while FJSS scheduling not only searches for sequencing of operations, but also has to find the routing of operations to the idle machines [8]. The good solution for FJSS can be searching for heuristic algorithms such as Truncated exponential schemes, Greedy algo-

rithms, Simulated Annealing and Tabu search [3]. Dispatching rules are a distributed sequencing strategy which can assign the priority of job to the machine and also make routing of operations on the machine. There are numbers of dispatching rules developed for FJSS, for example. the shortest Processing time (SPT), and the earliest due date (EDD) [3].

However, the real world of environment of Job Shop scheduling is not static. The job arrives randomly, the machine breakdown, and uncertain job operation processing time which makes the job shop in dynamic situation. Dynamic flexible job shop scheduling (DFJSS) does not have the prior information for scheduling job in advance. As the dispatching rules reacting to the uncertain change quickly, solving DFJSS problem is to study designing the better dispatching rule [5]. Hyper-heuristic is to search for heuristics in search space instead of searching solutions of problem directly [6]. It generates the heuristics as dispatching rules for DFJSS solution that can feasibly adopt the job shop scheduling dynamic environment changing automatically and quickly [5].

Genetic Programming is an evolutionary computation approach which can evolve the represented individual solution automatically by its evolutionary process. The tree structured individual representations of GP are easily encoded to dispatching rule for DFJSS problem. The priority function of dispatching rules of JSS can be represented by GP tree easily as its mathematical expression. The GP crossover and mutation evolve the individuals from selected parents by fitness. Employing Genetic programming with hyper-heuristic (GPHH) to discover the solution of DFJSS becomes the popular domain study area which attracted many researchers.

Dispatching rules in Dynamic Flexible Job Shop scheduling (DFJSS) divided into two parts sequencing and routing rules which can be generated by Genetic Programming with Hyper-heuristic (GPHH) effective and efficiently from cooperative coevolution perspective. Multi-objective cooperative coevolution GPHH for DFJSS has achieved a very high performance result [30][61]. In addition, the single objective GPHH cooperative coevo-

lution method was developed for DFJSS by Yska et al.[48], and the more advance study have proposed by Zhang et al. [54].

The most expensive part of GP computation is long time fitness evaluation. Surrogate model is manipulated to reduce the computation time of GPHH for DFJSS [17][33][60][57]. On the other hand, GPHH with multitask approach for DFJSS has improved the efficiency of algorithm performance dynamically [50][52][56].

In order to have better understanding and good interpretability of GP trees representing policies rules, the multi-objective genetic programming with performance and rule-size conflict objective was sutyies by Wang et al. [44][45]. Particularly, the α dominance [19] with archive MOGP method archived very good effectiveness [45].

Good interpretabiltity and better understanding of GP trees representing DFJSS scheduling dispatching rule is very important perspective of evolutionary optimization learning. Because the highest performance GP generated dispatching rule often have larger complicated GP tress rule size. It is interesting to study the relationship between the performance and rule-size of GP as multi-objective problem. As many researches have already studied this topic by MOGP in other problem, it is necessary and valuable to study the two conflicted objective performance and rule-size for DFJSS problem by MOGP.

1.2 Research Goals

The goal of this research is to develop the effectiveness algorithm to solve the multi-objective performance and rule-size of DFJSS problem by MOGP. The existing algorithm α dominanc self-adaptive and archive αMOGP-a [45] was successfully implemented on Uncertain Capacitated Arc Routing Problem (UCARP) problem. However, it cann't archive the same better effectiveness on DFJSS. What is the limitation of existing algorithm for DFJSS problem, and how to implement α dominance in different approach

to improve the effectiveness of MOGP for DFJSS? This research will have two phases into the following key objectives.

1. Investigate the existing α dominance MOGP algorithm to obtain better understanding about unbalance objective of DFJSS problem. Discover the adjustment and approach from α dominance self-adaptive and archive MOGP to obtain better effectiveness for DFJSS problem.

2. Develop a new approach for DFJSS multi-objective problem from different aspects such as multi-task transfer learning, or decomposition multi-objective, which apply α dominance multi-objective can achieve reasonable better effectiveness.

1.3 Major Contributions

This research have achieved the following major contributions.

1. This research developed an adapting α dominance and archive MOGP approach which applying the adjustment and normalization on original algorithm. It adjusted the algorithm by using $\alpha = 0$ Pareto front as the reference front for α self-adaptive schema and updating individuals to archive. The normalization makes algorithm α dominance non-dominate sorting process are more stable and have more uniformity with variety DFJSS scenarios. The new algorithm αMOGP-n obtained more significant better effectiveness than the traditional NSGP algorithm and original αMOGP-a [45] algorithm.

2. This research proposed a novel algorithm for DFJSS multi-objective problem called αMOGP-mp by implemented multi population decomposed into multi task MOGP that oriented one task with α dominance and other with traditional non-dominate. The novel algorithm αMOGP-mp achieved the best significant better effectiveness in both algorithm αMOGP-a [45] and αMOGP-n.

1.4 Organisation

The rest of this report is organised as follows:

- Chapter 2 gives the background information, including the problem description and genetic programming explanation. Then, the related work of genetic programming with hyper-heuristic for dynamic flexible job shop scheduling problems studies also are learned and analysed in this Chapter.
- The adapting α self-adaptive and archive MOGP with $\alpha = 0$ reference Pareto fornt adjustment and normalization approach αMOGP-n be introduced in Chapter 3.
- Chapter 4 introduces a novel algorithm α dominance multi population multi-objective genetic programming αMOGP-mp for DFJSS.
- Chapter 5 gives the conclusions and future work.

Chapter 2

Literature Review

This Chapter describes the review of background literature on the thesis topic Genetic Programming for Dynamic Job Shop Scheduling problem related concepts and research. First, section 1 will introduce the basic concepts. Job shop scheduling problem will be explained at section 2. The section 3 of this chapter will introduce genetic programming. The following section 4 described the related work of GPHH for Dynamic Flexible Job Shop Scheduling. Section 5 discussed the issues and challenges of related research works. The final section is the summary of the Chapter.

2.1 Basic Concepts

This section will focus on the basic concepts related to the main research area. It will explain some definitions about Machine Learning, Scheduling, Routing and Sequencing, Hyper-Heuristics, and Evolutionary Computation.

2.1.1 Scheduling

Scheduling is a type of process for decision-making which becomes a crucial role in manufacturing and modern industries [36]. It allocates the re-

sources to organize the given time periods of tasks and its goal is to make the optimization of objectives. Nowadays, the resources and tasks can be utilized by many different forms in an organization. For instance, the resources can be workshop machines, airport runways, hospital staff, computing process unites, and so on. The tasks may be a production operation process, airplane take-off and landing, the shift of staff, programming executions, and so on [36]. Scheduling the resources and tasks to achieve the objective goals which can be seen as searching the solution of job shop scheduling problem.

In Job shop scheduling problem (JSS) the scheduling is to plan a set of processing sequences for each of the machines which has a prior assignment of operations to machines. Flexible Job Shop Scheduling (FJSS) problem don't have the prior assignment to fix the machines. For each job need to plan a sequence of operations, and for each operation there are a set of equivalent machines can be process the operation with possibly different times. The routing is to assign each operation to a machine, and the sequencing is to assign operations on each machine. Routing and sequencing rule must be solved simultaneously in order to obtain the feasible schedule minimizing a given objective function [3]. The objective function depended on the different environment of manufacturing and industry. For example, it might be the minimization of make-span, flowtime, and tardiness in job shop scheduling.

2.1.2 Machine Learning

Machine Learning is part of artificial intelligence. It applies example data or past experience programming computers to find solutions of problems, or optimize a performance criterion [12]. Machine learning can build the model with defined parameters, using training data or past experience to learn and optimize the parameters of the models. The model may be predictive for predictions, or descriptive to obtain the knowledge from

the data, or both [12]. Machine Learning can be divided broadly into three categories which are $SupervisedLearning$, $UnsupervisedLearning$, and $ReinforcementLearning$.

Supervised Learning

The machine learning task or model is learnt from maps of input to output which are from the training data or example of input-output pairs [12]. The task of supervised learning can be explained this way. Given a training set of N instances, the pairs input-output $(x_1, y_1), (x_2, y_2), ...(x_N, y_N)$ of each y_j can be generated by an unknown $y = f(x)$. The task explores a h that approximates the true function f [39]. In the task, if the output y are categorical value, the supervised learning will be classification learning problem. Meanwhile, if the y are numerical value it will be regression problem. [39].

Unsupervised Learning

The machine learning algorithm or model which learns patterns or structures of data from unlabeled training instances [12]. Clustering is the most common unsupervised machine learning task that detects potentially useful clusters of input instances[39]. Clustering method called density estimation which is to find clusters or grouping of input. For example, the company allocates their customers to different natural groupings by the past transactions of customers with the clustering model [12].

Reinforcement Learning

The machine learning agent learns what to do and how to associate situations to actions in order to get maximum rewards and less punishments. The agent discovers which actions yield the most reward when trying it, but is not told which actions to take [43]. The actions may not only generate reward immediately and also the next situation, through that, all

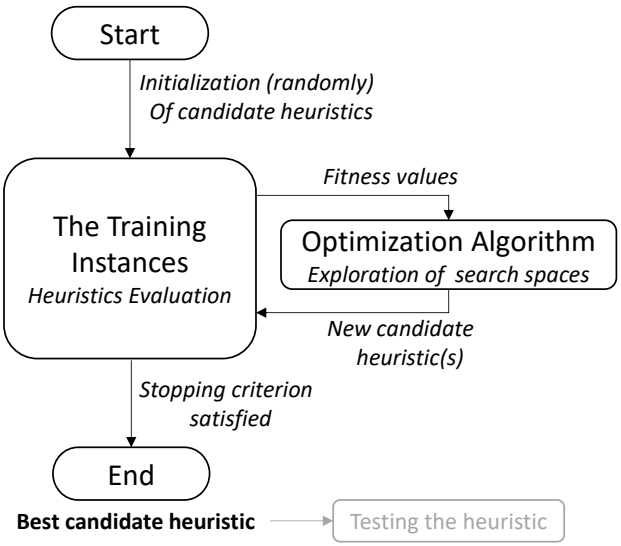

Figure 2.1: The procedure of a hyper-heuristic for heuristics generation [5]

subsequent rewards. A learning agent has to sense the state of its environment to some extent and has to take actions which affect the state, and also should have goal relating to environment state [43]. Reinforcement learning is different with supervised learning, the former learned from its own experience and the latter learned from training set of labeled examples. Reinforcement learning also is different with unsupervised learning which is typically about finding pattern hidden in unlabeled data [43]. In addition, applying Reinforcement Learning approach is another interesting popular topic to solve Dynamic Job Shop Scheduling problems.

2.1.3 Hyper-Heuristic

Heuristics definition in search strategy is the method which is applied in order to search a solution space for an optimization of problem solution [34]. Hyper-Heuristic is searching for heuristics in search space instead of directly searching solutions of problem [6].

Fig. 2.1 shows the basic illustration of the procedure of a hyper-heuristic for the heuristics generation. Encoding or representation of candidate heuristics defined the search space that optimization algorithm explores the search space depended on the fitness value as the quality of candidate heuristic, and recreated new candidate solutions during the iterations until the stopping criterion is achieved[5].
Many real world combinatorial complex problems are implemented successfully by the hyper-heuristics approach such as bin-packing, personnel scheduling, timetabling, production scheduling, and vehicle routing problems[6]. Applied hyper-heuristic search method to find the suitable dispatching rule which can be the best way of solution for dynamic flexible job shop scheduling problem. Because hyper-heuristic can generate heuristics which has the feasibility to adopt the dynamic job shop scheduling environment change automatically and quickly [5].

2.1.4 Evolutionary Computation

Evolutionary computation is a group of algorithms for artificial intelligence deep machine learning optimization which is inspired by biological evolution. Evolutionary computation can be broadly divided to two subcategory which are evolutionary algorithms (EAs) and swarm intelligence (SI).

Evolutionary algorithms

Evolutionary algorithms (EAs) have the common method that started with initialization. The initialization of the population randomly creates the individuals which are the set of candidate solutions. After the creation of the population, the standard EAs evolving as natural evolution has three processes to produce the new generation of population for each iteration. Selection based on the value of objective function from each individual choose the parents to produce the offspring. Breeding is to create the off-

spring from selected parents by two common operations crossover and mutation, which will create a new generation of population. Fitness function evaluate the objective value of individual which is setup for the measurement of solution. Those three process can be iterated until the best solution is found or a preset of generation limit is reached [11][9]. Meanwhile, there are four main algorithms classified from EAs which are genetic algorithm (GA), genetic algorithm (GP), evolutionary programming (EP), and evolution strategies (ES) [11].

2.2 JSS Problem

This section covers the concepts of job shop scheduling (JSS) problem. From the degree of the difficulty, there are three levels of the category into those problems. They will describe separately by following part:.

2.2.1 Job Shop Scheduling

The Job shop scheduling Problem (JSS) is the real world complected optimization problem which belongs to the NP-hard problems [16]. In a job shop scheduling (JSS) problem, there are a number of jobs $J = \{J_1, J_2, ..., J_n\}$ and machines $M = \{M_1, M_2, ..., M_m\}$. For each job J_i it consists of a sequence of operations $O_i = \{O_{i1}, O_{i2}, ..., O_{ij}\}$, as well as an release time $t_0(J_i)$ and a due date $d(J_i)$. Each operation O_{ij} must be allocated to an eligible machine $m(O_{ij}) \in M$.
Job shop scheduling has to follow the constraints:

- Each job predefined the operations in order, and one operation O_{ij} cannot start processing until all its precedent operations are completed.

- Each operation O_{ij} can be processed only by one of its candidate machines $m(O_{ij})$.

- Each machine can process only one operation at same time.
- The scheduling is non-preemptive, i.e., once started, the processing of an operation cannot be stopped or paused until it is finished.

A schedule X in JSS, which will be represented as a set of processing sequences for each of the machines. The processing sequence $\mathbf{P}(M_k)$ consists the processing of all the operations $\{O_{ij}|m(O_{ij}) = M_k\}$, and to be with the starting time $\tau(O_{ij})$. The objectives of JSS commonly includes the minimize of make-span (C_{max}), the total flowtime (ΣC_j), and the total weighted tardiness $(\Sigma w_j T_j)$ [26].

2.2.2 Flexible job shop scheduling

The flexible job shop scheduling (FJSS) problem is an extension of the JSS problem. The flexible job shop scheduling allows a job operation can be processed by any machine if the candidate machines are in idle. In a flexible job shop scheduling (FJSS), there is a set of jobs $J = \{J_1, J_2, ..., J_n\}$ to be processed on set of machines $M = \{M_1, M_2, ..., M_m\}$. The job consists of a sequence of operations $O_i = \{O_{i1}, O_{i2}, ..., O_{ij}\}$. Each the operation O_{ij} of J_i has to be operated on an available machine $m(O_{ij}) \in M$. All jobs and machines are available at time 0. The following constraints is for FJSS:

- Each job predefined the operations in order, and one operation O_{ij} can only start processing when all its precedent operations completed.
- Each operation O_{ij} can be processed only by one of machines $m(O_{ij})$.
- Each machine can process only one operation at same time.
- It is non-preemptive operation, i.e., once start, the processing of an operation cannot be interrupted until it is completed.

The JSS requires a sequencing of operations on suitable machines, while in the FJSS the allocation of an operation is not fixed in advance and can thus

be processed on a set of capable machines. Hence, in FJSS problem, there are not only for searching the sequencing of operations, and also need to deal with the routing assignment of operations to suitable machines[8]. The problems of FJSS is to allocate each operation to an idle machine (routing problem), and to sequence the operations on the machine (sequencing problem)[35].

2.2.3 Dynamic flexible job shop scheduling

JSS and FJSS can be see as static scheduling problems. Although, The flexible job shop scheduling (FJSS) problems as we know belongs to the category of NP-hard problems[10]. All jobs information is known before the implementation of the system in JSS and FJSS environment. In the real world the situation of Job shop scheduling might not always be under static conditions such as stochastic job arrivals, uncertain processing times, and unexpected machines breakdowns. This uncertain environment of flexible job shop scheduling is called Dynamic Flexible Job Shop Scheduling (DFJSS) problem. The problem of DFJSS don't have the prior information for scheduling job in advance. A research for DFJSS is to analyse and study the different levels influence of the parameters from machine breakdown [18]. However, the Dynamic flexible job shop scheduling has various properties for the problem consideration. Because dispatching rules can react to unexpected change quickly, particularly for solving dynamic and stochastic problems [5]. Therefore, Dispatching rules study become the main stream for DFJSS. As heuristics are the effective designation for the specific and class problems, hyper-heuristic is the best approach for designing DFJSS dispatching rules.

2.3 Genetic Programming

Genetic Programming (GP) is an Evolutionary Computation (EC) approach inspired by natural selection into the evolutionary process of artificial computing system[22]. Genetic programming started from the initialization of the population. Each of the individuals from the population represents a solution of the problem. The population will be evolved by the GP iteration process: selection, crossover, and mutation. The selection process based on the objective value which was evaluated by the fitness function. The termination of iteration can be completed when the GP achieved the goal of criteria or meet the setup of generation number. The individual of GP is tree structure that contains terminals as leaf nodes of the tree, and functions referred as nodes of the tree.

2.3.1 Representations

GP tree is the representation of GP. The individual of GP population contains two parts composed by terminals and functions which appropriate to the problem domain [22]. The set of terminals applied typically comprises appropriate to the problem domain and numeric constants [22]. For example, in the dispatching rule of JSS, release time of a job (RT), current operation processing time (PT), work content in next queue(WINQ), next processing time (NPT), and constant 0 which are used as the terminals representing the current time decision of job [4][29]. Fig. 2.2 shows benchmark dispatching rule heuristics of JSS represented by tree-based GP. Earliest Release Data first (ERD) means the priority of processing the job order is calculated by $-RT = (0 - RT)$. Shortest Processing Time First (SPT) represented by $-PT = (0 - PT)$, and Holthaus rule use $-(2PT + WINQ + NPT)$[4]. The set of functions can be used by arithmetic operations $(+, -, \times, \div)$, mathematical functions, conditional logical operations, and domain-specific functions [22]. The initialization process randomly generate individuals of population by compositions of available functions

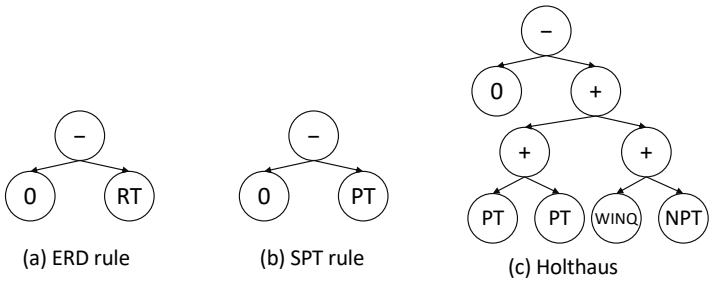

Figure 2.2: Dispatching rule heuristics represented by GP tree. [4]

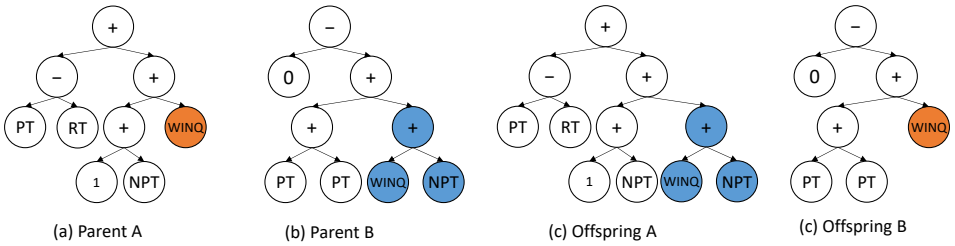

Figure 2.3: GP crossover operation

and terminals. However, the sufficient set of primitive function should be well defined for any combination of arguments with every terminals their value may encounter, which terminals and functions satisfies the closure properties [22]. For the GP tree based representation, which evolve the new generation of individual by genetic operations during the GP iterations. Subtree crossover and subtree mutation are the commonly genetic operation in GP to search the new solutions by exploration and exploitation [29]. The subtree crossover operation reproduces new individuals by randomly combining subtrees, which are the parts of subtree from selected two parents. Fig. 2.3 shows the selected parents produce the offspring by GP crossover operation. In addition, the operation subtree mutation creates the new individual by modification of chosen individual, which by selecting a node from the individual and replacing by randomly-generated

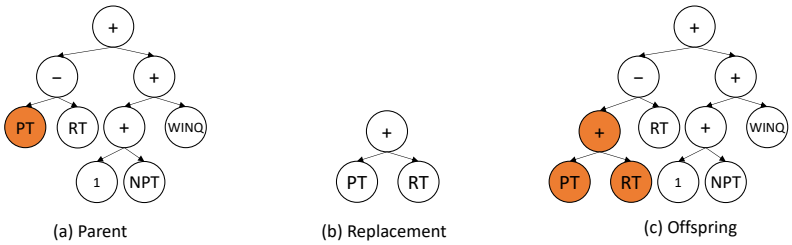

Figure 2.4: GP mutation operation

subtree from the root of that selected node[29]. Fig. 2.4 shows the mutation operation of GP.

2.3.2 Fitness Function

Fitness function in GP is either to measure or compare in terms of how well each individual solution in the population which performs the particular problem environment [22] [23].

Fitness naturally is measured by the error calculated by the computer program which error closer to zero is the better one [22]. In classification problem the Error of Accuracy can be minimized as the GP fitness function. Meanwhile, in GP regression problem the fitness function can be minima the root mean square error (RMSE). In real practice of many problems, fitness cases may represent different values of the program inputs, different system initial conditions, or different environments [23]. For instance, the design of new scheduling heuristics may contains multiple conflicting fitness objectives such as the three objectives of dispatching rules of DFJSS (makespan, mean tardiness, and flowtime) [29]. Some GP program in multiple fitness cases that its fitness is measured as sum, average, or weighted over the variety representative of different situations [22].

2.3.3 Genetic Programming Hyper-Heuristic for DFJSS

Genetic Programming is fixed length encoding representation which is very suitable with length of heuristics in the given problem domain[6]. DFJSS dispatching rule can be represented by GP tree structure. Terminal set of GP tree can be easily represented from DFJSS features. Function set of GP tree will be relevant to dispatching rule of DFJSS. GP tree has the fixed length structure which is suitable to encode different scheduling rules as its flexibility [29] [32]. Genetic programming is particularly suitable for dynamic job shop scheduling problem. Although, before applying GP for DFJSS, many research show the dispatching rule was hard to cope with multiple objective dynamic job shop situation [2] [37] [40] . Genetic programming based hyper-heuristic (GPHHs) has the advantage to design heuristics as their search algorithms are very flexible and many advanced apparochs have been developed to cope with multiple objectives of DFJSS [29]. The GP tree can easily represented priority function of dispatching rules by it's mathematical expression [21], and GP has large heuristic search space to discover unknown and effective dispatching rules [29]. Moreover, sophisticated dispatching rules can be encoded and evolved by GP individuals which has the flexibility of representations, and the advantages of available EC search mechanisms makes GP can enhance the dispatching rules quality of DFJSS [29].

2.4 Related Work of GP for DFJSS

This Section introduces the recently related studies of Dynamic Flexible Job Shop Scheduling (DFJSS) problem by various approach with Genetic Programming algorithm. In addition, this research related Multi-objective GP study α dominance MOGP will be introduced in this section.

Table 2.1: The Related Work of GPHH for DFJSS Addressed by Approach

Approach	References
Multi-objective	[30], [61], [44], [45]
Cooperation Coevolution	[48], [54]
Surrogate Model	[17], [33], [57], [60]
Multitask Method	[50], [52], [56]
Feature Selection Method	[13], [25], [26], [55], [58].

2.4.1 Multi-objective Genetic Programming

multi-objective cooperative coevolution (DMOCC)

Nguyen et al.[30] developed a novel genetic programming based hyper-heuristic (GPHH) for DFJSS in 2012, The method was called multi-objective cooperative coevolution (DMOCC) method, which was the first developer applying the GP to solve dispartching rules (DRs) and due-data assignment rules (DDARs) simultaneously. In their study, the GP representations divided to two sub-populations DDARs and DRs by multi-objective GP methods. The individuals of DDARs and DRs use different terminals. DDAR determine flowtimes/due-dates more focusing on detailed operation information which assign a due-date to the job when the new job arrived. DR is a priority function which use different pieces of information from job and machines which applied for the jobs in the queue of the machine when the machine becomes idle. Those scheduling policies are combined the representative individuals by two sub-population.

Nguyen at al.[30] proposed the novel approach DMOCC employed cooperative coevolution method to evolve two decision rules combined to two sub-populations as scheduling policies. They store the non-dominated scheduling policies to an external archive for the selection of GP individuals, which was cooperative genetic operators between the two sub-population based on the crowding distance and the non-dominated rank of individuals.

Their experimental results of DMOCC evolved scheduling policies are

much better on the Pareto front multi-objective method as compared to NSGA-II and SPEA2, and also it reduced the complexity of evolving sophisticated scheduling policies by multiple scheduling decisions evolved from different sub-populations. This method can be modified to reduce the computational time by take the advantage of the parallelisation techniques [30].

Hyper-Heuristic Coevolution Multi-Objective GP

Another research paper about multi objective GP for DFJSS is Hyper-Heuristic Coevolution for Multi-objective Dynamic flexible job shop scheduling which was proposed by Zhou and Yang et al.[61]. They employ three different hyper-heuristic coevolution methods with multi objective GP approach to evolve dispatching rule for dynamic flexible job shop scheduling problem. The first method called MO-CCGP which initializes two sub-populations of individuals each two types of individual GP trees represents sequencing rule and routing rule respectively. MO-TTGP is the second method which starting one population its individual contains two decision rules by two GP trees. The last MO-GEP method employs genetic expression programming (GEP) [14] individual consisting two chromosomes for the two decision rules. Those three coevolution method all employed with multi-objective approach NSGAII and SPEA2 [61].

The final performance of those methods shows the best methods is MO-CCGP with NSGAII approach refer to Pareto front value [61]. On the other hand, some studies focus on single objective GP hyper-heuristic coevolution GP for DFJSS [48][54].

2.4.2 Cooperation Coevolution GP

Cooperation Coevolution with Genetic programming hyper-heuristic for Dynamic flexible job shop scheduling is the first piece of work to co-evolving routing and sequencing rule simultaneously[48]. The algorithm started

initialization with two populations representing routing rule and sequencing rule respectively. Then the routing rules and sequencing rules evolved separately by the GP process operator crossover, mutation and reproduction at each generation. The new generated rule evaluated by evaluate function to get fitness value. The best individual is in the final generation of each population the minimum fitness value rules [48].

Zhang et al.[54] proposed correlation coefficient-based recombinative guidance with GPHH for DFJSS is based on cooperation coevolution GPHH [48] to guide GP process genetic operators help GPHH find better scheduling heuristics of DFJSS. This algorithm finds the more useful subtrees of rule tree to guide the crossover bias better subtrees to be swiped and inherited. The difference of this proposed approach comparing with the traditional cooperation coevolution GPHH is there are two extra components: calculation of the importance of subtrees and crossover with recombinative guidance. The calculation of importance of subtrees based on Spearman correlation coefficient between subtrees and the rule tree which is the correlation of decision made by the subtree and decision made by whole tree. The recombinative guidance crossover selecting the important subtree instead of selecting the unimportant subtree will depend on the subtrees correlation value, which higher value will be high probability to be chose [54].

2.4.3 Surrogate Genetic Programming

The most time-consuming part of Genetic programming as an evolutionary algorithm is the fitness evaluation. The surrogate function was developed in order to reduce the cost of GP fitness evaluation process. The concept of surrogate model aims to minimize the number of expensive fitness evaluation in preference to the computationally cheaper function as far as possible[17]. There are couple of research about the implementation of surrogate model into GP of DFJSS.

GP employs surrogate function

Hildebrandt and Branke [17] first introduced GP surrogate function called phenotypic characterization solution for DFJSS. Their surrogate function base on GP phenotypic characterization to estimate the fitness value of new individuals which has much lower computational cost than the full simulation evaluation. The surrogate function uses the nearest neighbor regression to predict the fitness of a new individual. All the individuals will calculate their Euclidean distance between phenotypic characterization in the database. The most similar fitness of the individual will return the fitness estimate to new individual. The database contains the full fitness evaluations are the last two generations which applied for the nearest neighbor surrogate model. As applying the surrogate function in GP by measure distance on tree structure representation is much more difficult than GA's genotypic distance metric. Their model computes the phenotypic characterization using a reference rule and a set of decision situation.

The result of this surrogate function GP approach for DFJSS has reduced the fitness evaluations cost up to 80% compare to the standard GP model[17]. Also their work shows the phenotypic characterization surrogate model achieved much more accuracy than the model based on genotypic distance.

simplified surrogate assisted GP

The simplified surrogate assisted GP devoloped by Nguyen et al.[33] aims to improve the quality of the dispatching rules of DFJSS without the huge computational costs. This simplified surrogate GP model has improved some aspects of limitation in previous approach [17]. For instance, if the dimension of decision vectors doesn't contain enough good differentiating evolved rules, it might not cope some complex situation of DFJSS. Also it will not be suitable with some scheduling decisions such as due data assignment, routing, and order release [33].

Nguyen et al.[33] proposed surrogate assisted GP (SGP) based on the simplified simulation model which potentially reduced the computational cost of fitness estimation. The simplified model of simulation which is by reducing the warmup and running time of the simulation or reducing the number of machines and the number of operation per job. Nguyen et al.[33] found the most suitable simplified model is HalfShop which only half the scale of the original shop.

The process of SGP is the same normal genetic programming mainly that through the population initialisation, fitness evaluation, update best individual of GP run, and termination criteria archived to stop or iteration by apply genetic operation. The difference in SGP, the genetic operations create an intermediate population which is a large size than the original population in order to improve the chance to discover better results by increasing the diversity of population. SGP has three fitness functions:

1. $f(\Delta_i)$ the real fitness of evolved rules Δ in DFJSS .
2. $f'(\Delta_i)$ the estimated fitness obtained with the simplified model S.
3. $f_g(\Delta_i)$ the performance of rule Δ_i in a particular generation (intermediate population with a specific replication π).

As applying the large number of replications to obtain $f(\Delta_i)$ is very expensive, SGP apply one replication per generation to evaluate the quality of evolved rules $f_g(\Delta_i)$. Only the best $f_g(\Delta_i)$ will fully evaluate to obtain $f(\Delta_i)$. The newly generated rules fitness evaluated faster by applying the simplified model S for the intermediate population generated based on the generation fitness $f_g(\Delta_i)$.

The fitness $f'(\Delta_i)$ helps the SGP produce more potential rules by roughly determining rules quality, and fitness $f_g(\Delta_i)$ improve the diversity of SGP by discovering the most potential rules for full evaluations. In addition, the intermediate population is k times larger than the original population which increases the diversity of GP and has more chance to find the better rules.

Surrogate-assisted cooperative coevolution GP

The study cooperative coevolution GP with surrogate assisted model GP for DFJSS proposed by Zhou et al. [60] was to investigate and analyse the appropriate parameters of fitness evaluation of GP model. The study introduced three types coevolution GP representation tree: two sub-populations for dispatching rules routing and sequencing separated by two individual trees called CCGP, one population two parts of dispatching rule into one individual tree named TTGP, and GEP same as TTGP by sub-chromosome types tree. Those three coevolution GP employed with different configuration of surrogate-assisted simulation model to generate the DFJSS dispatching rule, CCGP with surrogate assisted model testing result is the highest performance than the other two [60].

Adaptive surrogate GP

Zhang et al.[57] proposed a surrogate-assisted GP for DFJSS called adaptive surrogate genetic programming ASGP. The main idea of this algorithm is to build a very simple surrogate model at the GP early generation then deliberately increase the surrogate models to enlarge the accuracy purpose. It is different with the stable surrogate model at the whole GP process such as HalfShop model [33]. The surrogate model at simulation jobs number and warmup jobs number are using formula:

$$Number = Number * (generation/Generation_{max})$$

That will increase simulation jobs and warmup number linearly which the surrogates range from low-fidelity to high-fidelity increase following the generation increase. [57].

Collaborative multi-fidelity surrogate GP for DFJSS introduced by Zhang et al.[53] is a novel multi-fidelity based surrogate-assisted genetic programming. It has a different method with paper[57] for surrogate model which is k designed surrogate models wtih different fidelities. The k different surrogates model for k multiple subpopulations will be evaluated

respectively, while during the evaluation process each subpopulation will share knowledge by the particular knowledge transfer algorithm to produce next generation. The key point of the approach is to transfer the knowledge between different fidelity levels of surrogate model. The subpopulation with lower fidelity (simpler problem) easily search promising individuals, and learning knowledge from the high fidelity model to improve its quality quickly. Meanwhile, the subpopulation with higher fidelity (complex problem) will speed up the convergence as well. The transfer algorithm applies to find the promising individuals for knowledge transfer which the individuals to be selected with smaller fitness values than the fitness of the knee point individual [59] as promising individuals [53].

This multifidelity-based surrogate model of GP [53] has achieved higher performance that to be able to dramatically reduce the computational time of genetic programming at all the DFJSS scenarios.

2.4.4 Multitask Genetic Programming

Multitask learning is an inductive transfer approach that improves generalization performance by leveraging the domain-specific information contained in training related tasks sharing representation simultaneously[7].

Genetic programming with multitasking method which improves the efficiency of solution for multiple dynamic flexible job shop scheduling problems with scheduling heuristics [50][52][56].

A preliminary multitask GP for DFJSS

Zhang et al.[52] introduced multitasking genetic programming hyper-heuristic to evolve DFJSS scheduling heuristics, which is the first attempt to explore the tree-based heuristic search space by evolutionary multitask GP algorithm. The different job shop tasks in DFJSS have the same objective but different configurations which can be optimised as different but related

simultaneously.

This preliminary multitask GP defines the tasks from the DFJSS with different utilisation levels but with same objective as related tasks to be optimised together[52]. The algorithm separates the entire GP population into several subpopulations which represents different utilisation levels but the same objective. The larger utilisation level contains more complex scheduling tasks. The individuals in the different subpopulations match the different tasks and evolved respectively, however, different subpopulations share their knowledge from each other.

The transfer knowledge from different tasks through the crossover GP operation which defines a transfer ratio tr to control the frequency of transfer knowledge from other subpopulations and simply transfer knowledge at each generation.

- If $rand <= tr$, The first $parent_1$ selected from current subpopulations, The second $parent_2$ selected from other subpopulations.
- If $rand > tr$, two parents selected both from the current subpopulations.

The knowledge transfer will share the knowledge for the complex task subpopulation which speeds up its convergence from simple task. Meanwhile, the simple task of subpopulation will increase the quality of individuals from others[52].

Multitask generative hyperheuristic GP

Zhang et al.[50] proposed multitask GP-based generative hyperheuristic algorithm has improved the quality of the evolved high-level heuristics which considered all the tasks in the multitask scenario.

This novel approach for DFJSS different scenarios defined the multitask problem with same objective while with the different utilization levels as homogeneous multitask. However, the multitask problem with differ-

ent objectives but with the same utilization level as heterogeneous multitask. The two types multitask method focus on DFJSS problem objective and utilization levels respectively. The individual sharing knowledge through GP crossover operation which the mechanism is mostly the same with the preliminary multitask GP [52]. This not only obtains useful information for the individual but also maintains the individuals original characteristics.

Both homogeneous and heterogeneous multitask scenarios of DFJSS implemented by the multitask generative hyper-heuristic GP can achieve better quality scheduling heuristics and get better performance. The heterogeneous multitask scenario has more potential chance to be optimized [50].

Surrogate-Assisted Multitasking GP for DFJSS

Surrogate-assisted evolutionary multitasking genetic programming is combined surrogate function [17] and multitask generative hyper-heuristic algorithm [50] to share useful knowledge between different DFJSS scheduling tasks. This novel approach[56] can improve the quality of scheduling heuristics at all the DFJSS scenarios significantly.

The algorithm of Surrogate-Assisted Multitasking GP[56] initialises the population with k subpopulations for the k tasks solution first. At GP evaluation process, the individuals are evaluated with different training instances according to the tasks in different subpopulations. Meanwhile, the phenotypic characterisations of individuals and their corresponding fitness in each subpopulation build the surrogate models. The phenotypic characterisation individual with Euclidena distance apply the KNN method to discover the most similar individual as a decision vector based on a set of decision situations [17]. There are k surrogates for each generation, and the surrogates will be rebuilt at the next generation[56]. At the evolution stage, an offspring pool is filled by $n * subpopsize$ number of offspring which are generated for each subpopulation. The final off-

spring new individuals for subpopulation for tasks by removing the duplicated individuals according to their phnotypic characterisation from the offspring pool, and selected by the ranking of the fitness which estimated by the surrogate S_i.

Zhang et al.[56] proposed the surrogate-assisted evolutionary multitasking algorithm with genetic programming hyper-heuristics for dynamic flexible job shop scheduling problems has very good performance with high convergence speed, quality of heuristics in different scenario and utilization level. Because the GP involved large number of new offspring, evaluation by surrogate model, and share individual knowledge by multitasks. There are some interesting directions which can be studied in the future, such as Gaussian processes and neural networks surrogate model.

2.4.5 Feature Selection

Feature selection process removes the irrelevant redundant and noisy attributes of data, which is an important and necessary task in machine learning and data mining [46][24]. There are broadly three categories of feature selection methods, which are filter, wrapper, and embedded [24]. However, traditional feature selection methods are not suitable applying on GPHH for DFJSS [25] because the tasks of GPHH are completely different to traditional methods, and the instances in GPHH for DFJSS are not independent on each other which is opposite to traditional machine learning and data mining [25].

This section will introduce feature selection approaches by GPHH for DFJSS which recently implemented successfully [26][25][58][55][13]. The following subsections organised as: subsection I explains a domain-knowledge-free feature ranking and selection method of GP for DFJSS [26]. Subsection II introduces an efficient practical feature selection based on niching and surrogate model algorithm [25]. Subsection III illustrates the feature selection process applied in GPHH for DFJSS with two-stage approach and its

extension version [58][55].

Domain-knowledge-free feature ranking and selection

Feature selection of GP mainly focus on genetic programming terminals selection. The important factor for GP to evolve good dispatching rules of JSS effectively is the terminals to be involved [26]. There are large number of job shop features involved in different scenarios for different DFJSS objectives optimization such as global shop-level features (current time), job-related features (processing time, due data, waiting time), and machine-related features (machine ready time, work in the queue, utilisation level). Mei et al.[26] proposes a domain-knowledge-free feature ranking and selection approach with GP for DFJSS which improves the disadvantages of feature frequency ranking method [15] and has created the basic concept for feature selection process in GP for JSS.

The domain-knowledge-free feature ranking and selection of GP algorithm [26] based on feature removal in priority function to find the irrelevant features. The feature removal in priority function Eq. 2.1:

$$g(\overrightarrow{x} \setminus x_i) = g(\overrightarrow{x}|x_i = 1). \tag{2.1}$$

Where $\overrightarrow{x}$ is the vector of feature, and x_i is the removing feature fix with 1 instead.

Then, apply this function on training instances to calculate the contribution on different objective values. The formula is Eq. 2.2:

$$\zeta(x_i; g(\overrightarrow{x})) = \phi\tau(g(\overrightarrow{x} \setminus x_i)) - \phi\tau(g(\overrightarrow{x})). \tag{2.2}$$

Where $\zeta(x_i; g(\overrightarrow{x}))$ is defined as the contribution of the feature x_i to a priority function $g(\overrightarrow{x}$ on a set of JSS instances τ. Note that if contribution $\zeta(x_i; g(\overrightarrow{x}))$ is positive value implicates that removing x_i feature from the dispatching rule will generate worse schedules for the tested JSS instances. So the more contribution value means the feature is the more relevant feature.

This algorithm of GP feature selection is an offline process, which has three stages [26]. Stage1 obtain 30 best rules by running 30 independent standard GP with entire feature set of JSS. Stage2 to select the relevant features based on contribution value as the new terminal set. The final stage conducted standard GP again by using the new terminal set.

The results of this feature selection GP [26] have shown the better performance compare with standard GP for JSS.

An efficient practical feature selection Algorithm

Mei et al.[25] proposed an efficient practical feature selection of GP algorithm for JSS by employing niching and surrogate model which is the first online process of feature selection of GP. The Niching-GP feature selection algorithm has two steps which the first step is to obtain the best diverse set of individuals by Clearing method of Niching techniques, the second step is to select the best set of terminals by the important feature depending both the individual fitness and contributions on individuals [25].

In the first step, the clearing method and selecting the best diverse set of individuals both apply the niching technique by set two parameters radius σ for controlling the range of each niche, and the capacity k to determine the number of individuals in each niche. The clearing method based on phenotypic characterisation distance measure [17] applying niching algorithm to clear the poor individual in the crowding areas. The final best diverse set of individual is also obtained by applying the niching technique from the cleared population [25].

The feature selection step select the feature subset from the final best diverse individual set base on the feature into individual fitness and contributions of individual. The contributions of individual calculated by the Domain-knowledge-free feature ranking method [26] show Eq.2.3. The individual fitness will be used for weighted majority voting process Eq.2.4 and Eq. 2.5 to select features set. The feature is to be selected when its $\zeta(t, \overrightarrow{r})$ larger than $threshold = 0.001$ and the total weight voting for it is

larger than the total weight not voting for it [25].

$$\zeta(t, \overrightarrow{r}) = fit(\overrightarrow{r} - fit(\overrightarrow{r}|t = 1), \tag{2.3}$$

$$w(r) = max\left\{\frac{g(r) - g_{min}}{g_{max} - g_{min}}, 0\right\}, \tag{2.4}$$

$$g(r) = \frac{1}{1 + fit(r)}, \tag{2.5}$$

This piratical feature selection algorithm with surrogate model with GP for JSS have achieved significantly better results than using the entire feature sets GP [25].

Feature selection approach with GPHH for DFJSS

A two stage genetic programming hyper-heuristic approach with feature selection for DFJSS proposed by Zhang et al.[58], which can be considered the first feature selection method with GPHH for DFJSS. This approach is different with classical GP separating the whole GP process into two stages. The first stage (set it before generation 50) is for feature selection GP proceeds with a niching evaluator and a surrogate model that produces two sets of informative terminals for evolving routing and sequencing rules respectively by feature selection mechanism triggered when GP at generation 50. The second stage GP main process is the same as the classical GP which the population will be evaluated without niching and surrogate model, while the mutation process will be using the two new routing and sequencing terminals which selected by the first stage.

The feature selection algorithm with GPHH for DFJSS [58] is modified from feature selection with GP for JSS approach [25]. In the Clearing method of Niching technique [25] at feature selection stage, the clearing process with a radius $\sigma = 0$ and a capacity $k = 1$ after traditional fitness evaluation. The surrogate model is aims to reduce the computation cost for evaluate the fitness objective [58].

The two-stage GP with feature selection and individual adaptation strategies approach [55] is the extension version of previous algorithm [58]. The

difference of the two implemented approach is the initialisation of stage 2, which the initialisation adapts the final population of stage 1 as part of the initial population, the other part generated from selected feature terminals randomly. Meanwhile, the mutation of stage 2 will choose selected feature terminals as well.

Both two feature selection with GPHH for DFJSS approach [58][55] shows that they have better interpretability as there are fewer unique features in the rules and smaller size of rules. They weren't compromising the performance, and be able to efficiently evolve more interpretable rules automatically [55].

Fan and Xiong et al. [13] presented a GPHH for DFJSS with extended technical precedence constraints (ETPC) approach which employs a problem specific attribute selection and individual selection with threshold and probabilistic function algorithm. For the DFJSS with ETPC in the GP terminal set, the ETPC related specific attribute will increase its weight to obtain more chance to be processed. The second method in the study is applying the threshold condition to detect inferior individuals. The individuals are applied to the predefined training instance to obtain objective value if exceeded the threshold value that individual will be inferior. Then, whether the inferior individuals terminating or not which have to check the probabilistic value by a probability function Eq. 2.6.

$$p_r = e^{\left(-\frac{C}{x}\right)}, \tag{2.6}$$

where C is a constant factor set on 5, and x is iteration number of GP generations. The probability function adapted by GP iteration which may be suitable on other GPHH technical model for DFJSS.

2.4.6 Other related work for Multi-Objective GP

Multi Objective Genetic Programming is dealing with the problem has the conflict multiple objectives. The conflict objectives of DFJSS can be max

flow-time and mean flow-time, and max-tardiness and mean-tardiness [49]. Meanwhile, the dispatching rule size and the performance of DFJSS problem are the conflict objectives as well. The more large of dispatching rule size generated by GPHH has the higher performance.

The study of GPHH generated solution of individual size with performance as multi objective GPHH for other problem such as Uncertain Capacitated Arc routing Problem (UCARP) proposed by Wang et al.[45]. Their research focus on GPHH interpretability which means the small size of individual of GP tree will have more interpretability for human to easy understanding. However, the objectives reduce the rule size and improve performance in GPHH are unbalanced. Reduce the rule size is much easy to improve the performance, both the the fitness function of GP are to minimize the objective. The α dominance [19] with archive Multi-Objective GP solved this problem very well [45].

2.5 Issues and Challenges

The genetic programming with hyper-heuristic evolved scheduling policies for dynamic flexible job shop scheduling problems have been studied from many aspects for many years. That research has achieved significantly better performances by improving effectiveness, reducing the computational time, obtaining more efficiency and better interpretability.

The GP trees representing DFJSS scheduling policies should have good interpretability which is the important aspect of judgement in machine learning. Applying suitable machine learning features not only reduces the computational cost but also improves the algorithm interpretability. Feature selection approach employed in GPHH for DFJSS has manipulated successfully [25][58][13]. In addition, feature construction method proposed by Yska et al.[47] which might need more investigation to have better understanding the individual heuristics based on GP trees structure.

Although, the GPHH generated dispatching rules can have higher com-

petitive performance than the manually designed rules by the experiences of experts. The GP dispatching rules are still difficult to clearly explain how it works in detail because of its complicated deep structures. Both feature selection and feature construction approaches can potentially reduce the size of DFJSS dispatching rule which is generated by GPHH. It is an interesting direction for further research which will reduce the computation time and concurrently increase the interpretability of GP for DFJSS.

To study the relation of dispatching rule size with performance is to discover the GP interpretability in DFJSS. The performance and rule-size are the conflict objective in DFJSS problem. Higher performance of individual will has large rule-size. Multi-objective genetic programming will trade off those problem. It will give us better understanding the relationship of dispatching rule size with performance. The small rule-size is more easy to explain the dispatching rule of DFJSS which will potentially increase interpretability of GP.

We proposed two methods of multi-objective GP for DFJSS to explore the relationship between objectives performance and rule-size. The following Chapter 3 introduce an adapting α dominance self-adaptive and archive Multi-Objective Genetic programming (αMOGP-n) for unbalance objectives of DFJSS. The Chapter 4 will demonstrate we proposed the novel method α dominance Multi Population Multi-objective Genetic Programming αMOGP-mp for unbalance objectives of DFJSS.

Chapter 3

α Dominance and Archive MOGP for DFJSS

This Chapter describes the approach by applying the α Dominance and Archive for unbalance objective Multi Objective Genetic Programming (MOGP) for Dynamic Flexible Job Shop Scheduling (DFJSS). Section 1 explains applying α Dominance for unbalance objective Multi Objective Genetic Programming. Section 2 introduces the adapting α Dominance self-adaptive and Archive MOGP algorithm αMOGP-n for DFJSS. The next Section explains the experiment design. Section 4 discusses the results and challenge of this algorithm.

3.1 α Dominance for Unbalance Objectives MOGP

Studying the GP tree will get better understanding of GP individuals to increase the interpretability of GPHH generated dispatching rule solution for DFJSS. Genetic Programming Hyper-Heuristic solving the Dynamic Flexible Job Shop Scheduling problem is to search GP tree individuals as the dispatching rule of DFJSS. The individual GP tree represented dispatching rule addresses the performance of DFJSS. The individual GP tress has the spacial tree size which is total the nodes of each GP tree called

rule-size. The good performance of solution which GP individual often contains the large rule-size. The small size of GP tree will be easy to understanding by human.

3.1.1 Unbalanced Objectives

The performance and rule-size are the conflict objectives at GPHH. Use Multi Objective Genetic Programming (MOGP) to find the relationship of performance and rule-size. It not only to trade off the performance and GP tree rule-size when we chose the solutions, but also obtains the more meaningful of solution with small rule-size GP tree.

Multi objective with performance and rule-size in DFJSS that objectives are unbalanced. Improve performance and reduce rule-size are the two objectives of MOGP. The optimisation of DFJSS's performance is much more difficult than reduce the rule-size. In DFJSS problem, the GPHH fitness function can be minimize the mean flow-time, max flow-time, mean tardiness, or max tardiness. It is NP-hard computational optimization problem by many studies [2][4][25][33][52]. Rule-size is generated DFJSS dispatching rule by GPHH which contains Sequencing rule and Routing rule. The higher performance dispatching rule has larger rule-size in DFJSS problem.

3.1.2 α Dominance

Using α dominance based multi-objective evolutionary algorithm was proposed by Ikeda et al.[19]. The concept of this algorithm is to bias the unbalance objective by introducing α dominance.
There are K-objective minimization problem

$f_1(x), ..., f_n(x) \quad x \in X.$ Let

$$g_i(x,y) = f_i(x) - f_i(y) + \sum_{j \neq i}^{1..k} \alpha_{ij}(f_j(x) - f_j(y)) \quad x, y \in X \tag{3.1}$$

$$g_i(x,y) = \sum_{j=1}^{k} \alpha_{ij}(f_j(x) - f_j(y)) \quad \text{with } \alpha_{ij} = 1. \tag{3.2}$$

When $\alpha_{ij} = 0$ indicates that sames to the traditional normal domination.

In Fig. 3.1 Pareto optimal A applys the α domination, so the C dominated by A. However, in the same situation, if it is traditional non-dominate sorting $\alpha = 0$, D isn't dominated by B.

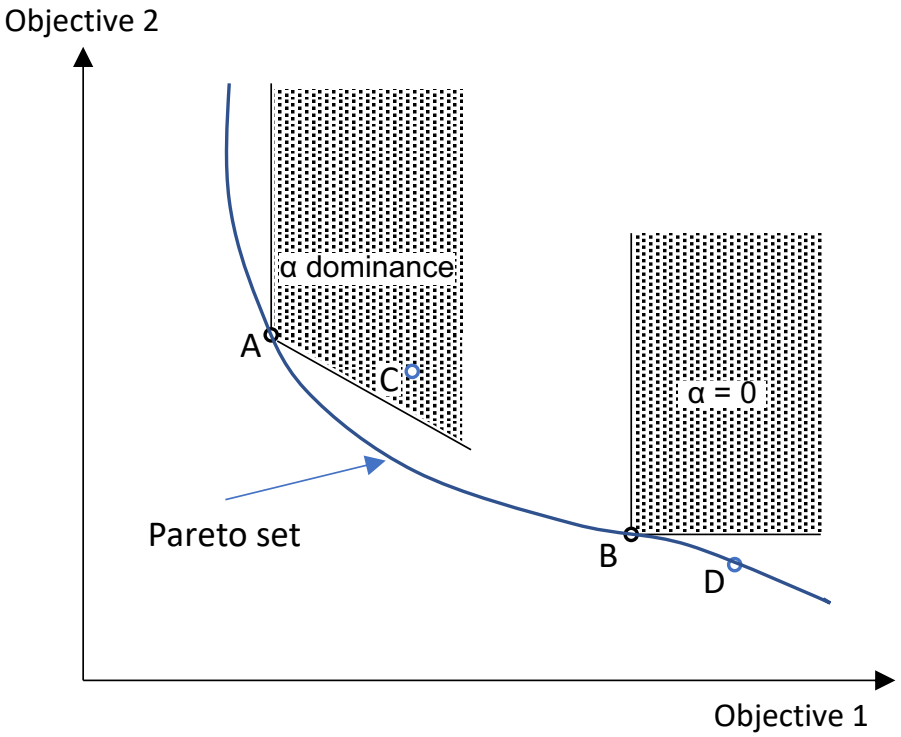

Figure 3.1: α-domination and normal domination $\alpha = 0$ in a minimization problem

3.1.3 α Dominance and Archive

The α dominance MOGP biasing on one objective will push more higher performance individuals with larger rule-size into Pareto front set. It can obtain partly good result for unbalanced objective multi objective GP problem. However, in order to keep the small rule size of individuals during the α dominance biasing to the unbalance objective, Wang et al.[45] proposed α Dominance and Archive algorithm αMOGP-a for UCARP problem. It not only bias on unbalance objective and also saved the small rule size individual into achieve as well.

3.2 α Dominance and Archive MOGP for DFJSS

We proposed an α dominance self-adaptive and arhcive algorithm αMOGP-n for DFJSS which is adapting from αMOGP-a [45] with an adjustment and normalization approaches. The algorithm is base on NSGP applying the population breeding with elite which the non-dominated sorting process only sorting with same size of original population size. The NSGP algorithm have two main calculations for each individual on non-dominate sorting process. The rank of individual that groups the individual solutions into different ranks. The rank value shows the individual how close to the Pareto front line, for example, rank 0 is the closest rank to Pareto front line. The sparsity of individual that shows the the individual solution how far the distance with his neighbours in the rank. The α dominance will only apply on individual ranking process in MOGP non-dominate sorting.

In DFJSS problem, the minimization of objective rule-size is much easy than optimization of performance. The α dominance Multi-objective GPHH for DFJSS with objectives performance and rule-size will apply the formula Eq. 3.3:

$$g(x, y) = (rs(x) - rs(y)) + \alpha * (pfm(x) - pfm(y)) \tag{3.3}$$

Where rs is for short rule-size, and pfm is for performance.

The concept of archive in MOGP is to build an archive saving the individuals from the process of genetic programming evolution. Each generation αMOGP-n extract the individuals from reference Pareto front updating into the archive.

Comparing with αMOGP-a [45], the new algorithm αMOGP-n applied two adjusted approaches:

- Using $\alpha = 0$ Pareto front as reference set to update the α, and also the individuals updating into the archive are from the reference Pareto front as well.

- Normalization of objective value during αMOGP-n non-dominate sorting process.

3.2.1 Overall Framework

Fig.3.2 shows the flowchart of the algorithm αMOGP-n. The α dominance involves the non-dominate sorting. The value of α is calculated by self-adoptive algorithm in each generation. The aim of achieve is to save the Pareto front individual with small rule-size during the evolving process. Because the small rule-size could be lose when the α dominance bias to performance objective. The archive individuals need to join to the GP breeding process producing the new generation in each GP evolution. There is particularly algorithm to check the duplicated individual between the archive and Pareto front set each generation. The criteria of duplicated based on the Euclidean distance of the two individuals. The Euclidean distance calculate the phenotypic characterization of the individuals [17][56].

However, the final Pareto Front of the solutions obtain by the traditional non-dominate sorting at αMOGP-n, because we don't want the results containing any bias of objective. Before the final traditional Non-dominate sorting, the archive individuals should be evaluating again, as

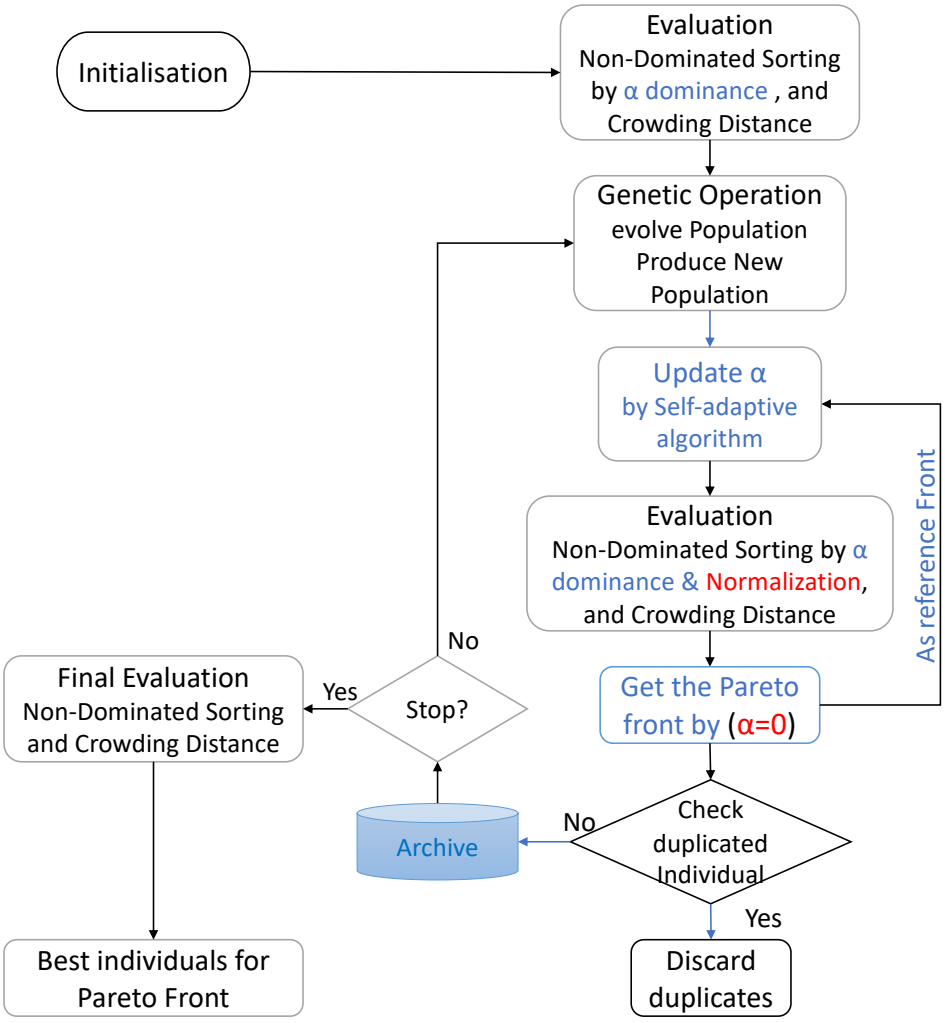

Figure 3.2: Flowchart of α-dominance and archive non-dominated Sorting with adjustment and normalization αMOGP-n for DFJSS

they were evaluated by different simulation instances from different generation of MOGP.

3.2.2 α Self-adaptive Scheme

The purpose of α Dominance Self-adaptive scheme is to update the α dominance value in each generation by following the rule-size changing. Because the unbalance objective in DFJSS multi objective problem, the α dominance value is difficult suitable for objective value change by generation and with different scenarios of DFJSS.

The more advanced method for α Dominance MOGP is using self-adaptive α value instead the stable α value[44][45]. The rule-size of GP tree changed dramatically during the GP generation evolution process for DFJSS problem. As reducing the rule-size objective in MOGP for DFJSS is very easy, the distribution of rule-size will concentrate on small tree size area after a few iterations. It causes the stable α dominance value are not fit the rule-size fitness objective.

We expect the rule-size covered wild range in order to get whole range Pareto Front solutions. The method is estimating the α dominance value which self-adaptive changed following the measurement of range of rule-size in Pareto Front set each MOGP generation. For example, we calculate the mean rule-size at each generation. If the rule-size mean value less than the median rule-size of Pareton front set, it will increase the α dominance to get more bias to performance. By contract, if the rule-size mean value larger than the median value, it will reduce the α dominance value.

We measure mean of rule-size from Pareto Front set in the current generation firstly, donated $Size_{pop}$. Notice, obtain the Pareto front set have to set α dominance to 0 in order to get reference $Pareto\ front$. If the mean value towards to minima range of population rule-size, it will increase the α value. On the other hand, if it toward to maxima range of rule-size, the α value will reduced. The procedure shows on Algorithm 1.

In the algorithm, $size_{min}, size_{max}$ are the minima and maxima of rule-size from previous generation of MOGP. The step size ϕ indicates how the degree of dominance area we need to increase or reduce for each generation.

Algorithm 1: α Self-Adaptive Algorithm

Data: $ParetoFront_{\alpha=0}, \alpha_{old}, size_{min}, size_{max}$

Result: α_{new}

$mean_size_{pop} \leftarrow ParetoFront_{\alpha=0}$;

$current_size_{min} \leftarrow ParetoFront_{\alpha=0}$;

$current_size_{max} \leftarrow ParetoFront_{\alpha=0}$;

if $current_size_{min} < size_{min}$ **then**

  $size_{min} = current_size_{min}$

end

if $current_size_{max} > size_{max}$ **then**

  $size_{max} = current_size_{max}$

end

$\theta_{old} \leftarrow arctan(\alpha_{old})$;

$\theta_{new} \leftarrow \theta_{old} + \frac{size_{min}+size_{max}-2*mean_size_{pop}}{size_{max}-size_{min}} * \phi$; /* ϕ is the step size */

$\alpha_{new} \leftarrow tan(\theta_{new})$;

return α_{new}

3.2.3 Archive Algorithm

The archive saves potential Pareto front individuals during MOGP evolution. After each generation of MOGP evolution process, the non-dominate individuals will add to the archive if they are not duplicated with inside archive individuals. The duplication of individuals dependent the value of Phenotypic Characterisation of GP rules. The Phenotypic Characterisation indicates the GP rule behavior [17][56]. The two individuals are duplicated, if their Euclidean distance based on Phenotypic Characterisation value equal to zero, the two individuals have same behavior of GP rules. However, if no duplication in the archive, the individual add to archive directly. In addition, if the duplicated individual's rule-size small than the archive's individual, the new individual replace the archive's individual.

Algorithm 2: Build Archive Algorithm

Data: The current population *pop*, The exist $Archive_{exist}$
Result: The new $Archive_{new}$

```
1  ParetoFront_{α=0} ← pop ;                    /* setting α = 0 */
2  while ind ∈ ParetoFront_{α=0} do
3      duplicate ← False;
4      while ind' ∈ Archive_{exist} do
5          distance ← ind & ind';
6          if distance = 0 then
7              duplicate = True;
8              if rule − size(ind) < rule − size(ind') then
9                  replace ind' with ind;
10             end
11             break;
12         end
13     end
14     if duplicate = false then
15         Archive_{exist}.add(ind)
16     end
17 end
18 return Archive_{new} ← Archive_{exist}
```

Algorithm 2 shows the process of building the archive. First, getting the Pareto front set from current generation population. To obtain the Pareto Front set, we have to set α dominance $\alpha = 0$ in order to get the Pareto front set. If the Pareto front individual is not duplicated from the existing archive, add it to the existing archive. Otherwise, if the rule-size of the individual less than the duplicated individual from the existing archive, replace duplicated individual form archive by the small rule-size Pareto front individual.

3.2.4 $\alpha = 0$ reference Pareto front for Update α and Archive

In the Self-adaptive algorithm 1, needs the current generation Pareto Front as a reference front to get the mean of rule-size $mean_size_{pop}$ and the current minima rule-size $current_size_{min}$ and maxima rule-size $current_size_{max}$ to calculate the new α. In addition, the individuals from Pareto Front be selected adding to archive at each generation. However, the reference Pareto front is α dominance front at αMOGP-a [45] .

We apply $\alpha = 0$ front for reference Pareto front to update α, and add to the archive at each generation instead using α dominance Pareto front set which is different with Wang et al. [45] αMOGP-a for UCARP algorithm. The comparing the reference Pareto front set size and archive size of the two algorithms in one training run on six DFJSS scenarios shows on Fig. 3.3.

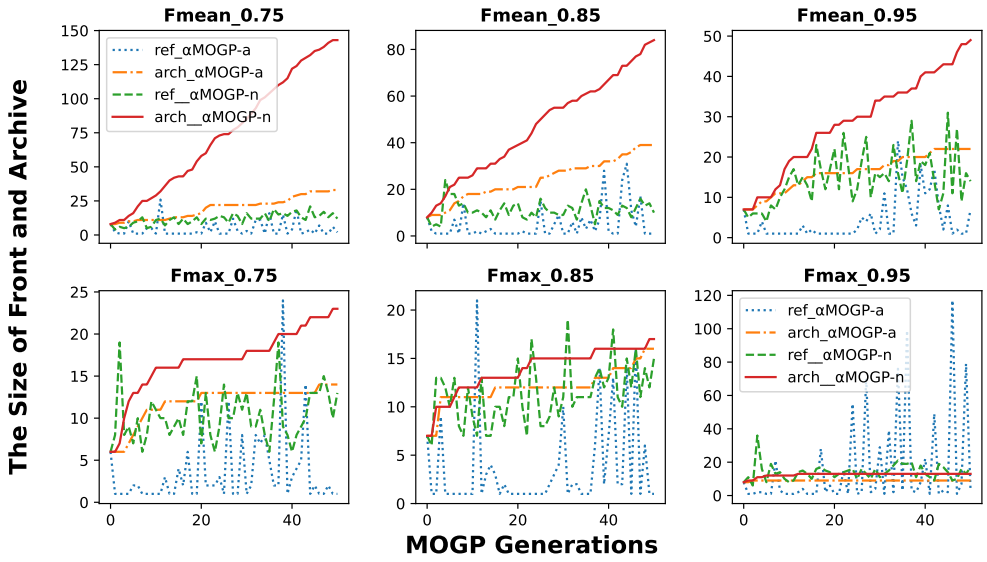

Figure 3.3: The size plots for α Update reference Pareto front set and Archive set from αMOGP-a and αMOGP-n. $ref_$: α Update reference Pareto front set size. $arch_$: Archive size

There are two advantages of this adjustment:

1. The mean of rule-size in update α schema is more accuracy as it is no bias on objective in Pareto front set from the population. Its size is increased and higher in most scenarios. In Fig. 3.3 the dash-line ref_αMOGP-n is $\alpha = 0$ Pareto front set size from αMOGP-n algorithm, and the dot-line ref_αMOGP-a is α dominance Pareto front set size from αMOGP-a. The dash-line is higher than dot-line in most scenarios.

2. There are more diversity as the archive size of αMOGP-n is larger than αMOGP-a algorithm. Fig. 3.3 the solid line arch_αMOGP-n is archive size in αMOGP-n algorithm , and the dash-line arch_αMOGP-a is archive size in αMOGP-a algorithm. In all scenarios, αMOGP-n's archive size bigger than αMOGP-a at all the generations.

3.2.5 Normalization of Objective Value

The Normalization should be applied during the non-dominate shorting process. Eq.3.3: $g(x, y) = (rs(x) - rs(y)) + \alpha * (pfm(x) - pfm(y))$ is to compare two individuals objective values with non-dominate shorting function. One objective of DFJSS GP is rule size, its value range is stable which will be $range_{rule-size} \in [2, 100]$ approximately in every iteration of MOGP. However, the performance value will dramatically changing during the generation evolution of MOGP. For example, the fitness value range of scenario mean-flowtime ($Fmean$) will be over 9000 in the first generation reduced to 1500 in the last generation approximately. In order to have good sustainably α dominance non-dominate sorting process during whole GP generation, the normalization of the objective value is requisite. We calculate each objective minima and maxima before non-dominate sorting after each generation population evaluation. Eq.3.6 function $g(x, y)$ apply for the condition of non-dominate. If $g(x, y) < 0$, individual x dominate individual y.

$$range_{rs} = rulesize_{max} - rulesize_{min} \tag{3.4}$$

$$range_{pfm} = pfm_{max} - pfm_{min} \tag{3.5}$$

$$g(x,y) = (rs(x) - rs(y))/range_{rs} + \alpha * (pfm(x) - pfm(y))/range_{pfm} \tag{3.6}$$

Eq. 3.4 get the range of rule-size donated $range_{rs}$, and Eq. 3.5 get the range of performance donated $range_{pfm}$. Eq. 3.6 is applied normalization from Eq.3.3

3.3 Experiment Design

The Multi Objective of DFJSS problem is to find the Pareto Front solutions with conflicting objective performance and rule-size. We set the performance of DFJSS problem which are mean-flowtime ($Fmean$) and max-flowtime ($Fmax$). We apply the single individual GP tree which containing two parts as the representation of DFJSS solution. They indicate the dispatching rule which are Sequencing rule and Routing rule respectively. The objective rule-size ($Rulesize$) are the sum of node numbers of GP individual.

To verify the effectiveness of the new α Dominance Multi Objective Genetic Programming αMOGP-n, will set four algorithms:

1. The traditional non-dominated sorting Multi Objective Genetic Programming NSGP.

2. The original α self-adaptive and archive algorithm αMOGP-a [45].

3. The adjustment of α self-adaptive and archive algorithm by applying the $\alpha = 0$ Pareto front as reference set for update α and archive: αMOGP-adj.

4. The adjustment and normalization of α self-adaptive and archive algorithm αMOGP-n.

There are two comparisons designed for this experiment. Firstly, comparing NSGP, αMOGP-a, and αMOGP-adj to verify the adjustment of α self-adaptive and archive algorithm. Then, the second comparison set the adjustment and normalization algorithm αMOGP-n comparing with NSGP, αMOGP-a, and αMOGP-adj.

3.3.1 DFJSS Simulation Model and Scenarios

DFJSS Simulation Model

The DFJSS simulation model, we assume there are 5000 jobs to be processed by ten machines which mostly previous research widely set for DFJSS instances [31] [52] [41]. The job arrivals by Poisson process randomly, and there are number of operations are randomly sampled form 1 to 10 for each job. Also there are set of candidate machines are randomly sampled from 1 to 10 as a subset of the 10 machines (10 is the maximal number of machines) for each operation. The operation process time on each candidate machine is randomly sampled from 1 to 99 by a uniform discrete distribution. The jobs have weight of 1,2,and 4 which are randomly sampled as distribution from 20%, 60%, 20%, respectively. Each job due data factor is setting to 1.5.

Because the simulation is with a sufficiently large amount of job arrivals to calculate the steady-state performance. We set running 1000 job arrivals on an empty shop floor as a warm-up stage, then collect the information of the next 5000 job arrivals to measure the performance of the scheduling heuristics.

DFJSS Scenarios

The DFJSS scenarios, we characterise six DFJSS scenarios by the objective performance function and utilisation level. The DFJSS performance function are mean-flowtime ($Fmean$) and max-flowtime ($Fmax$). The utilisation level is a DFJSS parameter which explains the percentage of time that the job shop is busy by controlling the job arrivals frequency in the Poisson process. Here, the experiment will set three utilisation levels (i.e., 0.75, 0.85, and 0.95) with two performance objectives ($Fmean$ and $Fmax$) which will be six different DFJSS scenarios.

3.3.2 Parameter Settings

GP terminal and function set

We use the common parameter settings of GP for DFJSS, which is shown on Table 3.1. The terminals include machine-related, job-related, and system-related information for GP to evolve dispatching rule. The function set consists of $\{+, -, *, /, max, min,\}$, where the "/" (protected division) operator returns 1 if divided by 0.

MOGP Parameter Setting

The MOGP algorithm parameter setting shows on Table 3.2. The population size is 1000. The α dominance self-adaptive start with $\alpha = 0$, and the step size (learning rate) $\phi = 90$, which is the angle of degree to bias for performance.

3.3.3 Experiment Settings

For training, each generation the training instance has a unique assigned new random seed for the DFJSS simulation [28]. After the Pareto front scheduling heuristics are trained, they are tested on 50 unseen test simu-

Table 3.1: The terminal and function sets of GP for DFJSS.

Terminals	**Description**
NIQ	The number of operations in the queue
WIQ	Current work in the queue
MWT	Waiting time of a machine
PT	Processing time of an operation
NPT	Median processing time for next operation
OWT	Waiting time of an operation
WKR	Median amount of work remaining of a job
NOR	The number of operations remaining of a job
W	Weight of a job
TIS	Time in system
Functions	$+$, $-$, $*$, $/$, max, min,
	As usual meaning, $/$ is protected division

Table 3.2: The parameter setting of MOGP.

Parameter	**Value**
Population size	1000
Number of elites	10
The number of generations	51
Initialisation method	ramped-half-and-half
Initial minimum / maximum depth	2 / 6
Maximal tree depth	8
Crossover / Mutation / Reproduction rate	80% / 15% / 5%
Tournament parent selection size	7
Terminal / non-terminal selection rate	10% / 90%

lations to generate the test performance. The comparison are between the traditional non-dominate sorting algorithm NSGP, α dominance archive algorithm αMOGP-a, the $\alpha = 0$ Pareto front as reference set adjustment

algorithm αMOGP-adj, and adjustment and normalization α dominance archive algorithm αMOGP-n. The four algorithms NSGP, αMOGP-a, αMOGP-adj, and αMOGP-n will be compared in running 30 times by each DFJSS scenario independently. We will use Hyper Volume (HV) and Inverted Generational Distance (IGD) to verify the quality of the Pareto front solutions. The results of the algorithms will use Wilcoxon statistical significance test to compare them.

3.4 Results and Discussion

The results comparison based on 30 independent runs which use Wilcoxon rank sum test with significance level of 0.05 to compare the algorithms. In the results Table, "+", "−", and "$\approx$" indicate the results are statistically significantly larger than, smaller than, or similar. In addition, there are average ranking calculated by Friedman test which shows the rank of comparison algorithms.

We use Hypervolume (HV) and Inverted Generational Distance (IGD) [38] to analysis the algorithms. The HV and IGD value calculated from the comparison algorithms together. They have same reference point $ref_point = [1, 1]$ after normalise the result objective value.

The larger HV value indicates better performance of the algorithm as it covers more area from the ref_point. The smaller IGD value shows better performance of the algorithm as it is more close the Pareto front.

3.4.1 Results Comparison

Adjustment algorithm comparison

To compare the effectiveness of NSGP, αMOGP-a, and αMOGP-adj, Table 3.3 shows comparison of test results between the three algorithms. As we can see the best effectiveness algorithm is αMOGP-adj. The algorothm

Table 3.3: MEAN(STANDARD DEVIATION) of the HV and IGD on Test Instances of NSGP, αMOGP-a, and αMOGP-adj

Scenario	**NSGP**	**αMOGP-a**	**αMOGP-adj**
		HV	
Fmean-0.75	0.986(0.001)	0.981(0.008)($\approx$)	0.989(0.003)**(+)**
Fmean-0.85	0.980(0.001)	0.977(0.013)($\approx$)	0.989(0.004)**(+)**
Fmean-0.95	0.975(0.002)	0.970(0.018)($\approx$)	0.985(0.005)**(+)**
Fmax-0.75	0.837(0.033)	0.893(0.036)**(+)**	0.893(0.025)($\approx$)
Fmax-0.85	0.844(0.040)	0.907(0.036)**(+)**	0.915(0.019)($\approx$)
Fmax-0.95	0.907(0.023)	0.925(0.043)**(+)**	0.941(0.022)($\approx$)
Average Rank	2.57	2.04	**1.39**
		IGD	
Fmean-0.75	0.127(0.025)	0.064(0.024)**(–)**	0.036(0.020)**(–)**
Fmean-0.85	0.120(0.016)	0.073(0.017)**(–)**	0.040(0.017)**(–)**
Fmean-0.95	0.098(0.025)	0.047(0.022)**(–)**	0.032(0.014)**(–)**
Fmax-0.75	0.186(0.024)	0.128(0.026)**(–)**	0.116(0.020)($\approx$)
Fmax-0.85	0.143(0.034)	0.078(0.022)**(–)**	0.081(0.021)($\approx$)
Fmax-0.95	0.168(0.027)	0.113(0.041)**(–)**	0.092(0.033)**(–)**
Average Rank	2.91	1.82	**1.28**

αMOGP-a have three significant better HV results and six signficant better IGD results than NSGP. The algorithm αMOGP-adj have three significant better HV results and four signficant better IGD results than αMOGP-a. In addition, there are no worse at all scenarios of HV results comparing αMOGP-adj with αMOGP-a. Only $Fmax$-0.85 scenario IGD value of αMOGP-adj is 0.081 slightly haigher than αMOGP-a's 0.078.

By contrast, algorithm αMOGP-adj obtained all HV and IGD significant better results than NSGP. And the average rank shows αMOGP-adj both test HV and IGD are the first rank.

Fig.3.4 shows the scatter plot diagram of medium results from test Pareto front between the three algorithm NSGP, αMOGP-a, and αMOGP-adj. As we can see both αMOGP-a and αMOGP-adj algorithm show better

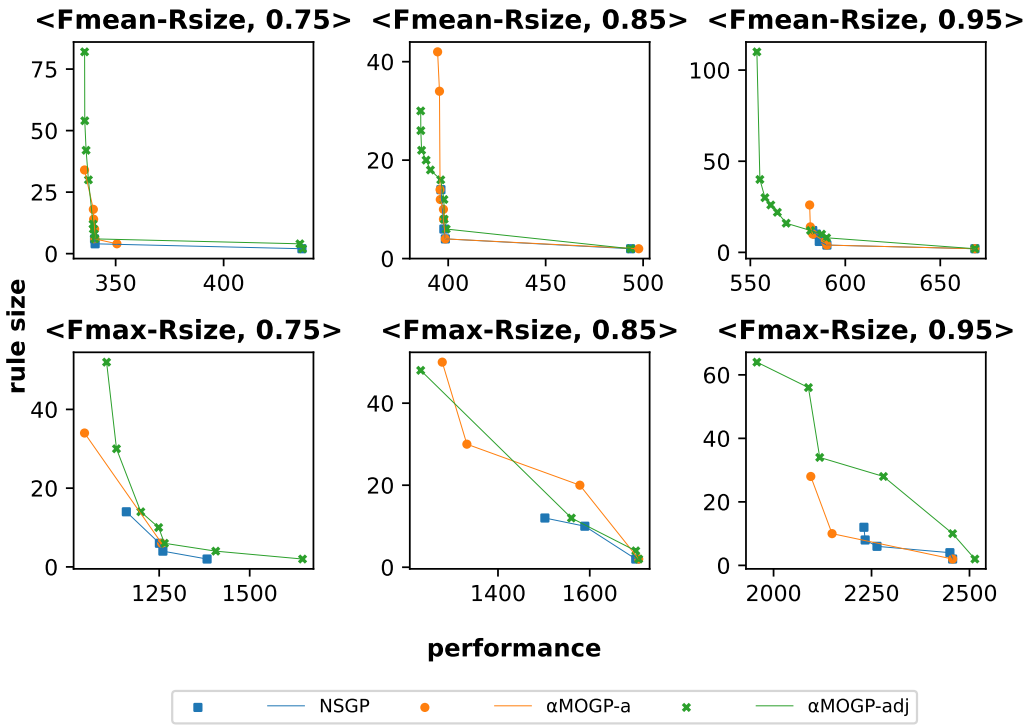

Figure 3.4: The Medium results of test Pareto front set: Traditional NSGP, α-dominance Non-dominated Sorting αMOGP, and α-dominance archive αMOGPa.

Pareto front curve than NSGP algorithm in all the scenarios. Comparing αMOGP-adj with αMOGP-a, there are two scenarios $Fmean_Rsize_0.85$ and $Fmean_Rsize_0.95$ of αMOGP-adj show better curve of Pareto front than αMOGP-a. The rest of scenarios of αMOGP-adj and αMOGP-a which Pareto front curve are very closer each other. There are some space to be improved.

Normalization and Adjustment algorithm comparison

To verify the effectiveness of proposed new algorithm αMOGP-n, the comparison is set between NSGP, αMOGP-a, αMOGP-adj, and αMOGP-n. Table 3.4 shows the Mean(Standard Deviation) of the HV and IGD value

on training and test results of four algorithms. The proposed algorithm αMOGP-n obtained the best effectiveness than others. As we can see the algorithm αMOGP-adj has mostly significant better HV and IGD results than αMOGP-a, and the αMOGP-a has mostly significant better resluts than NSGP. The αMOGP-n is slightly better than αMOGP-adj.

The Friedman test indicates αMOGP-n both training and test HV are the smallest average rank value, and test IGD is the smallest average rank. Overall, the best effectiveness algorithm is αMOGP-n.

Table 3.4: MEAN(STANDARD DEVIATION) of the HV and IGD on Training and Test of NSGP, αMOGP-a, αMOGP-adj, and αMOGP-n

Scenario	**NSGP**	**αMOGP-a**	**αMOGP-adj**	**αMOGP-n**
		Training HV		
Fmean-0.75	0.983(0.002)	0.979(0.008)**(–)**	0.988(0.003)**(+)**	0.989(0.003)(≈)
Fmean-0.85	0.975(0.001)	0.974(0.013)**(–)**	0.986(0.004)**(+)**	0.986(0.004)(≈)
Fmean-0.95	0.968(0.003)	0.968(0.016)**(–)**	0.983(0.006)**(+)**	0.984(0.005)(≈)
Fmax-0.75	0.883(0.018)	0.906(0.031)**(+)**	0.912(0.023)(≈)	0.915(0.024)(≈)
Fmax-0.85	0.851(0.034)	0.887(0.042)**(+)**	0.903(0.019)**(+)**	0.915(0.021)**(+)**
Fmax-0.95	0.879(0.025)	0.894(0.055)**(+)**	0.916(0.025)(≈)	0.907(0.033)(≈)
Average Rank	3.46	2.84	1.92	**1.78**
		Testing HV		
Fmean-0.75	0.986(0.001)	0.981(0.008)**(–)**	0.989(0.003)**(+)**	0.990(0.003)(≈)
Fmean-0.85	0.979(0.001)	0.977(0.013)**(–)**	0.989(0.004)**(+)**	0.989(0.004)(≈)
Fmean-0.95	0.975(0.002)	0.972(0.016)**(–)**	0.986(0.005)**(+)**	0.987(0.005)(≈)
Fmax-0.75	0.838(0.034)	0.896(0.036)**(+)**	0.897(0.025)(≈)	0.903(0.028)(≈)
Fmax-0.85	0.838(0.040)	0.901(0.035)**(+)**	0.910(0.019)(≈)	0.916(0.017)(≈)
Fmax-0.95	0.907(0.023)	0.925(0.043)**(+)**	0.941(0.022)(≈)	0.935(0.026)(≈)
Average Rank	3.48	2.73	1.96	**1.83**
		Training IGD		
Fmean-0.75	0.121(0.028)	0.064(0.020)**(–)**	0.035(0.016)**(–)**	0.038(0.017)(≈)
Fmean-0.85	0.150(0.016)	0.083(0.030)**(–)**	0.040(0.022)**(–)**	0.034(0.018)(≈)
Fmean-0.95	0.087(0.025)	0.040(0.019)**(–)**	0.026(0.009)**(–)**	0.026(0.012)(≈)
Fmax-0.75	0.154(0.022)	0.093(0.028)**(–)**	0.081(0.018)(≈)	0.088(0.027)(≈)
Fmax-0.85	0.170(0.030)	0.104(0.034)**(–)**	0.085(0.017)**(–)**	0.080(0.022)(≈)
Fmax-0.95	0.103(0.021)	0.086(0.032)**(–)**	0.072(0.017)(≈)	0.076(0.025)(≈)
Average Rank	3.86	2.61	1.76	**1.78**
		Test IGD		
Fmean-0.75	0.101(0.025)	0.045(0.021)**(–)**	0.026(0.016)**(–)**	0.029(0.019)(≈)
Fmean-0.85	0.150(0.017)	0.091(0.024)**(–)**	0.049(0.021)**(–)**	0.041(0.017)(≈)
Fmean-0.95	0.101(0.024)	0.048(0.021)**(–)**	0.031(0.014)**(–)**	0.030(0.012)(≈)
Fmax-0.75	0.181(0.025)	0.118(0.027)**(–)**	0.108(0.019)(≈)	0.112(0.027)(≈)
Fmax-0.85	0.218(0.034)	0.121(0.035)**(–)**	0.107(0.023)**(–)**	0.100(0.031)(≈)
Fmax-0.95	0.167(0.027)	0.112(0.041)**(–)**	0.091(0.033)(≈)	0.103(0.046)(≈)
Average Rank	3.89	2.53	1.79	**1.78**

Fig. 3.5 shows that αMOGP-n is better than others with larger HV values. And Fig. 3.6 shows that αMOGP-n is better than others with small IGD values as well. Fig. 3.7 shows the medium results of test Pareto front curve of NSGP, αMOGP-a, αMOGP-adj, and αMOGP-n. There are five of six scenarios in αMOGP-n have good Pareto front curve than other three algorithm.

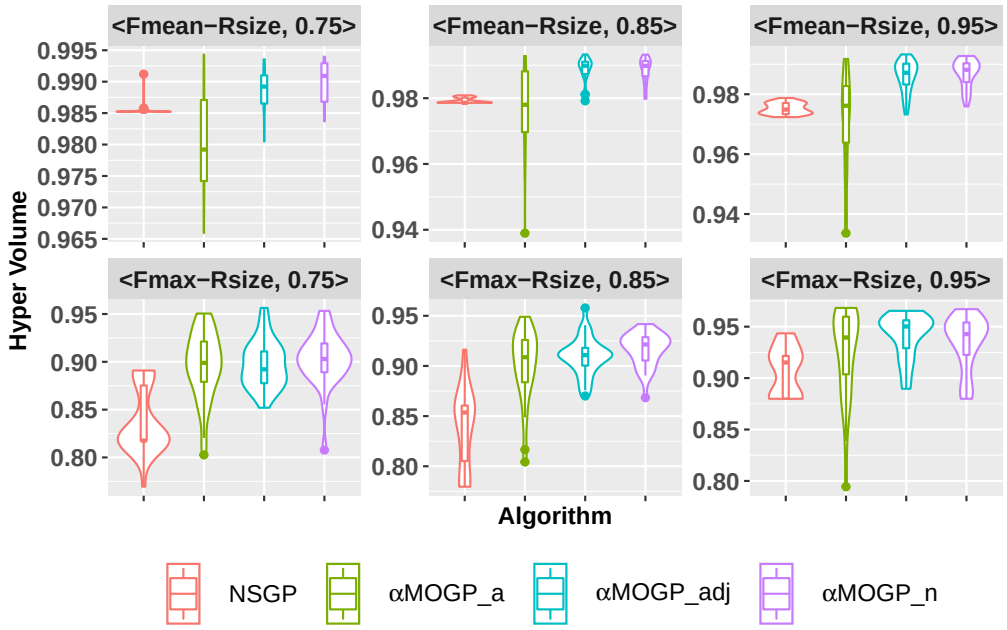

Figure 3.5: Violin plots of the HV on test instances of NSGP, αMOGP-a, αMOGP-adj, and αMOGP-n according to 30 independent runs in six scenarios.

3.4.2 Results Discussion

The comparison of four algorithms shows αMOGP-n obtained the best effectiveness. The α dominance solved the unbalanced objective in DFJSS problem. By applying α self-adaptive and archive αMOGP-a archived better effectiveness than NSGP. The adjustment by using $\alpha = 0$ Pareto front set as reference front to update α and update saving to archive which has

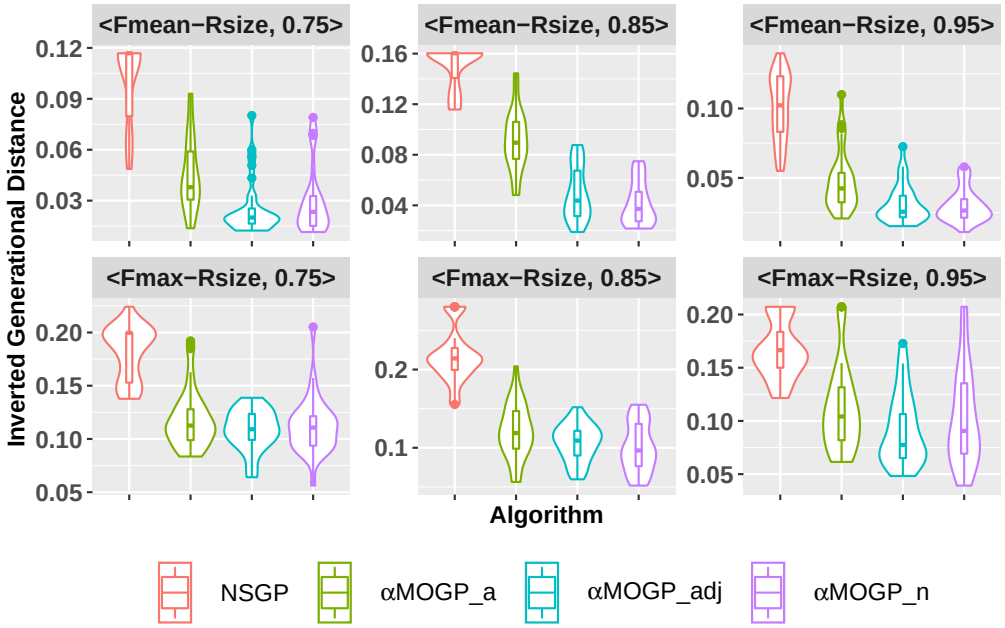

Figure 3.6: Violin plots of the IGD on test instances of NSGP, αMOGP-a, αMOGP-adj, and αMOGP-n according to 30 independent runs in six scenarios.

improved bigger effective than αMOGP-a. Although, the normalization has the slightly progressing in effective. The approach αMOGP-n achieved significant better results than other algorithms.

However, the medium test Pareto front plot diagram indicates there are two scenarios $Fmax_Rsize_0.85$ and $Fmax_Rsize_0.95$ are not showing the clearly Pareto front curve than others. It has further improvement for future study.

The archive algorithm has some issues which need further more investigation. Because we add the un-duplicated Pareto front individuals to the archive in each generation. In fact, the individuals in the archive doesn't have the chance to update directly during the whole MOGP evolution. It is only based on the algorithm find the same rule-size individual to replace the old one. Because the archive individuals need re-evaluation by same simulation instance at each generation. It will increase big computation

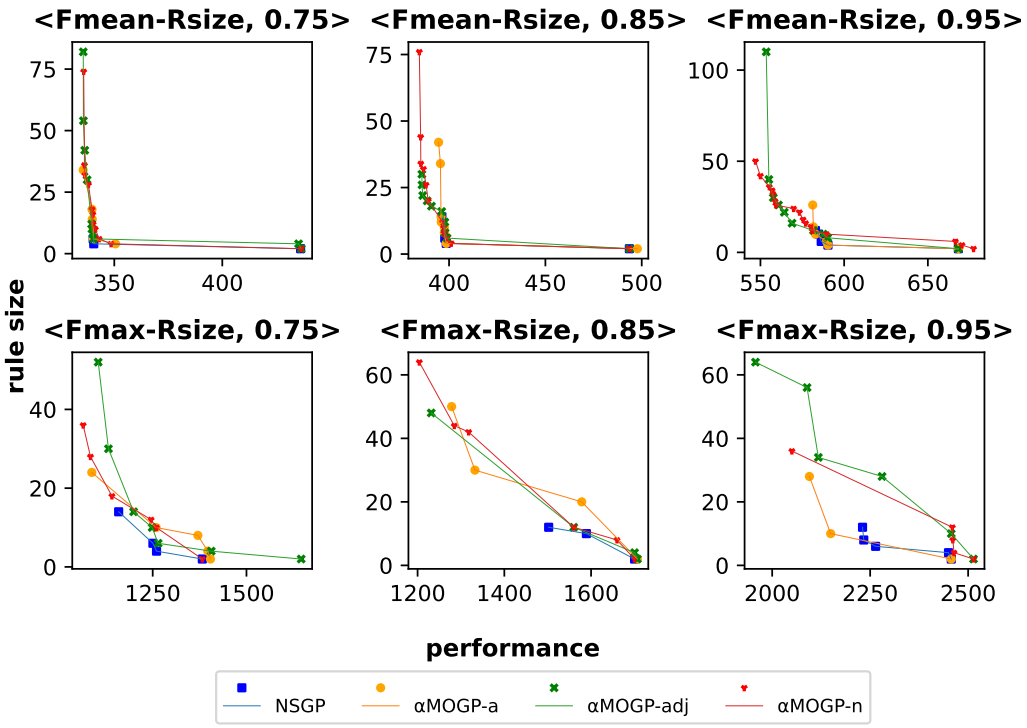

Figure 3.7: The Medium results of test Pareto front set: NSGP, αMOGP-a, αMOGP-adj, and αMOGP-n

cost. On the other hand, following the archive size increase generation by generation, it potentially reduced offspring size and reduces diversity of population of GP. It means we need to do more investigation about the self-adaptive and archive algorithm in future.

3.5 Summary

The goal of this Chapter was to develop multi-objective GP using α dominance on unbalanced objective for DFJSS. We implemented the approach α dominance and archive multi-objective GP to solve unbalanced DFJSS performance and rule-size objectives problem. The goal has been successfully achieved by proposing an more effectiveness approach α self-adaptive and

archive multi-objective GP with $\alpha = 0$ Pareto front as reference set and normalization strategy method.

This Chapter gives us the better understanding of unbalanced objectives in multi-objective GP. It has the deeply contribution helping researcher to understand α dominance on unbalanced objective and the effect archive on MOGP for DFJSS.

Although, applying $\alpha = 0$ reference Pareto front set and normalization improved the effectiveness of α self-adaptive and archive MOGP algorithm. There are some key points might be interesting for future research. For instance, the more efficient α dominance self-adaptive scheme and how to update the archive individuals by the evolving process. Furthermore, αMOGP-n put the archive individuals to the new generation breeding process directly. It is seem to reduce the population size indirectly at the later period of MOGP evolution when the archive is growth. Particularly, if the archive size getting bigger, it will affect the diversity of MOGP population.

Chapter 4

Multi Population MOGP Approach

4.1 Introduction

The Chapter 3 shows that α Dominance and Archive Multi Objective Genetic Programming approach with applying real Pareto front as the reference set and normalization strategies (αMOGP-n) archived significant better results than Traditional Multi Objective GP (NSGP), and α self-adaptive and archive MOGP (αMOGP-a) for DFJSS with unbalance objective problem. The α dominance non-dominated sorting set the bias for unbalanced objective. But the disadvantage is that bias to performance with large rule-size individuals causing lost the small rule-size solution. The archive in αMOGP-n has the capability to keep the small rule-size individuals during GP evolving process. However, the individuals in the archive are not update and fully involved the GP evolution process.

The results of αMOGP-n indicated the algorithm can find the whole range of Pareto front solution. There is improvement should be discovered by more study. We need to combine the advantages from the traditional Non-Dominated sorting algorithm NSGP which optimized the small rule-size solutions and the α Dominance algorithm αMOGP which bias to op-

timize the higher performance solutions.

The balance of convergence and diversity is the issues consistently in Evolutionary multi-objective optimization problem. Decomposition multi objective approach was successfully applied by many researches [42] [20] [27]. Multi task multi objective is also proposed by many studies to achieve good balance of convergence and diversity at evolutionary computation problem [51] [1] [49].

The overall of this chapter is to propose a novel MOGP algorithm that can use α dominance biasing to unbalanced objective, and also don't miss the extreme Pareto front. To archive this goal, the specific research objectives as following:

1. We propose a new multi population multi objective algorithm by applying α dominance and traditional non-dominated sorting approach into two populations multi objective genetic programming called αMOGP-mp. The proposed algorithm is same as a decomposition MOGP which potentially separate the MOGP to two tasks.

2. We design a scheme multi task transfer learning to share knowledge between the multi population by crossover pipeline into genetic programming process.

3. We investigate the algorithm parameter setting in order to achieve the better Pareto front solution.

4.2 Applying Multi Population αMOGP-mp

4.2.1 Overall Framework

Fig. 4.1 shows the flowchart of αMOGP-mp. The concept of this algorithm is applying two populations evolving multi-objective GP tasks respectively. It divided one population to two sub populations. Sub-population 0 implement self adaptive α dominance non-dominated sorting MOGP,

and sub-population 1 apply traditional non-dominate sorting MOGP. Finlay, both two population generate one Pareto front set of MOGP for DFJSS problem.

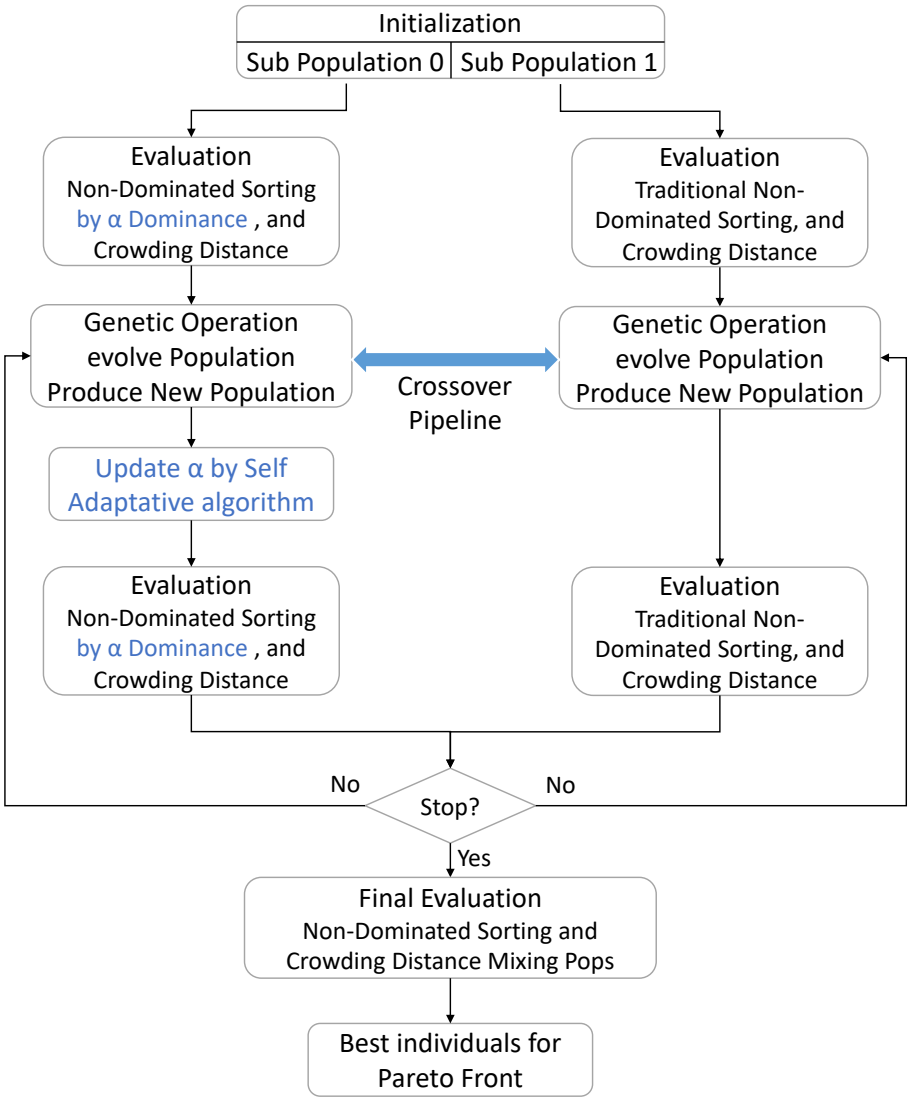

Figure 4.1: Flowchart of α-dominance Multi Population Multi Objective GP: αMOGPmp for DFJSS

4.2.2 Multipopulation Multi-Objective GP

αMOGP-mp potentially becomes decomposition multi objective genetic programming algorithm. Because the two population generate the Pareto front set into two direction. The two directions controlled by two different non dominate sorting MOGP tasks.

Initialization of αMOGP-mp start with two sub populations that manipulate two different MOGP tasks . The sub-population 0 task implement α Self-Adaptive dominance non-dominate sorting biasing to performance objective which search higher performance Pareto front individuals with large rule-size mostly. The sub-population 1 task apply traditional non-dominate sorting which concentrate the small rule-size as same as the nature unbalance objective of DFJSS problem.

The two tasks can be learning each other during the MOGP evolution process. The transfer learning will share knowledge between the two populations that will accelerate the convergence of MOGP.

4.2.3 Transfer Learning between Two Populations

The transfer learning between the two sub-populations, and sharing the knowledge between the two task which improve the convergence of MOGP. It throws crossover pipeline during MOGP breeding process. Because the two population have different boundary of rule-size from individuals which are particularly decomposed for the proposed algorithm. However, there are same rule-size individuals overlap between the two sub-populations. The knowledge share only apply on the overlap rule-size from part of two sub-populations, not on entire sub-populations. It will keep the diversity attitude of multi population of MOGP particularly.

4.2.4 Share Knowledge Crossover-Pipeline Algorithm

The share knowledge Crossover pipeline pseudo-code shows on Algorithm 3. The transfer probability $transfer_prob$ is the parameter setting for how much chance to share knowledge between two tasks. The $size_boundary$ defined by the same rule-size individuals overlap from two populations. It is between the minima rule-size objective of sub-population 0 and maxima rule-size objective of sub-population 1. Pseudo-code line 1-3 show the process.

Algorithm 3: Share knowledge Crossover-Pipeline Algorithm

Data: $transfer_prob$
Result: $Crossover\ Operation$
1 $min_rulesize_{pop0} \leftarrow sub_pop0$;
2 $max_rulesize_{pop1} \leftarrow sub_pop1$;
3 $size_boundary \leftarrow (min_rulesize_{pop0} \rightarrow max_rulesize_{pop1})$;
4 $rnd \leftarrow random()$;
5 **if** $(rnd < transfer_prob)$ **then**
6 $sub_pop0_parent0 \leftarrow sub_pop1^*$;
7 $sub_pop0_parent1 \leftarrow sub_pop0^*$;
8 $sub_pop1_parent0 \leftarrow sub_pop1^*$;
9 $sub_pop1_parent1 \leftarrow sub_pop0^*$; `// * Selection by rule-size between` $size_boundary$ `of sub_pop`
10 **else**
11 $sub_pop0_parent0 \leftarrow sub_pop0^{**}$;
12 $sub_pop0_parent1 \leftarrow sub_pop0^{**}$;
13 $sub_pop1_parent0 \leftarrow sub_pop1^{**}$;
14 $sub_pop1_parent1 \leftarrow sub_pop1^{**}$; `// ** Selection by entire sub_pop`
15 **end**

The crossover selection apply standard GP Tournament Selection method

with tournament $(size = 7)$ that parents first randomly select 7 individuals then chose the best one as the crossover parent. Pseudo-code line 4-9 indicate when random $rnd < transfer_prob$, the two parents will select from different sub-population. Meanwhile, each parent only select the individual which rule-size not over the $size_boundary$, for example, $size_boundary \in [4, 12]$, the individual rule-size between 4 to 12 to be considerably selected.

However, when random $rnd >= transfer_prob$, parents selected for crossover are from same sub-population, and they from their own entire sub-population.

4.3 Experiment Design

The multi-objective DFJSS problem defines two objective are performance and rule-size. The two DFJSS performance are mean-flowtime ($Fmean$) and max-flowtime ($Fmax$) with three utilisation levels $(0.75, 0.85, 0.95)$ combine into sex scenarios. The experiment of algorithm is to verify the effectiveness of the Multi Population MOGP: αMOGP-mp. We set the comparison with αMOGP-n (Chapter 3) algorithm and the original α self-adaptive and archive MOGP algorithm αMOGP-a [45] in six scenarios of DFJSS.

4.3.1 Simulation Model and Scenarios

The DFJSS simulation model and scenarios setting are same as Chapter 3 setting.

4.3.2 Parameter Settings

GP terminal and function set

The GP terminal and function set are same as Chapter 3 settings. It shows on Table 3.1.

MOGP-mp Parameter Setting

The MOGP-mp algorithm parameter setting shows on Table 4.1. The MOGP-mp algorithm population size is 1000 divided to two populations $sub\text{-}population0 = 900$ and $sub\text{-}population1 = 100$. Two sub-pops are different size which will explain on Section 4.3.3 population size tuning. The α dominance self-adaptive start with $\alpha = 0$, and the step size (learning rate) $\phi = 90$ applied on population 0 task.

The share knowledge transfer probability $transfer_prob = 0.1$. We have the experiment result from Chapter 3 shows that traditional non-dominate sort algorithm NSGP obtained Pareto front individuals which the rule-size $Rsize \in [2, 12]$ approximately. That indicates only around 20% rule-size range in small rule-size individuals will be found by $sub\text{-}population1$. We have tested the results of $transfer_prob = 0.1$ is better than $transfer_prob = 0.2$.

Table 4.1: The parameter setting of MOGP-mp.

Parameter	**Value**
Sub-Population 0 size	900
Sub-Population 1 size	100
Number of elites	10
The number of generations	51
Initialisation method	ramped-half-and-half
Initial minimum / maximum depth	2 / 6
Maximal tree depth	8
Crossover / Mutation / Reproduction rate	80% / 15% / 5%
Tournament parent selection size	7
Terminal / non-terminal selection rate	10% / 90%
Transfer Probability	0.1

4.3.3 Experiment Setting

For training, each generation the training instance has a unique assigned new random seed for the DFJSS simulation [28]. After the Pareto front scheduling heuristics are trained, they are tested on 50 unseen test simulations to generate the test performance. The comparison are between the traditional non-dominate sorting algorithm NSGP, adjustment and normalization α dominance archive algorithm αMOGP-n, and the novel multi population multi-objective GP αMOGP-mp.

The three algorithms αMOGP-a, αMOGP-n, and αMOGP-mp will be compared in running 30 times by each DFJSS scenario independently. We will use Hyper Volume (HV) and Inverted Generational Distance (IGD) to verify the quality of the Pareto front solutions. The results of the algorithms will use Wilcoxon statistical significance test to compare them.

Population size tuning

The two population of new algorithm αMOGP-mp represent different part of Pareto front of multi-objective of DFJSS. Because the two parts aren't simply divided equally. Although, we can equally separated population size 1000 to $(500 : 500)$. In order to obtain the best effectiveness of new algorithm αMOGP-mp, we set the experiment with population divided from $(100 : 900)$, $(200 : 800)$,...,$(500 : 500)$, until $(900 : 100)$ total 9 different sub-population running. Table 4.2 shows the effectiveness by different setting of population size of αMOGP-mp. Overall, we can find the best result is from setting sub-population0:sub-population1=900:100. Because it has average highest HV value and average lowest IGD value comparing with other settings.

Table 4.2: MEAN of the HV and IGD on Test Instances of αMOGP by 9 different sub-population size setting. **900:100** is the best setting

pop0:pop1	**Fm0.75**	**Fm0.85**	**Fm0.95**	**Fx0.75**	**Fx0.85**	**Fx0.95**
			HV			
100:900	0.936	0.875	0.703	0.714	0.601	0.489
200:800	0.940	0.891	0.760	0.753	0.688	0.562
300:700	0.940	0.905	0.788	0.795	0.721	0.656
400:600	0.945	0.905	0.787	0.826	0.768	0.704
500:500	0.948	0.918	0.817	0.851	0.766	0.669
600:400	0.955	0.917	0.838	0.862	0.783	0.754
700:300	0.954	0.925	0.837	0.871	0.777	0.732
800:200	0.950	0.929	0.851	0.862	0.789	0.738
900:100	0.954	0.926	0.850	0.876	0.804	0.776
			IGD			
100:900	0.067	0.092	0.206	0.175	0.249	0.322
200:800	0.059	0.078	0.153	0.144	0.179	0.256
300:700	0.060	0.064	0.130	0.121	0.149	0.182
400:600	0.058	0.068	0.132	0.098	0.121	0.150
500:500	0.055	0.057	0.108	0.087	0.122	0.179
600:400	0.039	0.057	0.097	0.080	0.112	0.114
700:300	0.040	0.051	0.093	0.075	0.115	0.131
800:200	0.049	0.050	0.086	0.080	0.107	0.123
900:100	0.042	0.050	0.080	0.071	0.099	0.097

4.4 Results and Discussion

The results comparison based on 30 independent runs which use Wilcoxon rank sum test with significance level of 0.05 to compare the algorithms. In the results Table, "+", "−", and "≈" indicate the results are statistically significantly larger than, smaller than, or similar. Furthermore, there are average ranking calculated by Friedman test which shows the rank of comparison algorithms.

We use Hypervolume (HV) and Inverted Generational Distance (IGD)

[38] to analysis the algorithms. The HV and IGD value calculate the three algorithms αMOGP-a, αMOGP-n, and αMOGP-mp together. They have same reference point $ref_point = [1, 1]$ after normalise the result objective value.

The larger HV value indicates better performance of the algorithm as it covers more area from the ref_point. The smaller IGD value shows better performance of the algorithm as it is more close the Pareto front.

4.4.1 Results Comparison

To compare the effectiveness of αMOGP-a, αMOGP-n, and αMOGP-mp, table 4.3 shows comparison of training and test results between the three algorithms. Noticed that in the table each result comparison is compared with its previous column. The best effectiveness algorithm is αMOGP-mp which display all the HV value and IGD value obtaining better results. We have compared αMOGP-n with αMOGP-a at Chapter 3 αMOGP-n show all the HV and IGD are significant better than αMOGP-a.

This novel algorithm αMOGP-mp shows all 6 scenarios on the HV training obtained significant better results than αMOGP-n, and 5 scenarios on HV test results are significant better. Meanwhile, The IGD training and test results have 8 significant better results. There are no worse results comparing with other two algorithms.

The Friedman test indicates that algorithm αMOGP-mp achieved the highest ranks in all the HV and IGD comparisons.

Fig. 4.2 shows that αMOGP-mp is significantly better than others with larger HV values. And Fig. 4.3 shows that αMOGP-mp is significantly better than others with small IGD values as well. Fig. 4.4 displays the medium results of test Pareto front set of three algorithm αMOGP-a, αMOGP-n, and αMOGP-mp. All the 6 scenarios plots line indicate that αMOGP-mp has the best Pareto front curve in three algorithms.

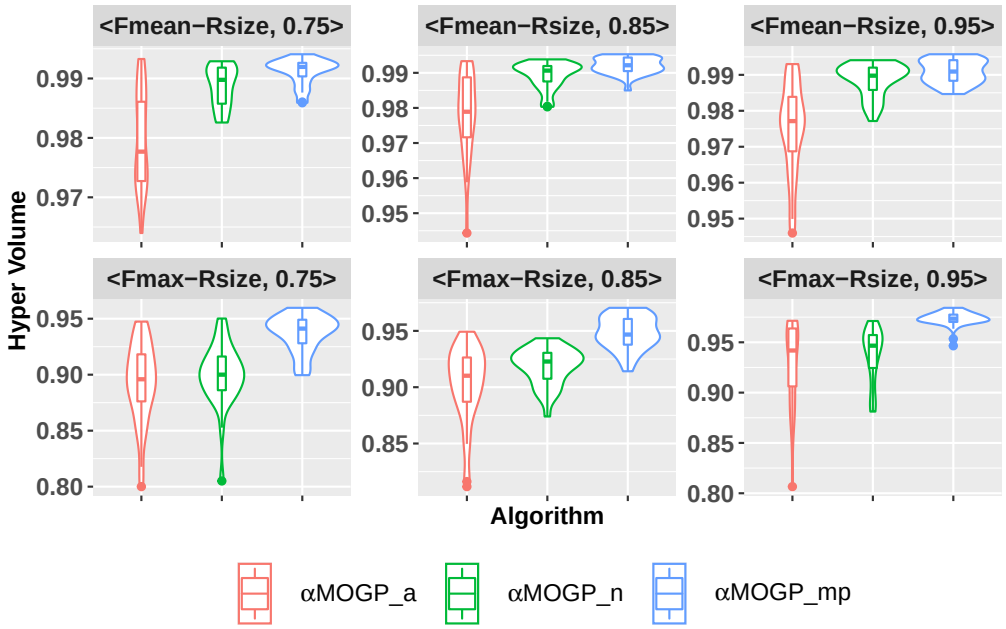

Figure 4.2: Violin plots of the HV on test instances of αMOGP-a, αMOGP-n, and αMOGP-mp, according to 30 independent runs in six scenarios.

4.4.2 Results Discussion

The α dominance multi population multi-objective genetic programming αMOGP-mp achieved magnificent better effectiveness than other algorithm for unbalanced multi objective DFJSS problem. The results of αMOGP-mp effectiveness that HV is the highest value and IGD is the lowest value. In addition, the Pareto front curve plots shows that αMOGP-mp has more individuals in the curve and makes it more smother. Also all the scenarios

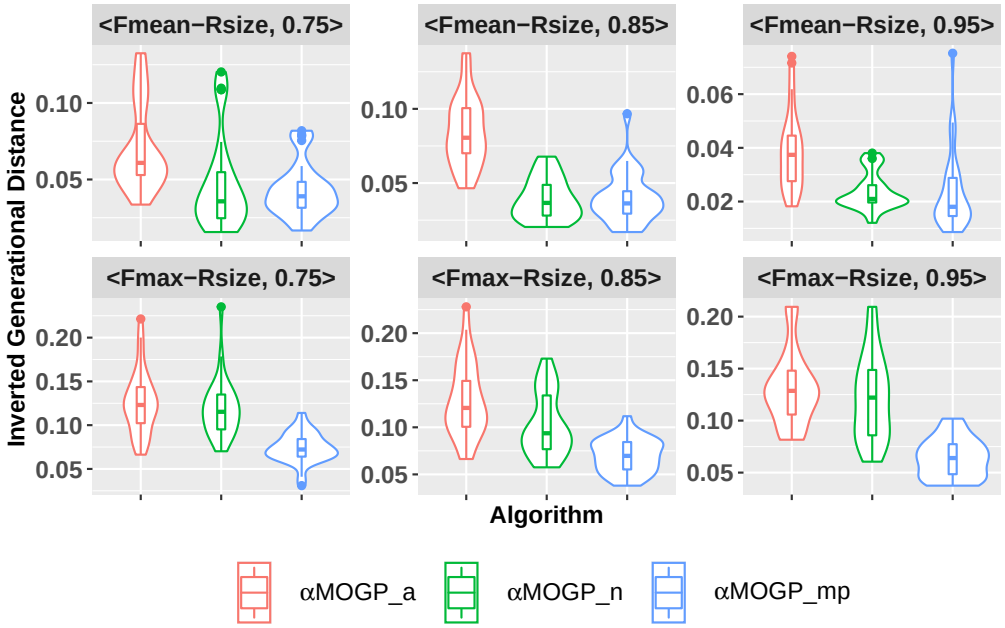

Figure 4.3: TViolin plots of the IGD on test instances of αMOGP-a, αMOGP-n, and αMOGP-mp, according to 30 independent runs in six scenarios.

Parto front indicate that αMOGP-mp archive smaller objective values on both performance and rule-size objectives.

The two tasks of multi population of αMOGP-mp is more effective than the archive algorithm. It is easily combined two advantages both from traditional non-dominate sorting and α dominance non-dominated sorting MOGP approaches. On the other hand, It keep the good diversity of MOGP than αMOGP-n. Beacuse it avoids the disadvantage by applying archive saving individuals during MOGP evolving process.

In addition, the algorithm has better convergence by implementing transfer learning between the two tasks. The share knowledge between the overlap part of two populations that makes the two different rule-size boundary heuristics solutions learning each other and accelerates the MOGP evolving process.

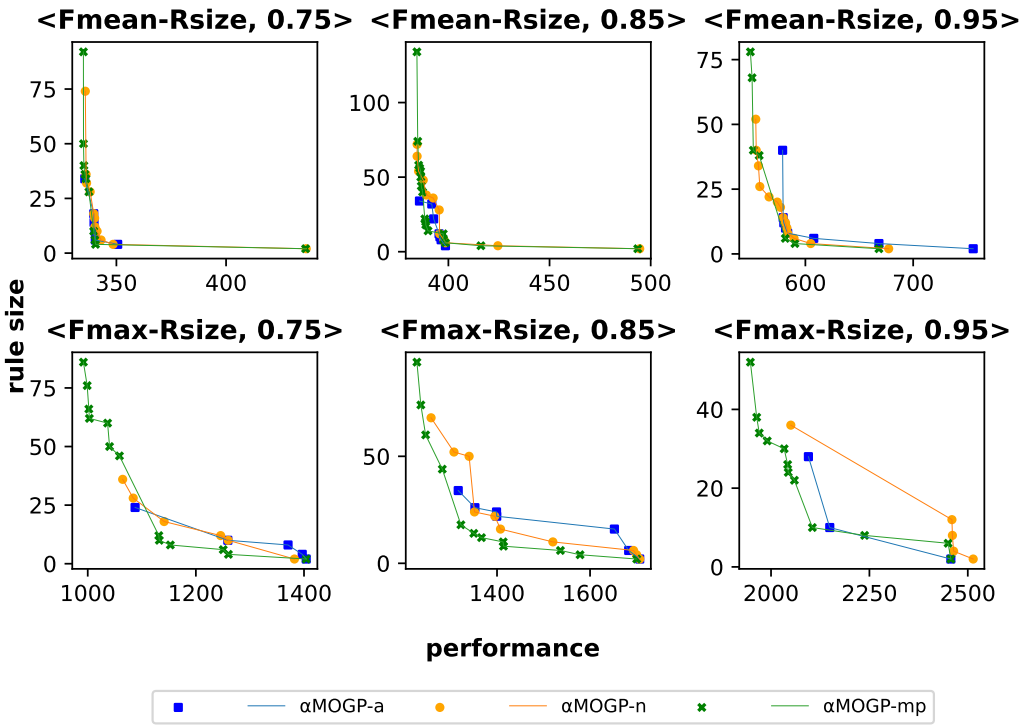

Figure 4.4: The Medium results of test Pareto front set: original α-dominance archive αMOGP-a, adjustment & normalization α-dominance archive αMOGP-n, and multi population αMOGP-mp.

4.5 Summary

The goal of this Chapter was to develop a new approach by implementing α dominance MOGP for unbalanced objective DFJSS problem. We proposed a novel algorithm α self-adaptive dominance Multi Population Multi-objective genetic programming αMOGP-mp. The goal has been successfully achieved as the novel algorithm αMOGP-mp obtained the best effectiveness approach comparing with Chapter 3 proposed algorithm αMOGP-n and original α self-adaptive dominance and archive algorithm αMOGP-a [45].

This Chapter gives us the more understanding about the α dominance multi-objective GP. We applied the multi population potentially decom-

posed multi-objective GP to multi task approach. This study indicates that decomposition multi-objective GP is very suitable for unbalanced objective problem with α dominance method. The α dominance bias the objective automatically to different GP evolving directions. Particularly, the different α value with different expectation bias level to different objective level, we can build multiple populations to decomposed many objective levels. On the other hand, we might change the GP tree constrains from the initialisation which produce the different levels of rule-size heuristics into multi population. It will find all the Pareto front individuals in different rule-size directly without the α dominance.

Because applying the transfer learning method, the α dominance multi-objective GP becomes multi-task learning. It encourages future researcher may use this method in many different optimisations studies.

Table 4.3: MEAN(STANDARD DEVIATION) of the HV and IGD on Trainging and Test Instances of αMOGP-a, αMOGP-n, and αMOGP-mp

Scenario	**αMOGP-a**	**αMOGP-n**	**αMOGP-mp**
		Training HV	
Fmean-0.75	0.978(0.008)	0.988(0.003)**(+)**	0.991(0.002)**(+)**
Fmean-0.85	0.976(0.012)	0.987(0.004)**(+)**	0.990(0.003)**(+)**
Fmean-0.95	0.972(0.012)	0.986(0.005)**(+)**	0.989(0.004)**(+)**
Fmax-0.75	0.892(0.030)	0.901(0.023)**(+)**	0.944(0.013)**(+)**
Fmax-0.85	0.878(0.041)	0.907(0.021)**(+)**	0.945(0.013)**(+)**
Fmax-0.95	0.887(0.053)	0.899(0.033)**(+)**	0.954(0.010)**(+)**
Average Rank	2.67	2.15	**1.18**
		Test HV	
Fmean-0.75	0.980(0.008)	0.989(0.003)**(+)**	0.991(0.002)**(+)**
Fmean-0.85	0.979(0.011)	0.989(0.004)**(+)**	0.992(0.002)**(+)**
Fmean-0.95	0.975(0.013)	0.989(0.005)**(+)**	0.991(0.003)(≈)
Fmax-0.75	0.893(0.036)	0.900(0.028)**(+)**	0.937(0.016)**(+)**
Fmax-0.85	0.903(0.034)	0.918(0.016)**(+)**	0.948(0.015)**(+)**
Fmax-0.95	0.930(0.041)	0.938(0.027)**(+)**	0.973(0.008)**(+)**
Average Rank	2.61	2.14	**1.25**
		Training IGD	
Fmean-0.75	0.112(0.034)	0.071(0.033)**(–)**	0.051(0.025)**(–)**
Fmean-0.85	0.113(0.037)	0.053(0.021)**(–)**	0.052(0.019)(≈)
Fmean-0.95	0.067(0.024)	0.033(0.009)**(–)**	0.027(0.012)**(–)**
Fmax-0.75	0.153(0.048)	0.146(0.050)**(–)**	0.065(0.023)**(–)**
Fmax-0.85	0.174(0.051)	0.132(0.042)**(–)**	0.070(0.019)**(–)**
Fmax-0.95	0.121(0.044)	0.113(0.042)**(–)**	0.048(0.014)**(–)**
Average Rank	2.72	2.04	**1.24**
		Test IGD	
Fmean-0.75	0.070(0.028)	0.046(0.030)**(–)**	0.043(0.017)(≈)
Fmean-0.85	0.085(0.023)	0.039(0.014)**(–)**	0.040(0.017)(≈)
Fmean-0.95	0.039(0.014)	0.024(0.007)**(–)**	0.023(0.015)(≈)
Fmax-0.75	0.126(0.036)	0.119(0.033)**(–)**	0.074(0.017)**(–)**
Fmax-0.85	0.128(0.039)	0.102(0.033)**(–)**	0.070(0.018)**(–)**
Fmax-0.95	0.131(0.036)	0.122(0.041)**(–)**	0.064(0.019)**(–)**
Average Rank	2.69	1.91	**1.39**

Chapter 5

Conclusions

This research is mainly focused on the relationship of GPHH performance and GP rule-size in DFJSS problem which related to interpretabiltity and understanding of GP trees represented DFJSS dispatching rule. The aim of this research is to develop better effectiveness method with unbalance conflicting multi-objective GP performance and rule-size MOGP algorithm for DFJSS problem.

The goal of this research has achieved better results. Both two developed new algorithms have obtained significant better effectiveness than previous studies. The proposed new α dominance self-adaptive and archive Multi-objective genetic programming with applying the $\alpha = 0$ Pareto front as the reference front for updating α and archive, and applying the normalization strategies algorithm (αMOGP-n) obtained more significant better effectiveness results than the traditional NSGP method and the original α dominance self-adaptive and archive algorithm (αMOGP-a) [45]. The proposed novel algorithm multi population multi-objective genetic grogramming with α dominance self-adaptive shame (αMOGP-mp) which has achieved the best effective results comparing with both αMOGP-a [45] and αMOGP-n algorithm.

This research achievement is not only have better understanding with α dominance in unbalance objective of MOGP, but also investigated the

influence of archive and effective implementation on MOGP. The novel multi population multi-objective GP algorithm inspired more interesting area for future study. For instance, apply decomposition with variety α dominance into multi tasks for multi objective problem.

More investigation and further research can point to some interesting directions in future. Applying the surrogate model to update archive individuals directly. Developing a more accuracy α dominance self-adaptive scheme for unbalance multi-objective. Meanwhile, the more research might need to focus on detail of MOGP Pareto front solutions to investigate the relationship between DFJSS performance and dispatching rule-size.

Bibliography

[1] ARDEH, M. A., MEI, Y., AND ZHANG, M. A novel multi-task genetic programming approach to uncertain capacitated arc routing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference* (6 2021), ACM, pp. 759–767.

[2] BLACKSTONE, J. H., PHILLIPS, D. T., AND HOGG, G. L. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research 20* (1 1982), 27–45.

[3] BRANDIMARTE, P. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research 41* (1993), 157–183.

[4] BRANKE, J., HILDEBRANDT, T., AND SCHOLZ-REITER, B. Hyper-heuristic evolution of dispatching rules: A comparison of rule representations. *Evolutionary Computation 23* (6 2015), 249–277.

[5] BRANKE, J., NGUYEN, S., PICKARDT, C. W., AND ZHANG, M. Automated design of production scheduling heuristics: A review; automated design of production scheduling heuristics: A review. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION 20* (2016).

[6] BURKE, E. K., HYDE, M. R., KENDALL, G., OCHOA, G., OZCAN, E., AND WOODWARD, J. R. Exploring hyper-heuristic methodologies with genetic programming. In *Computational intelligence*. Springer, 2009, pp. 177–201.

[7] CARUANA, R. Multitask learning. *Machine Learning 28* (1997), 41–75.

[8] CHAUDHRY, I. A., AND KHAN, A. A. A research survey: Review of flexible job shop scheduling techniques. *International Transactions in Operational Research 23*, 3 (2016).

[9] DARWISH, A., HASSANIEN, A. E., AND DAS, S. A survey of swarm and evolutionary computing approaches for deep learning. *Artificial Intelligence Review 53* (2020), 1767–1812.

[10] DURASEVIĆ, M., AND JAKOBOVIĆ, D. A survey of dispatching rules for the dynamic unrelated machines environment. *Expert Systems with Applications 113* (12 2018), 555–569.

[11] EIBEN, A. E., AND SCHOENAUER, M. Evolutionary computing. *Information Processing Letters 82* (2002), 1–6.

[12] ETHEM, A. A. *Introduction to Machine Learning*, third edition. ed. Cambridge, Massachusetts : MIT Press, 2014.

[13] FAN, H., XIONG, H., AND GOH, M. Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints. *Computers and Operations Research 134* (10 2021).

[14] FERREIRA, C. Gene expression programming: A new adaptivealgorithm for solving problems. *Complex Systems 13* (2001), 87–129.

[15] FRIEDLANDER, A., NESHATIAN, K., AND ZHANG, M. Meta-learning and feature ranking using genetic programming for classification: Variable terminal weighting. *2011 IEEE Congress of Evolutionary Computation (CEC)* (6 2011), 941–948.

[16] GAREY, M. R., JOHNSON, D. S., AND SETHI, R. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research 1* (5 1976), 117–129.

[17] HILDEBRANDT, T., AND BRANKE, J. On using surrogates with genetic programming. *Evolutionary Computation 23* (9 2015), 343–367.

[18] HOLTHAUS, O. Scheduling in job shops with machine breakdowns: an experimental study. *Computers & Industrial Engineering 36* (1 1999), 137–162.

[19] IKEDA, K., KITA, H., AND KOBAYASHI, S. Failure of pareto-based moeas: does non-dominated really mean near to optimal? *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546) 2* (2001), 957–962.

[20] ISHIBUCHI, H., SETOGUCHI, Y., MASUDA, H., AND NOJIMA, Y. Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION 21* (2017).

[21] JAYAMOHAN, M. S., AND RAJENDRAN, C. New dispatching rules for shop scheduling: A step forward. *International Journal of Production Research 38* (2 2000), 563–586.

[22] KOZA, J. R. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing 4* (1994), 87–112.

[23] KOZA, J. R., AND POLI, R. Genetic programming. In *Search methodologies*. Springer, 2005, pp. 127–164.

[24] LIU, H., AND YU, L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering 17* (4 2005), 491–502.

[25] MEI, Y., NGUYEN, S., XUE, B., AND ZHANG, M. An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. *IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE 1* (2017), 339.

[26] MEI, Y., ZHANG, M., AND NYUGEN, S. Feature selection in evolving job shop dispatching rules with genetic programming. *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (7 2016), 365–372.

[27] NGUYEN, B. H., XUE, B., ANDREAE, P., ISHIBUCHI, H., ZHANG, M., NGUYEN, B. B. H. N. H., XUE, B., ANDREAE, P., AND ARE WITH, M. Z. Multiple reference points-based decomposition for multiobjective feature selection in classification: Static and dynamic mechanisms. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION 24* (2020).

[28] NGUYEN, S., MEI, Y., AND ZHANG, M. Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems 3* (3 2017), 41–66.

[29] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. *Genetic Programming for Job Shop Scheduling*. Springer International Publishing, 06 2018, pp. 143–167.

[30] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems. *2012 IEEE Congress on Evolutionary Computation* (6 2012), 1–8.

[31] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Dynamic multi-objective job shop scheduling: A genetic programming approach. In *Automated Scheduling and Planning. Studies in Computational Intelligence*, vol. vol 505. Springer, Berlin, Heidelberg, 2013, pp. 251–282.

[32] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming.

IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION 18 (2014).

[33] NGUYEN, S., ZHANG, M., AND TAN, K. C. Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. *IEEE Transactions on Cybernetics 47* (9 2017), 2951–2965.

[34] PEARL, J. *Heuristics : intelligent search strategies for computer problem solving*. Reading, Mass. : Addison-Wesley Pub. Co., 1984.

[35] PEZZELLA, F., MORGANTI, G., AND CIASCHETTI, G. A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers and Operations Research 35*, 10 (oct 2008), 3202–3212.

[36] PINEDO, M. L. *Scheduling*, fifth edition ed. Springer International Publishing, 2016.

[37] RAJENDRAN, C., AND HOLTHAUS, O. A comparative study of dispatching rules in dynamic flowshops and jobshops. *European Journal of Operational Research 116* (7 1999), 156–170.

[38] RIQUELME, N., LUCKEN, C. V., AND BARAN, B. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)* (10 2015), IEEE, pp. 1–11.

[39] RUSSELL, S., AND NORVIG, P. *Artificial Intelligence: a Modern Approach, EBook, Global Edition : A Modern Approach*. Pearson Education, Limited, Harlow, UNITED KINGDOM, 2016.

[40] SELS, V., GHEYSEN, N., AND VANHOUCKE, M. International journal of production research a comparison of priority rules for the job shop scheduling problem under different flow time-and tardiness-related objective functions a comparison of priority rules for the job

shop scheduling problem under different flow time-and tardiness-related objective functions. *International Journal of Production Research 50* (2012), 4255–4270.

[41] SHI, G., ZHANG, F., AND MEI, Y. A novel fitness function for genetic programming in dynamic flexible job shop scheduling. In *2022 IEEE Congress on Evolutionary Computation (CEC)* (7 2022), IEEE, pp. 1–8.

[42] SHIM, V. A., TAN, K. C., AND CHEONG, C. Y. A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42* (2012).

[43] SUTTON, R. S., AND BARTO, A. G. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018.

[44] WANG, S., MEI, Y., AND ZHANG, M. A multi-objective genetic programming approach with self-adaptive α dominance to uncertain capacitated arc routing problem. In *2021 IEEE Congress on Evolutionary Computation (CEC)* (6 2021), IEEE, pp. 636–643.

[45] WANG, S., MEI, Y., AND ZHANG, M. A multi-objective genetic programming algorithm with α dominance and archive for uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation* (2022), 1–1.

[46] XUE, B., ZHANG, M., BROWNE, W. N., AND YAO, X. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation 20* (8 2016), 606–626.

[47] YSKA, D., MEI, Y., AND ZHANG, M. Feature construction in genetic programming hyper-heuristic for dynamic flexible job shop schedul-

ing. *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (7 2018), 149–150.

[48] YSKA, D., MEI, Y., AND ZHANG, M. Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10781 LNCS* (2018), 306–321. GPHH cooperative coevolution for DFJSS.

[49] ZHANG, F., MEI, Y., MEMBER, S., NGUYEN, S., ZHANG, M., ZHANG, F. Z. F., AND IS, S. N. Multitask multiobjective genetic programming for automated scheduling heuristic learning in dynamic flexible job-shop scheduling. *IEEE Transactions on Cybernetics PP* (2022).

[50] ZHANG, F., MEI, Y., NGUYEN, S., TAN, K. C., AND ZHANG, M. Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling. *IEEE Transactions on Cybernetics* (2021), 1–14.

[51] ZHANG, F., MEI, Y., NGUYEN, S., TAN, K. C., AND ZHANG, M. Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling. *IEEE Transactions on Cybernetics* (2021), 1–14.

[52] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. A preliminary approach to evolutionary multitasking for dynamic flexible job shop scheduling via genetic programming. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion* (7 2020), 107–108.

[53] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Collaborative multifidelity-based surrogate models for genetic programming in dy-

namic flexible job shop scheduling. *IEEE Transactions on Cybernetics* (2021), 1–15.

[54] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation 25* (6 2021), 552–566.

[55] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. *IEEE Transactions on Cybernetics 51* (4 2021), 1797–1811.

[56] ZHANG, F., MEI, Y., NGUYEN, S., ZHANG, M., AND TAN, K. C. Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation 25* (8 2021), 651–665.

[57] ZHANG, F., MEI, Y., AND ZHANG, M. Surrogate-assisted genetic programming for dynamic flexible job shop scheduling. *AI 2018: Advances in Artificial Intelligence 11320* (2018), 766–772.

[58] ZHANG, F., MEI, Y., AND ZHANG, M. A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling. *Proceedings of the Genetic and Evolutionary Computation Conference* (7 2019), 347–355.

[59] ZHANG, X., TIAN, Y., AND JIN, Y. A knee point-driven evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation 19* (12 2015), 761–776.

[60] ZHOU, Y., JUN YANG, J., AND HUANG, Z. Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. *International Journal of Production Research 58* (5 2020), 2561–2580.

[61] ZHOU, Y., YANG, J.-J., AND ZHENG, L.-Y. Hyper-heuristic coevolution of machine assignment and job sequencing rules for multi-objective dynamic flexible job shop scheduling. *IEEE Access 7* (2019), 68–88.