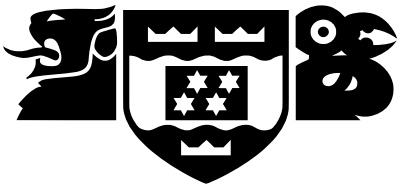
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui



School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Genetic Programming Hyper-heuristic for Dynamic Flexible Job Shop Scheduling

Fangfang Zhang

Supervisor: Yi Mei and Mengjie Zhang

Submitted in partial fulfilment of the requirements for
PhD in Computer Science.

Abstract

Dynamic flexible job shop scheduling (DFJSS), as a more realistic extension of job shop scheduling (JSS), has received a great deal of attention from academics and industry researchers due to its theoretical and applied research value. The challenge of DFJSS is how to capture both the machine assignment (routing) decision and operation sequencing (sequencing) decision simultaneously along with the new arrival jobs over time. In the previously proposed methods, dispatching rules come to the most state-of-the-art because of their low time complexity, ease of implementation and the ability to cope with both static and dynamic situations in the job shop floor. However, the dispatching rules are normally designed manually, which is very time-consuming and domain-dependent. Genetic programming (GP) has been proven to be a promising hyper-heuristic method to automatically design dispatching rules for JSS. The overall goal of this thesis is to improve the effectiveness and efficiency of GP to evolve promising and interpretable rules for solving DFJSS problems. This will be achieved by investigating new representations for GP and incorporating feature manipulation, surrogate and multi-objective technologies with different strategies.

Contents

1 Introduction **1**
- 1.1 Problem Statement . . . 1
- 1.2 Motivations . . . 2
- 1.3 Research Goals . . . 4
- 1.4 Publications . . . 9
- 1.5 Organisation . . . 9

2 Literature Review **11**
- 2.1 Job Shop Scheduling . . . 11
- 2.2 Genetic Programming . . . 14
  - 2.2.1 Representation . . . 14
  - 2.2.2 Initialisation . . . 15
  - 2.2.3 Evaluation . . . 15
  - 2.2.4 Selection . . . 16
  - 2.2.5 Evolution . . . 16
- 2.3 Existing Approaches for Job Shop Scheduling . . . 17
- 2.4 Related Work . . . 19
  - 2.4.1 GPHH for Evolving Dispatching Rules . . . 19
  - 2.4.2 Other Techniques for Evolving Dispatching Rules . . . 23
- 2.5 Summary . . . 23

3 Preliminary Work **25**
- 3.1 Genetic Programming with Multi-tree Representation . . . 25
  - 3.1.1 The Basic Crossover Operator . . . 26
  - 3.1.2 The Improved Crossover Operator . . . 26
  - 3.1.3 Experiment Design . . . 27
  - 3.1.4 Results and Analyses . . . 29
- 3.2 Surrogate-Assisted Genetic Programming . . . 32
  - 3.2.1 Adaptive Surrogates . . . 33
  - 3.2.2 Generation-range-based Surrogates . . . 33
  - 3.2.3 Experiment Design . . . 34
  - 3.2.4 Results and Analyses . . . 34
- 3.3 Summary . . . 37

4 Proposed Contributions and Project Plan **39**
- 4.1 Proposed Contributions . . . 39
- 4.2 Overview of Project Plan . . . 40
- 4.3 Project Timeline . . . 40
- 4.4 Thesis Outline . . . 41
- 4.5 Resources Required . . . 42

4.5.1 Computing Resources 42
4.5.2 Library Resources 42
4.5.3 Conference Travel Grant 42

List of Tables

2.1 The availability of job information and decision requirement in different types of JSS. 13
2.2 The characteristics of job shop scheduling problems involved in the previous researches. . . . 22

3.1 The benchmark rules used for different objectives. 27
3.2 The terminal set. 28
3.3 The average training time over 30 independent runs of the three algorithms. 31
3.4 The mean and deviation error of the compared algorithms over 30 independent runs for six scenarios. 34
3.5 The average training time (reduction) of the compared algorithms over 30 independent runs for six scenarios. 35

4.1 Phases of project plan. 40
4.2 Project timeline for the next 24 months. 41

List of Figures

2.1 An example of tree-based GP program. 15
2.2 An example of programs generated by *full* method and *grow* method. 15
2.3 An example of decision process of DFJSS. 18

3.1 Tree swapping crossover operator for multi-tree representation. 27
3.2 The boxplot of average normalized objective value obtained by sMTGP, MTGP and CCGP on test data set. 29
3.3 The convergence curves of the average best sequencing rule size (30 runs) obtained by sMTGP, MTGP and CCGP at each generation. 30
3.4 The convergence curves of the average best routing rule size (30 runs) obtained by sMTGP, MTGP and CCGP at each generation. 30
3.5 The convergence curves of average sequencing rule size (30 runs) obtained by CCGP, MTGP and sMTGP in population at each generation. 31
3.6 The convergence curves of average routing rule size (30 runs) obtained by CCGP, MTGP and sMTGP in population at each generation. 32
3.7 The variation trend of the number of jobs and warmup jobs along with generation in adaptive surrogates. 33
3.8 The number of jobs and warmup jobs in each generation in generation-range-based surrogates. 34
3.9 The convergence curves of the fitness value obtained by ASGP, GSGP and CCGP in the training processes. 36
3.10 The convergence curves of the normalized objective value obtained by ASGP, GSGP and CCGP in test processes. 36

Chapter 1

Introduction

1.1 Problem Statement

Job shop scheduling (JSS), as an important optimisation problem, has received a great deal of attention both from academics and industry researchers because it captures practical and challenging issues in real world scheduling tasks such as order picking in warehouses [48], designing manufacturing processes [33, 122] and managing grid/cloud computing [96]. In classical JSS, n jobs need to be scheduled on m machines, while trying to minimise objectives such as makespan (i.e., the total time to completely process all jobs) and tardiness (i.e., the total delay in executing certain jobs). Each job consists of a sequence of operations which need to be processed one by one. In essence, classical JSS assumes that one operation only can be processed on one specific machine. Thus, the task is to schedule the operations in the queue of the machines.

Flexible job shop scheduling (FJSS) is an extension of the well-known JSS problem, but it relaxes the restriction on candidate machines. In FJSS, one operation can be processed by any machine in a predefined set of candidate machines rather than only a single machine. Thus, FJSS includes two sub-tasks, which are machine assignment (*routing*) and operation sequencing (*sequencing*). Machine assignment is to select an appropriate machine for each operation from its candidate machines. Operation sequencing is to determine the order of processing the allocated jobs in each machine to obtain feasible and satisfactory solutions. FJSS is extremely NP-hard [11].

In *static* JSS, the jobs and their attributes are completely known in advance. The information of all the jobs is known before scheduling is performed and no new job comes in real time. In practice, however, the JSS problems are typically *dynamic* (DJSS), where the jobs arrive over time and their attributes are not known until they arrive at the shop floor. Actually, there are also other types of dynamic events in JSS problem such as machine breakdowns [118, 126], order cancellations [99], changes in due dates and arrival of urgent orders [5], but in this thesis, we only focus on new job (order) arrivals because it is the most frequent and common factor in the job shop. DJSS is strongly NP-hard [72] and cannot be solved efficiently with exact optimisation methods such as mathematical programming [103]. Many heuristic search approaches such as tabu search [98] and genetic algorithms [110] have been applied successfully to solve the JSS problems, but they are often not suitable for solving DJSS problems due to their lack of ability to reflect in time. Under this circumstance, *dispatching rules*, as priority functions, have been widely used in solving the DJSS problems [8, 38] due to their reusability and the ability to react in real time. This is because dispatching rules gradually build the schedule step by step by taking the latest information into account rather than optimising the schedules as a whole. A comprehensive comparison among a large number of dispatching rules can be found in [119].

Dynamic flexible job shop scheduling (DFJSS) takes both the characteristics of FJSS and DJSS into consideration. DFJSS is much closer to reality, but also more challenging than JSS, DJSS and static FJSS because not only does the decision of machine assignment need to be made but also the decision of operation sequencing has to be made simultaneously along with the new job arrivals over time. Naturally, two kinds of dispatching rule are needed in DFJSS, which are *routing rule* and *sequencing rule*, respectively. In this case, the quality of a schedule for the DFJSS problem depends highly on how well the routing rule and the sequencing rule work together. Since the term *dispatching rule* has been used in different contexts, it is worth highlighting that the concept of a dispatching rule in DFJSS consists of a routing rule and a sequencing rule (two kinds of rules for different uses). Briefly speaking, the routing rule will be triggered to decide where to allocate the ready operations (i.e., the first operation of a new job or the operation of a job that its preceding operations has been finished). When a machine becomes idle and its queue is not empty, the sequencing rule will be triggered to determine which operation in its queue will be chosen to process next.

However, dispatching rules are normally designed manually. In fact, the effectiveness of dispatching rules depends on the details (e.g., objective, due date tightness and utilisation) of job shop scenarios [119]. No single dispatching rules can perform well on all the job shop problems and there are many different job shop scenarios in real life. It indicates that the design process is very time-consuming and requires human expertise which is not always available. Actually, many manually designed dispatching rules are relatively simple and normally restricted to some specific assumptions [9, 35, 57] and have difficulties in handling complex practical scenarios [31, 40, 111].

Genetic Programming (GP) [64], especially tree-based GP, has been shown to be a promising hyper-heuristic approach to automatically evolving dispatching rules for solving job shop scheduling problems [40, 88, 111]. It is worth mentioning that GP has some key advantages that make it stand out among varieties of evolutionary computation approaches for solving the JSS problems. Firstly, GP has the ability of exploring both the structure and corresponding parameters with little domain knowledge due to its flexible representation. Secondly, GP provides the possibility of interpreting the evolved rules, which is very important for real-world applications. Finally, unlike in other evolutionary computation approaches, the computer programs evolved by GP can be reused in other problem instances efficiently.

The overall goal of this thesis is to investigate and enhance the ability of GP to evolve more promising dispatching rules effectively and efficiently for solving DFJSS problems according to the characteristics of DFJSS problems.

1.2 Motivations

Although GP has achieved certain success in solving JSS problems, there is little work related to DFJSS. In addition, there are still some limitations or shortages of current GP approaches, which will be illustrated below. These limitations foster the motivations to propose more effective strategies for GP to investigate and improve its ability for solving DFJSS problems.

Firstly, most current GP representations of rules are not suitable for DFJSS. For instance, the representations are only for sequencing decisions, however, both routing rule and sequencing rule are needed in DFJSS. For DFJSS, a crucial issue is how to capture both the machine assignment (routing) decision and operation sequencing (sequencing) decision simultaneously along with the new arrival jobs over time. It is noted that the interdependence between routing and sequencing should be well taken into consideration, because the quality of a schedule highly depends on how well they work together. Research in this area

is still in a very early stage. To the best of our knowledge, cooperative coevolution was firstly embedded into GP to evolve routing and sequencing rules together [129]. The proposed CCGP in [129] is the current state-of-the-art algorithm of DFJSS. In this work, there is an implicit assumption that routing rule and sequencing rule are independent and can be evolved separately. However, in DFJSS, routing rule and sequencing rule work together and they might not be totally independent. Besides, there is little research related to the terminal set of GP for DFJSS, especially the investigation of terminals for routing. It is obvious that additional terminals, which provide useful information, are needed due to the unique characteristics of the DFJSS problems. It is worth mentioning that the routing rule should use some different terminals from the sequencing rule because they perform different tasks. Existing works have investigated extensively on which terminals are effective for making sequencing decision. However, there is little work so far for making routing decision and it is still unknown which terminals are effective for routing rules. Another drawback of dispatching rules evolved using GP is their lack of a global perspective. It means the decisions are made based on the job shop information only at the decision points.

Secondly, evolving dispatching rules by the current GP approaches is too time-consuming. On one hand, a variety of attributes, which are job-related, machine-related and system-related, will help GP to evolve effective rules. On the other hand, a large number of features will enlarge the search space dramatically and irrelevant and redundant features may negatively affect the ability of exploitation of GP. In theory, the terminal set of GP should be informative, targeted and precise. Feature selection is a significant task that can be used to select informative features, which can be regarded as the terminal set of GP, but it is rarely applied to job shop scheduling. A compact and informative set of terminals will not only improve the interpretability, but also shrink the search space thus to reduce the computation time (training time). Although GP itself can automatically perform feature selection and feature construction for JSS, the ability of feature manipulation is limited. More advanced techniques are needed in this area. Mei et al. [81] proposed a feature ranking and selection approach with niching technique (i.e., clearing method) to evolve dispatching rules. The results showed that using only the selected features by proposed approach can lead to significantly better evolved rules. However, the approach followed the standard parameter setting and was only investigated in static or dynamic job shop scheduling. The parameter setting is not flexible to better tune the parameter along with the evolutionary process and there is no straightforward way to extend it to DFJSS.

Thirdly, one of main drawbacks of the GP approaches is the intensive computation time. It is well-known that the most time-consuming part of GP is the evaluation process. Simulation model is popularly used in job shop scheduling to measure the objective value and complex computational simulations will further increase computation time. The high computation time is a painful thing in the area of job shop scheduling. There are some works [39, 86, 92, 94] that apply surrogate techniques to solve JSS problem. However, all of them relate to DJSS. For DFJSS, the environment is more complex, the surrogate techniques involved might not applicable any more. In [95], a surrogate assisted model was developed based on an obtained map (i.e., growing neural gas and principle component analysis were applied to efficiently generate and update the map of explored areas based on the phenotypic characteristics of the evolved rules) to determine which heuristics to be explored in the next generation for DFJSS. However, the proposed phenotypic characteristics were applied on routing and sequencing rules separately. It cannot handle the behaviour of dispatching rules well. More advanced techniques that can evolve rules taking the routing and sequencing rules as a whole solution are needed in this area.

Finally, existing researches on multi-objective optimisation of dynamic flexible job shop scheduling are still very limited. Job shop scheduling involves simultaneous optimisation

of several incommensurable and often competing objectives such as minimising flowtime, minimising weighted tardiness and maximising the customer satisfaction. Often, there is no single optimal solution, but rather a set of alternative solutions. It is more practical and can help decision makers to make better decisions according to their preference. In addition, although the representation of GP provides a good opportunity to understand the rules, the existing works mainly focus on the performance of evolved rules. However, it is also important to interpret the evolved rules, particularly if the evolved rules will be implemented in real-world scheduling applications. There are some works aimed at improving the interpretability of evolved rules by limiting the size of rules [79, 84] and applying grammar guided GP and strongly-typed GP [45]. It has been proven that effective rules are often more complex [86]. It is obvious that it is a trade-off between the effectiveness and interpretability. The measurement, for example, the size of dispatching rules can be treated as one objective and weighted sum method can be applied to combine it with other objectives to balance the effectiveness and interpretability. However, the main drawback is that it needs to predefine the weights, which is normally not known in practice. Thus, multi-objective optimisation techniques should be considered. Moreover, it might be effective if the preference of users can be further taken into account. Further investigations in these directions need to be made.

1.3 Research Goals

To address the limitations or shortages above, the overall aim of this thesis is to develop effective and efficient GP to evolve *effective* and *interpretable* dispatching rules for DFJSS *efficiently*. We expect the new developments can extend the expressive power of GP to improve the effectiveness and interpretability of the evolved rules for DFJSS efficiently. To verify the performance of the evolved rules, we will assemble test beds based on dynamic flexible simulation model [129] for job shop scheduling. To be specific, this thesis will focus on GP for solving DFJSS problems with the following four objectives.

Objective 1: To investigate new GP representations that can effectively address complex DFJSS problems.

The objective is to explore new representations of GP to handle the routing and sequencing decisions well, thus to improve the ability of GP to evolve more effective dispatching rules. To achieve this goal, it is necessary to investigate different representations that can capture routing and sequencing tasks simultaneously. Due to the challenges of designing new representations and no prior knowledge about which terminals can benefit routing rules, this research objective contains three sub-objectives as follows.

(a) Develop a novel multi-tree representation to capture the machine assignment (routing) together with the operation sequencing (sequencing) in a single individual and propose a novel strategy to further enhance the performance of GP with multi-tree representation.

Multi-tree representation can combine routing rule and sequencing rule into one individual and these two rules can be evolved together. Thus, the interaction between them can be considered during the evolutionary process. To be specific, with this representation, each individual in the population consists of two trees. One serves as a routing rule and the other as a sequencing rule. In this way, the routing rule and sequencing rule can be evolved at the same time and the interaction between routing and sequencing rules can be handled well. The new representation is expected to tackle the interaction of routing and sequencing rules well, thus to achieve effective rules.

A comparison between the proposed algorithms (GP with multi-tree representation) and CCGP [129] which is the state-of-the-art approach for DFJSS (i.e., evolving routing and sequencing rules simultaneously), needs to be performed.

(b) Design a grammar guided GP to evolve routing and sequencing rules simultaneously and ensure that the evolutionary process respects the semantic typing of routing and sequencing rules to improve the interpretability of evolved rules.

Strongly-typed GP (STGP) [82] which is a simple example of grammar use, to semantically constrain the search space to evolve dispatching rules has been used in non-flexible job shop scheduling [29, 43, 88]. However, the sequencing rules evolved in the semantically constrained search space do not have performance that is as good as unstrained. The main reason is that the restrictions on the combinations of terminals make a huge part of the search space infeasible and the resultant search space consists of many isolated feasible regions.

This research will start with adopting the STGP with an effective strategy for constraining the combinations of terminals to evolve both effective and interpretable rules. Based on analysing the results and considering the characteristics of the DFJSS problems, a novel grammar guided GP with different grammars will be developed. The newly developed grammar representations are expected to help GP to achieve effective and interpretable dispatching rules.

A comparison among grammar guided GP, strongly-typed GP and GP with multi-tree representation will be compared with the respective of their effectiveness, efficiency and interpretability.

(c) Design new terminals according to the characteristics of DFJSS and investigate terminals with global perspective. Which terminals are effective for making routing and sequencing decisions in DFJSS will be further investigated respectively.

Current terminal set of GP in JSS is mainly explored for making sequencing decision, which may not satisfy the requirement of *sufficiency* of GP in solving DFJSS problems (also need to make routing decision). The sufficiency in GP requires the terminals are enough and informative to solve the problem combining with functions. This research will start with designing new terminals based on the characteristics of DFJSS. Then, terminals with global perspective will be explored. Firstly, we will develop additional *look forward* terminals which capture further properties of the job shop's current and potential future states, and incorporate a look-ahead element to further improve the performance. Besides, the history information might correct the previous false decisions. Thus, terminals which *look backward* in time will be developed to investigate whether they can improve the performance of GP or not. Finally, we will develop some terminals in different ways as *estimators*, such as the workload and average processing time of machines which are used to monitor the actual properties to balance the machine resources.

After the new terminals are designed, we will investigate the key terminals for making routing decisions and sequencing decisions, respectively. The new terminal sets (one for making routing decision, the other one for making sequencing decision) are expected to satisfy the requirements of GP to evolve effective dispatching rules efficiently.

Objective 2: To investigate feature manipulation (feature selection and feature construction) mechanisms to enhance the exploration and exploitation of GP approach.

The objective is to investigate a feature selection strategy to select only a small number of important and complementary features from the original features and develop a feature construction strategy that can help GP to find some useful patterns in the programs evolved by GP. This research objective contains three sub-objectives as follows.

(a) Investigate a new feature selection method using effective niching techniques to select only a small number of important and complementary features from the original features, thus to improve the effectiveness and efficiency of GP.

There are a variety of attributes in JSS problems that can provide information as terminals for GP. However, there are some drawbacks for using all these attributes. Firstly, the computation time is high when applying GP with lots of terminals. Secondly, the useless terminals and redundant terminals may mislead the exploration and exploitation of GP. In order to overcome these limitations, this research is to investigate an effective feature selection strategy for GP to solve DFJSS problems. More advanced niching strategies [69] such as adaptive choosing niching parameters [7] will be investigated and improved to maintain the diversity of population that can help select a set of diverse individuals with good performance.

According to the analysis of this research, an *offline* learning of GP with the selected features provided by the proposed feature selection approach will be developed. The offline means that the terminals will be selected before being used to initialize new population, thus to improve the performance of GP efficiently.

(b) Propose an online learning of genetic programming with the proposed feature selection approach.

It is important to consider feature selection as a part of the evolutionary process to avoid introducing feature bias into the model and make full use of the obtained structures of individuals during the feature selection phase. This research is to develop an online learning mechanism that the information of selected terminals achieved by feature selection is used to improve the ability of exploitation of GP as the evolutionary optimisation proceeds. The online learning mechanism is expected to help GP find more effective rules efficiently (i.e., with a short training time). The challenge is to determine how and when to use the terminal information obtained by feature selection mechanism.

A comparison between online learning and offline learning (developed in last sub-objective) will be performed to investigate whether the online learning mechanism has advantage over the offline learning or not.

(c) Develop feature construction mechanism to allow the involved good patterns in the programs are able to be evolved and accessed by the whole population.

In terminal set, some terminals are interdependent and some useful patterns rather than terminals themselves independently play an important roles in the evolutionary process. It is noted that the structures of automatically defined functions (ADFs) allow for sub-functions to be called by a GP individual [71], however the ADFs are associated to the specific individual rather than belonging to the population as a whole. This may limit its ability. Thus, this research is to investigate feature construction strategy to obtain some useful patterns and apply them to the whole population. The challenge is how to measure and get the useful patterns and how to use them.

Objective 3: To develop new surrogates with different strategies that can speed up the evolutionary process of GP for DFJSS without sacrificing its performance.

The objective is to adopt surrogate techniques [55] to improve the efficiency of GP without sacrificing its performance. This research can be achieved in the following four ways.

(a) Design surrogates based on problem approximation that use approximate simple problem instead of the original problem. Appropriate and simple simulation models will be investigated to reduce the computation time.

Based on a common assumption for surrogates that higher fidelity models are generally more accurate at the expense of higher computation time, different multi-fidelity models such as linear model and generation-range-based model (i.e., changing models according to a fixed interval) and other possible models will be investigated. This research is to simplify the problem to improve the efficiency of GP. The challenge is how to find an accurate and representative problem approximation for the original problem. Comparative and sensitive analyses with different settings of models are needed.

(b) Develop surrogates based on function approximation in which the fitness function is replaced by another approximate function.

In GP, fitness evaluations are often very expensive. Building an accurate approximation model for fitness itself can be very hard. However, an approximation model that keeps the relative rank of the individuals can be easier to be found. In fact, the exact fitnesses are used to rank the individuals (i.e., compare the quality of the individuals). The ranking of individuals in the population is more important than getting the exact fitnesses. Thus, this research is to explore an approximate function, which is computational cheaper and accurate in comparing the quality of individuals. The newly developed approximate function is expected to identify the relative quality of individuals accurately and efficiently. The challenge is to develop another measurement that can measure the quality of individuals accurately with less computation time.

(c) Propose surrogates based on evolutionary approximation in which fitness of a individual is estimated (not the exact fitness) by other techniques.

In general, this research is expected to reduce the computation time by reducing the frequency of fitness evaluation.

Some individuals in the population may have similar behaviour, thus the fitnesses of the individuals can be estimated according to the similar individuals to reduce the frequency of fitness evaluation. This research will start with considering the phenotype (behaviour) of the individuals to measure the similarity between individuals. The individuals with high similarity will be assigned the same fitness, and vice versa. Which individuals will be chosen as reference rules and how to estimate fitness value based on the chosen individuals are challenges here.

Then, the genotype (structure) of individuals will be considered to reduce the fitness evaluation. To be specific, the individual without important terminals will not be evaluated (the fitness is estimated as infinity), thus only evaluating relatively promising individuals. This is domain-dependent. The importance of different terminals need to be investigated firstly (obtained from objective 2). The challenge is how to measure the importance of terminals for making routing and sequencing and the interaction between them should be considered.

(d) Model management will be investigated to figure out how approximate model should be used together with the original fitness function.

In order to make sure the approximation approaches can handle the problem well, model management [55], in which the original fitness function is used to evaluate some of the individuals or all individuals in some generations, will be adopted. There are generally two kinds of approaches of model management, one is individual-based, and the other is generation-based [54].

For individual-based model management, the challenge is to decide which individuals should be controlled in each generation. For generation-based model management, in every M generations, $N(N <= M)$ generations will be managed (i.e., use the original model to evaluate all the individuals in these N generations). The challenge here is to decide the M, N and which generations will be managed specifically. In addition, adaptive model management is straightforward to think that the frequency of model management should be depend on the fidelity of the approximate model. The challenge here is how to design the adaptive model management strategies.

In general, this research is expected to enhance the approximation approaches by adopting model management. A comparison of different model management strategies will be performed.

Objective 4: To investigate multi-objective techniques to make a trade-off among different objectives.

Multi-objective aims at solving problems with more than one (i.e., two or three) conflicting objectives. Multi-objective are more realistic models for job shop scheduling optimisation problems. This research contains four sub-objectives as follows.

(a) Develop a novel multi-objective GP approach that can achieve a trade-off between two or three conflicting objectives of DFJSS.

The researches of the JSS problems with multi-objectives are very limited, especially the DFJSS. Makespan, mean flowtime, maximum tardiness and mean tardiness are potentially conflicting objectives. In this research, firstly, more conflicting objectives will be investigated in DFJSS problems. Then, an improved multi-objective GP approach will be explored to match the DFJSS problem. Finally, local search [1] and niching techniques [69] will be investigated to improve solution diversity and boost the performance. The challenge is how to match the GP-based multi-objective method well according to the characteristics of DFJSS.

(b) Investigate multi-objective techniques to balance the performance and interpretability of evolved rules.

It is mentioned in [63] that the program size constraint in GP can evolve interpretable programs but also may restrict GP to find more promising programs. The performance and the interpretability of dispatching rules are potentially conflicting objectives. In this research, firstly, a function based on rule size, fragment or interactions will be investigated to quantify interpretability. This requires manual examination the structure and the semantics of evolved dispatching rules. The challenge is how to find the factors that affect the interpretability of the evolved rules and how to quantify them. Secondly, this research will adopt multi-objective GP approach [131] to solve this multi-objective problem. The challenge is to develop a novel multi-objective GP, which can find a set of uniformly distributed solutions.

(c) Incorporate user preferences into evolutionary multi-objective optimisation.

For many real-world optimisation problems, there are normally multiple conflicting objectives. It is difficult to weight the different criteria exactly before alternatives are

known. Evolutionary multi-objective optimisation usually solves this by searching for the whole Pareto-optimal front of solutions. However, the user typically has a vague idea about what kind of solutions (knowledge) might be preferred. This research aims at considering such knowledge into evolutionary multi-objective algorithms to focus the search on the most interesting areas of the Pareto-optimal front. This research is expected to speed up the search and conduct a more fine-grained selection of alternatives in the most relevant parts of the Pareto-optimal front. The challenge is how to extract the preference of users and how to incorporate it into the evolutionary process.

(d) (optional) Design a novel many-objective GP approach that can deal with many conflicting objectives for DFJSS.

Many-objective refers to a class of optimisation that have more than three objectives. In real life, normally, there are more than three objectives we need to balance when making a decision. NSGA-III is the state-of-the-art method [68] to solve many-objective problems. This research will start with developing a new GP based approach incorporating with NSGA-III to solve many-objective problems in DFJSS. However, the objective space of NSGA-III is defined with uniformly distributed reference points. Thus, there will be few reference points with no pareto optimal solutions associated with them. Especially, in the cases with discrete and non-uniformly pareto front (i.e., resulting in many usefulness reference points during evolution). The challenge is how to achieve a better match in between reference points and sampled solutions.

1.4 Publications

In the first stage of this research, some studies on the application of GP for dynamic flexible job shop scheduling have been conducted. GP with a multi-tree representation was developed for evolving routing and sequencing rules simultaneously and a surrogate-assisted GP was proposed to reduce the computation time for DFJSS. In addition, a novel genetic programming method with grammar representation was developed to make full use of the information of all terminals. Following are a collection of research papers from the preliminary studies.

- Fangfang Zhang, Yi Mei, and Mengjie Zhang, "Genetic programming with multi-tree representation for dynamic flexible job shop scheduling" Accepted as a full paper by *The Australasian Joint Conference on Artificial Intelligence (AI 2018)*
- Fangfang Zhang, Yi Mei, and Mengjie Zhang, "Surrogate-assisted genetic programming for dynamic flexible job shop scheduling" Accepted as a short paper by *The Australasian Joint Conference on Artificial Intelligence (AI 2018)*
- Fangfang Zhang, Yi Mei, and Mengjie Zhang, "A novel genetic programming approach to evolving dispatching rules for dynamic flexible job shop scheduling" Submitted to *The AAAI Conference on Artificial Intelligence (AAAI 2019)*

1.5 Organisation

The remainder of this proposal is organised as follows. Chapter 2 presents a literature review about the job shop scheduling problem, especially the dynamic flexible job shop scheduling problem, and the methodologies for solving it. In addition, genetic programming and related work about using GP and other techniques to evolve dispatching rules to

solve job shop scheduling problems are also shown in this chapter. The preliminary work is presented in Chapter 3. In the first work, GP with multi-tree representation is introduced for evolving routing and sequencing rules at the same time for DFJSS. In the second work, surrogate-assisted GP is used to reduce the computation time for DFJSS. Some results are also provided to measure the effectiveness and efficiency of the evolved rules. Finally, Chapter 4 provides a detailed research plan and a timeline of the tasks for this thesis.

Chapter 2

Literature Review

This chapter begins by introducing different types of the job shop scheduling (JSS) problems such as classical JSS, static/dynamic JSS and non-flexible/flexible JSS with a special focus on dynamic flexible job shop scheduling (DFJSS). A review of the approaches to solving job shop scheduling problems is given in this chapter to provide a summary of the research in this field. The introduction of genetic programming (GP) and the applications of using GP and other techniques to learn dispatching rules to solve JSS are also presented.

2.1 Job Shop Scheduling

The rapid development of globalisation and information technologies has made our world a Global Village, where the interest of countries is interconnected. The core of the connection highly relies on international trade. Thus, it brings more opportunities and also thrives competition among companies. The study of allocating the jobs to machines and determining the order of processing the allocated jobs on each machine to optimise criteria such as flowtime, tardiness or customer satisfaction will benefit the companies by increasing their efficiency, profit or reputation.

Classical Job Shop Scheduling

In the basic version of JSS problem, n jobs need to be scheduled on m machines, while trying to minimise the makespan. In this standard, for each job, there is a set of operations which need to be executed in a specific order and each operation can be processed at a specified machine. In essence, the former JSS is based on the assumption that only one machine is able to run a particular operation. Some basic or commonly used definitions and notations that will be used in this study are described as follows.

parameters:

- n: the number of jobs in the job shop
- m: the number of machines in the job shop
- i: index of job
- j: index of operation
- l_j: the number of operations for each job, $l_j <= m$
- $O_{ij} = (O_{i1}, O_{i2}, ..., O_{il_j})$: the set of operations of job_i

- w_i: the weights of job_i
- d_j: the due date of job_i
- $\delta(O_{ij})$: the processing time of operation O_{ij}
- $m(O_{ij})$: the machine that processes operation O_{ij}
- $\pi(O_{ij})$: the optional machines of O_{ij}, $\pi(O_{ij}) \in M$. This parameter is used in flexible JSS which will be described later.

variables:

- C_i: the completion time of job_i
- $r(O_{ij})$: the release time of operation O_{ij}. That is, the time that jth operation of job_i is allowed to start. In our research, for the first operation of each job, it is set to zero. Otherwise, it is set to the completion time of its preceding operation.

constraints:

- The $(j+1)th$ operation of job_i (denotes by $O_{i(j+1)}$) can only be processed after its preceding operation O_{ij} has been processed.
- Each machine can only process at most one operation at a time.
- The scheduling is non-preemptive, i.e. the processing of an operation cannot be stopped or paused until it is completed.

objectives:

- Minimisation of makespan: $C_{max} = max\{C_1, C_i, ..., C_n\}$
- Minimisation of mean tardiness: $\frac{\sum_{i=1}^{n} max\{C_i - d_i, 0\}}{n}$
- Minimisation the maximum flowtime: $max_{i \in \{1,2,...,n\}}\{C_i - r_i\}$
- Minimisation of mean flowtime: $\frac{\sum_{i=1}^{n} \{C_i - r_i\}}{n}$
- Minimisation of mean weighted flowtime: $\frac{\sum_{i=1}^{n} w_i * \{C_i - r_i\}}{n}$

The last three objectives are used as the measure of schedules obtained in this research.

Dynamic Job Shop Scheduling

Static job shop scheduling implies that the information of jobs is known when we make decision. Actually, the classical JSS problem is static JSS. With available information, this makes it easier for us to get a promising schedule. However, this is not the normal case. In practice, the environment is usually dynamic and jobs arrive in the job shop over time without prior information. Dynamic job shop scheduling (DJSS) is used for considering this situation.

Whether a job shop scheduling problem is static or dynamic depends on the information of jobs is known in advance or not. For dynamic job shop scheduling, jobs arrive in the job shop over time and their information can only be known when they arrive. More specially, in DJSS, at any given time point, only the information of the jobs that have arrived before the current time is available, while the future jobs are still unknown. This characteristic is opposite to static job shop scheduling.

Flexible Job Shop Scheduling

The flexible job shop scheduling (FJSS) problem is an extension to classical JSS problem, which breaks the above restriction through the constraint of resources uniqueness. That is, one operation can be processed on more than one machine, which leads itself to a more complex problem. In order to tackle the FJSS problem, two decisions, which are a machine-specific decision (*routing*) and a job-specific decision (*sequencing*), have to be made. The machine-specific decision is to allocate a ready operation to an appropriate machine while the job-specific decision aims to select one operation as the next to be processed when a machine becomes idle and there are operations in its queue. FJSS is NP-hard [11]. Dispatching rule, as priority function, is the state-of-the-art technique to solve the FJSS problems. Two types of dispatching rules (*routing rule* and *sequencing rule*) are needed here. Routing rule and sequencing rule are used to priority the machines and operations (jobs) for making machine-specific and job-specific decisions, respectively.

Given a set of machines $M = \{m_1, m_2, ..., m_m\}$ and jobs $J = \{j_1, j_2, ..., j_n\}$, FJSS aims to determine which machine to process a particular job and which job will be chosen to process next by a particular machine. To be specific, each job J_i has a sequence of l_j ($l_j <= m$) operations $O_{ij} = (O_{i1}, O_{i2}, ..., O_{il_j})$. Each operation O_{ij} can only be processed by one of its own optional machines $\pi(O_{ij})$ and its processing time $\delta(O_{ij})$ depends on the machine that processes it.

Except for the constraints mentioned above, for FJSS, there is a special constraint.

- Each operation O_{ij} can be processed on one of the corresponding set of machines $\pi(O_{ij}) \in M$ with $\delta(O_{ij})$.

Dynamic Flexible Job Shop Scheduling

Dynamic flexible job shop scheduling (DFJSS) considers both the characteristics of FJSS and DJSS problems. DFJSS is more challenging since not only does the specific machine need to be determined but also the processing sequence of operations must be decided simultaneously along with the new arrival jobs over time. Thus, the first challenge is how to evolve the routing and sequencing at the same time. Another challenge is how to consider the interaction between routing and sequencing well. This is very important because the performance of schedule highly depends on how well the routing and sequencing rule work together. DFJSS is strongly NP-hard [56].

Table 2.1: The availability of job information and decision requirement in different types of JSS.

	Problem	JSS	DJSS	FJSS	DFJSS
Job Information	known	√		√	
	unknown		√		√
Decision	routing			√	√
	sequencing	√	√	√	√

In general, DJSS and FJSS are variants of classical JSS. Further, DFJSS are the combination of DJSS and FJSS to some extent. The details about the availability of job information and the needed decisions of different types of JSS are shown in Table 2.1. The table 2.1 shows that the DFJSS is the most complex one in the four types of JSS with two decisions under unknown job information.

2.2 Genetic Programming

GP is a domain-independent approach that can automatically generates computer programs to solve problems. Different from genetic algorithm [34], GP has a variable length representation rather than a fixed length string of genes (e.g. bits, real numbers or symbols). As a population based evolutionary computation technique, the main steps of GP are shown in Algorithm 1. In the rest of this section, the main concepts of GP are presented to show how GP works.

Algorithm 1: Pseudo-code of GP

```
// Initialization
while N_ind < Popsize do
    foreach individual
        Randomly initialize each individual by ramp half-and-half
end
gen = 0
while gen < maxGen do
    Evaluation: Evaluate the individuals by fitness function
    Selection: Select individuals from population based on fitness value
    Evolution: Generate new population by applying genetic operators
        reproduction: copy the selected individuals to new population
        crossover: create offsprings by swapping chosen subtrees of parents
        mutation: create an offspring by mutating a chosen part of a parent
    gen = gen + 1
end
return best individual
```

2.2.1 Representation

Tree based GP is widely used to solve complex problems with the programs using tree-based representation in its population. Figure 2.1 shows an example of a tree-based representation of the program $5x + max(y - x, 0)$. In this program, the terminals consist of the variables $\{x, y\}$ and two constants $\{5, 0\}$ and the functions compose of $\{\times, +, -, max\}$. The terminals are the leaves of the tree while the functions cannot be located at the leaves of the trees. In GP, the collections of the terminals and functions are called *terminal set* and *function set*, respectively. Obviously, the program is the combination of the components in terminal set and function set. In tree-based GP, one individual can contain more than one tree. This special trait can be utilized to tackle different problems at the same time. To be specific, each tree can be used for solving one sub-problem and then the whole problem will be solved by combining all the sub-solutions together with some relationships. Except for tree-based GP, there are also some other representations such as linear genetic programming (LGP) [4, 101] and graph-based genetic programming (GGP) that used in GP. LGP uses a specific linear representation of computer programs in which an imperative language, like C, are evolved. The programs of GGP are regarded as N nodes in a directed graph, with as many as N arcs going out of each node.

It is noted that the selection of the terminals and functions is critical for GP to success. The terminal set and the function set should be selected so as to satisfy the requirements of *sufficiency* and *closure*. The sufficiency means that there must be some combinations of terminals and functions that can solve the problem while the closure means that any function can accept any input value returned by any function and terminal. In this example, the return values of all functions and terminals are numerical. However, it is useful to have programs with different return types. Strongly typed GP and grammar guided approaches such as tree-based grammar guided GP and linearised grammar guided GP [76], can be

used to handle this situation [115]. For strongly typed GP [82], types and constraints are incorporated into the GP system. Thus, each terminal possesses a type and each function will only accept the arguments of certain types and it also has its own return type. In addition, grammar-based approaches are also used to ensure that the initial population is made up of legal "grammatical" programs by using grammar to express constraints [37, 42, 102]. Moreover, the corresponding genetic operators also need to follow the grammar. A comprehensive discussion about the advantages and disadvantages of these solutions are presented in [115].

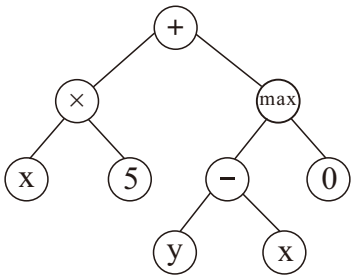


Figure 2.1: An example of tree-based GP program.

2.2.2 Initialisation

GP, as a population-based approach, the first step of the evolutionary process is to generate a population randomly. For GP, two popular methods, which are *full* and *grow* [64] methods, were proposed in the early stage. A maximum depth is determined for each GP individual to restrict the size of one program. For the full method, the terminals in generated trees are all located at the maximum depth of the trees. In grow method, nodes are randomly selected from the terminal and function set until maximum depth of the tree is reached. In order to improve the diversity of initial population, these two methods are combined, which is known as *ramped half-and-half* to generate the population. That is, this hybrid method is to generate half population by full method and the other half population by grow method. An example of generated individual by full method and grow method are shown in Figure 2.2 (a) and 2.2 (b).

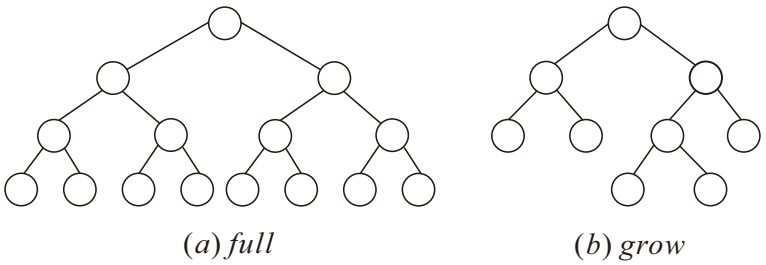


Figure 2.2: An example of programs generated by *full* method and *grow* method.

2.2.3 Evaluation

Evaluation is an important step in GP to measure the fitnesses of the programs in the population according to the fitness function. The fitness function plays a significant rule in GP to guide the search to find good programs (solutions). Since an evolved program is normally used to solve different problems, its fitness should the performance of the program when it is applied in different problems. For example, in the classification problem, its fitness can

be defined as the rate of succeeded predicted instances while the fitness can be the total flowtime in JSS problem. Although in the same problem, the fitness also can be defined in different forms. For example, the fitness in classification problem also can be defined as the aggregate errors between the outputs and the target outputs of the evolved programs. In general, the fitness function is defined based on the objectives of the problems.

2.2.4 Selection

The fitness of an individual obtained in evolutionary process decides its chance to be selected to generate new individuals. That is, good individuals are more likely to be selected as parents to generate new offsprings. There are three popular selection methods in evolutionary computation, which are roulette wheel selection, fitness proportionate and tournament selection [63]. For roulette selection, individuals are randomly selected based on the distribution determined by their fitness. Individuals with good fitnesses have larger chances to be selected while individuals with bad fitnesses have smaller probabilities to be chosen. Proportionate selection is performed based on the rank of individuals in the population. For tournament selection is performed with two steps. In the first step, a number of (tournament size) individuals will be sampled in the population with equal probability, thus bad individuals also have larger relatively chances to be chosen. In the second step, the individuals will be selected depends on their fitnesses. Thus, individuals with bad fitnesses also have more chances to be chosen compared with the other two methods.

2.2.5 Evolution

Evolution is the main process of GP. Three are three basic operators of GP, which are *crossover*, *mutation* and *reproduction*. These operators aim at generating new population by inheriting good materials from old population into new population. The probabilities of applying these three operator can be predefined by users.

For crossover, two parents are selected as parents by using one of the selection methods. The most commonly used form of crossover in GP is *subtree crossover* [64]. Firstly, a subtree (crossover point) will be selected randomly in each parent. Secondly, these two subtrees from the parents will be swapped. Thus, two new individuals (offsprings) are created and then copied into new population. It is suggested by Koza [64] that the probabilities of choosing functions and terminals are 90% and 10%, respectively. The reason is that a large number of nodes in GP trees are terminals, uniformly selecting nodes for crossover may lead to very small exchanges of the materials.

Different from crossover, one parent is needed for mutation. The most popular type of mutation in GP is the *subtree mutation*. Firstly, one individual in the population will be chosen according to one of the selection method. Secondly, a subtree (mutation point) will be selected randomly. Finally, the chosen subtree will be replaced by a newly generated subtree.

Reproduction in GP is quite straightforward. A individual will be firstly chosen by selection method in the population then copied into the new population directly. Besides, the elitisms mechanism, which just simply picks up the top several individuals of the current generation and then inserts them into the population of the next generation, is used to ensure that the best individuals will not be lost when generating new populations.

2.3 Existing Approaches for Job Shop Scheduling

Over the years, a lot of approaches have been adopted for solving JSS problems. Different types of these methods that link to this filed are briefly discussed as follows.

Exact Methods

Many techniques to search for optimal solutions, which are known as exact approaches such as dynamic programming [6] and branch-and-bound [66], have been investigated in the literatures [17, 18, 19]. However, they mainly relate to small scale problems. On one hand, exact methods aim at finding optimal solutions. On the other hand, these techniques are too time-consuming when the problems are getting large. Normally, exact methods are only limited to solve small scale static job shop scheduling problems. It is hard to handle dynamic problems where a lot of real time decisions are needed to be made quickly.

Heuristic Methods

A heuristic method is a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution [109]. The objective of a heuristic method is to produce a solution in a reasonable time frame that is good enough for solving the problem at hand.

Heuristic search methods such as tabu search [98], simulated annealing [123], bee colony algorithm [23] and genetic algorithms [21, 110] have been proposed to find "good enough" solutions for solving JSS problems [26, 36, 46] with a reasonable time. There are also some heuristic search methods, which are combined with exact methods, such as branch and bound algorithm to improve the ability for solving JSS problem [2]. Jia et al. [53] proposed a modified genetic algorithm to solve the traditional scheduling problems as well as distributed shingling problems. Pezzella et al. [110] presented a genetic algorithm, which integrates different strategies for generating the initial population, selecting the individual for reproduction and reproducing new individuals, for the flexible JSS problem. Chen et al. [20] separated the chromosomes into two parts. The first part was used for routing policy and the second part was used for the sequence of the operations on each machine. Genetic operators were introduced and used in the reproduction process of the algorithm. Numerical experiments showed that the proposed algorithm can find out high-quality schedules. However, they are often not suitable for solving DJSS problems due to their lack of ability to reflect in time.

It is noted that priority rules are probably the most frequently applied heuristics for solving JSS problem in real-world applications due to their ease of implementation and their low time complexity. To be specific, dispatching rules, as priority functions, have been widely adopted for solving DJSS problems [8, 38] due to the ability to react in real time. However, dispatching rules (heuristic) are normally manually designed [22, 28, 51, 52, 124]. The design of dispatching rules requires human experts, which is typically expensive and time-consuming. In addition, manually designed dispatching rules are relatively simple and they are normally restricted to some specific assumptions [9, 35, 57] and have difficulties in handling complex practical scenarios [31, 40, 111]. A comprehensive comparison among a large number of dispatching rules can be found in [119].

Hyper-heuristic

A hyper-heuristic [15] is an automated methodology for selecting or generating heuristics to solve hard computational search problems. It can be used to automatically evolve dis-

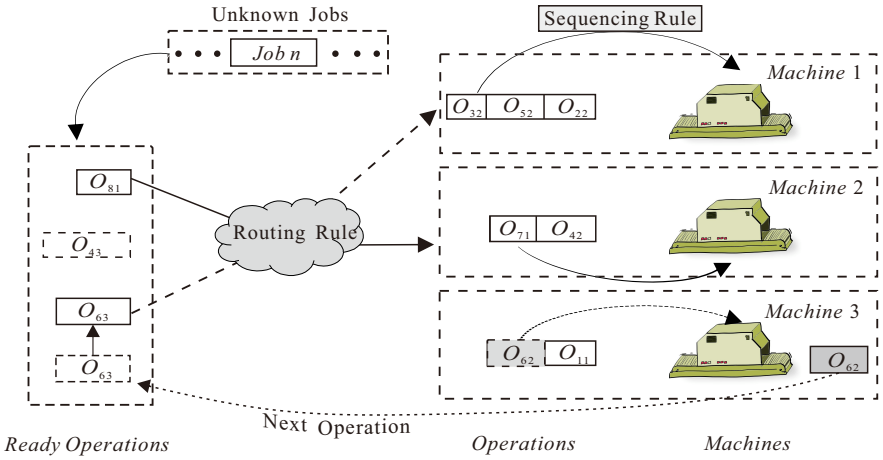


Figure 2.3: An example of decision process of DFJSS.

patching rules for JSS. There are two main categories of hyper-heuristics, which are *heuristic selection* and *heuristic generation* [12]. Heuristic selection methodologies aim at choosing or selecting existing heuristics while heuristic generation methodologies aim at generating new heuristics from components of existing heuristics. The area of automated heuristic or *hyper-heuristics* [12, 16, 117] have been proven to be promising for designing heuristics. The main difference of heuristic methods and hyper-heuristic methods is that hyper-heuristic methods are to explore the "heuristic search space" rather than the solution space. A most state-of-the-art survey of existing studies on hyper-heuristics and its applications can be found in [10].

Genetic Programming Hyper-heuristic A general GP-based hyper-heuristic (GPHH) framework was presented in [15]. It has been successfully used in some applications such as 2-D strip packing [13], bin packing [14], timetabling [3, 113, 112] and job shop scheduling [44, 87, 88, 91].

In the last decade, GP has been the dominating technique to automatically evolve dispatching rules for JSS compared with hyper-heuristics based on supervised learning such as particle swarm optimisation [62], decision tree [100, 120], logistic regression [47], support vector machine [121], and artificial neural networks [30, 125]. Compared with different supervised learning methods mentioned above, the heuristics obtained by GP can be interpreted to some extent, which is a very important characteristic to improve the applicability in real-world.

For DFJSS, naturally, two kinds of dispatching rules are needed, which are routing rules and sequencing rules, respectively. In this case, the quality of DFJSS schedule depends highly on how well the routing rule and the sequencing rule work together. Since the term *dispatching rule* has been used in different contexts, it is worth highlighting that the concept of a dispatching rule in DFJSS consists of a routing rule and a sequencing rule (two kinds of rules). It is noted that the dispatching rule used in DJSS takes the same role as the sequencing rule in DFJSS. To avoid confusion, we use the term *dispatching rule* to contain two kinds of rules (routing rule and sequencing rule).

Figure 2.3 shows an example of decision process of DFJSS. In the figure, the solid lines stand for what is happening and the dotted lines indicate what will happen. There are three machines in the job shop and each job can be processed by any machine. Each job consists of several operations in a certain order. In the current system state, the operations (O_{32}, O_{52}, O_{22}, O_{71}, O_{42}, O_{62} and O_{11}) have been allocated to different machines by the routing rule.

Then, each machine uses the sequencing rule to decide the next operation to be processed, e.g. *machine* 3 selects O_{62}. After O_{62} is completed, its subsequent operation O_{63} becomes ready, and will be allocated by the routing rule.

2.4 Related Work

As mentioned above, to our best of knowledge, dispatching rules are more effective and efficient for solving JSS problems, especially for DJSS. Although there is a huge amount of literature on JSS, DFJSS does not have a rich literature. In this section, the works, which aim at solving JSS by dispatching rules, will be presented with a focus on using GPHH approach. This is because GPHH is a dominating approach to automatically evolve dispatching rules for DJSS, which is the focus of this thesis. We will also briefly summarise other techniques for evolving rules to solve the JSS problems.

2.4.1 GPHH for Evolving Dispatching Rules

In this section, the previous literatures related to JSS are classified into two categorisations according to the machine environment (i.e., the number of machines in the job shop). The reason is that the machine environment (e.g., single machine environment and multi-machine environment) is a key factor in determining the type of the JSS problems. In the single machine environment, there is only one machine in the job shop. Meanwhile, in the multi-machine environment, more than one machine are available. The task here is not to enumerate all the literatures, but to give a big picture of the research about DFJSS.

Single Machine Environment

All the works mentioned here are performed in single machine environment. In this environment, the job shop scheduling problems can be static or dynamic, but cannot be flexible because flexible problems are generated in the multi-machine environment.

In 2001, Dimopoulos and Zalzala [27] firstly investigated GP framework to evolve dispatching rules for solving static scheduling problems with one machine. The results showed that the evolved dispatching rules are better than traditional rules in different scenarios with different levels of tardiness and tightness of due dates. Geiger et al. [33] examined a GP learning system for scheduling in a single machine environment. In this work, both the static and dynamic problem were considered and the results showed GP can handle well this two situations. At the end of this paper, a two-machine flowshop environment were involved. However, this work tried to evolve dispatching rule for each machine, which will increase the complexity of the job shop system. Nie et al. [97] presented a gene expression programming based scheduling rules constructor to construct effective scheduling rules for dynamic single-machine scheduling problems with one or more objectives. Yin et al. [127] proposed bi-tree structured representation scheme for GP to make it possible to search sequencing and idle time in different uncertain environments. In this paper, the use of GP to evolve single-machine predictive scheduling heuristics with stochastic breakdowns was investigated, where both tardiness and stability objectives in face of machine failures were considered. Jakobović et al. [49] proposed a multiple tree adaptive heuristic for dynamic single machine scheduling problem, where decision tree was used to distinguish between resources based on their load characteristics. It works as a GP-3 system that evolves three components, a discriminant function and two dispatching rules. Among them, the discriminant serves as a monitor. Which rule will be chosen to use depends on the decision made by discriminant function, which was designed to identify the bottleneck machine.

Most of the mentioned works relate to single objective and there is little work about multi-objectives. Even multi-objectives are considered, the works are converted to single objective problems by weighted sum method, which are single objective problems in essence.

Multi-machine Environment

In the multi-machine environment, JSS can be investigated in either static or dynamic conditions. It can be non-flexible or flexible. The type of JSS can also be any combination of the four factors (i.e., static, dynamic, non-flexible and flexible). In this section, we will give a high-level survey of these works with focusing on DJSS, FJSS and DFSS.

Dynamic Job Shop Scheduling. Jakobović et al. [50] applied GP to build scheduling algorithms for multiple machine environment and also considered the dynamic variants (job arrival) of the problem. The work showed the effectiveness of GP than existing scheduling methods. Mei et al. [77] presented a feature selection, in which a niching-based search framework was used for extracting a diverse set of good rules, for job shop scheduling. In addition, a surrogate model was applied to reduce the training time. The experimental studies showed that it took less than about 10% of the training time of the standard GP training process, and can obtain much better feature subsets than the entire feature set. Nguyen et al. [92] adopted the surrogate model in [39] for fitness approximation and proposed a select scheme to investigate the influence of surrogate models in dynamic job shop scheduling. This paper also analysed the advantages and disadvantages of different selection schemes in surrogate-assisted GP. Mei et al. [79] defined the concept of time-invariance and developed a new terminal selection scheme to guarantee the time-invariance throughout the GP process in a dynamic environment. Mei et al. [81] aimed at selecting a small set of useful features according to the contribution of the features and investigated the feature selection mechanism in static and dynamic environment. The results showed that using only the selected features can lead to significantly better GP-evolved rules on both training and unseen test instances. Nguyen et al. [87] proposed a diversified multi-objective cooperative coevolution method based on GP to evolve dispatching rules and due-date assignment rules in dynamic job shop scheduling. The results showed that the proposed method can effectively evolve Pareto fronts of scheduling policy compared to NSGA-II and SPEA2 while the uniformity of scheduling policy is better than those evolved by NSGA-II and SPEA2. Nguyen et al. [88] investigated different representations of the dispatching rules based on the previous literatures. The results showed that the representation that integrates system and machine attributes can improve the quality of the evolved rules.

All of these works are investigated in multi-machine environment and the dynamic events are considered, however, they are still non-flexible JSS.

Flexible Job Shop Scheduling. Ho et al. [41] applied GP to evolve composite dispatching rules for solving FJSS problem and the results showed that the obtained rules outperform the selected benchmark rules in 74% to 85% of problem instances. Tay et al. [122] tried to evolve dispatching rules for solving multi-objective flexible job shop scheduling problems using genetic programming. However, the multi-objective problem was converted into the single objective problem by linearly combining all objective functions. Hildebrandt et al. [40] re-examined this work in different dynamic job shop scenarios and showed that the rules evolved in [122] are only slightly better than the earliest release date rule and quite far away from the performance of the SPT rule. They explained that the poor performance of these rules was caused by the use of linear combination of different objectives and the fact that the randomly generated instances cannot effectively represent the situations that

happened in a long term simulation. Thus, Hildebrandt et al. [40] aimed at only minimizing mean flow time by evolving dispatching rules which were trained them on four simulation scenarios. The experimental results indicated that the evolved rules were quite complicated but effective when compared to other existing rules.

The investigations among these works belong to FJSS, which are more practical. However, they are conducted in static environments which are not the normal cases in real-world. Moreover, the routing rules in these work are fixed and actually only sequencing rules are evolved.

Dynamic Flexible Job Shop Scheduling. Yska et al. [129] proposed a new GPHH algorithm with cooperative coevolution to explore the possibility of evolving both routing and sequencing rules together. The results showed that co-evolving the two rules together can lead to much more promising results than evolving the sequencing rule only. This is the first work that considered to evolve routing rule and sequence rule at the same time using GP. Feature construction with different strategies was firstly developed in [128] to improve the efficiency of GP. The experiment results showed that although the proposed feature construction methods did not manage to improve the results, they improved the stability of the evolutionary process.

It is noted that these works are consider to evolve the routing and sequencing rules simultaneously, which have real merits. However, the research in this field is still in a very early stage and little work has been reported on this significant aspect.

Others. There are a few works which are investigated in the multi-machine environment but not included in DJSS, FJSS and DFJSS. Because they are related to this thesis, we will describe them here.

Many objectives [25] optimisation is a more realistic task in JSS and they are investigated in static environment so far. Masood et al. [74] firstly proposed a hybridized algorithm that combines genetic programming and NSGA-III [25] to evolve a set of trade-off dispatching rules for many-objective JSS. In [73], a reference point adaption method was proposed based on the distribution of the candidate solutions. The results showed that the proposed method outperformed the existing state-of-the-art algorithms for many-objective JSS. A new reference point adaptation mechanism inspired by particle swarm optimisation was also proposed in [75]. The results showed that the proposed mechanism can significantly improve the importance of GPHH and NSGA-III in terms of both HV [132] and IGD [130].

These works are more applicable and they are start points of involving many-objective in JSS. However, they are conducted in static and non-flexible JSS.

Summary. Table 2.2 shows the characteristics of job shop scheduling problems in the previous research. "$\surd$" means the reference has the corresponding characteristic. It is noted that we treat a work as single-objective if it handles multi-objective by converting the multi-objective to a single-objective (e.g., a weighted sum approach). These literatures in Table 2.2 not only include the mentioned works in this section but also other related works.

According to the statistical information, most GPHH works belong to dynamic non-flexible job shop scheduling problems with single objective. Although there are some works relate to flexible job shop scheduling, they are investigated in static environments. One of the main shortcomings is that in these works the routing rule is fixed and only sequencing rule is evolved. The research about dynamic flexible job shop scheduling, which is much more closer to real-world job shop scheduling, is still at the very early stage. It is noted that

Table 2.2: The characteristics of job shop scheduling problems involved in the previous researches.

Ref.	Job Attributes		Machine Resource		Objective Type		
	Static	Dynamic	Non-flexible	Flexible	Single	Multiple	Many
[27]	√		√		√		
[29]	√		√		√		
[32]	√		√		√		
[49]	√		√		√		
[88]	√		√		√		
[94]	√		√		√		
[85]	√		√		√		
[89]	√		√		√		
[81]	√		√		√		
[80]	√		√		√		
[33]	√	√	√		√		
[44]		√	√		√		
[45]		√	√		√		
[97]		√	√		√		
[127]		√	√		√		
[49]		√	√		√		
[111]		√	√		√		
[86]		√	√		√		
[108]		√	√		√		
[50]	√	√	√		√		
[77]		√	√		√		
[92]		√	√		√		
[79]		√	√		√		
[78]		√	√		√		
[107]		√	√		√		
[60]		√	√		√		
[59]		√	√		√		
[106]		√	√		√		
[58]		√	√		√		
[104]		√	√		√		
[105]		√	√		√		
[116]		√	√		√		
[81]	√	√	√		√		
[90]	√		√			√	
[87]		√	√			√	
[91]		√	√			√	
[93]		√	√			√	
[61]		√	√			√	
[88]	√		√		√		
[41]	√			√	√		
[122]	√			√	√		
[40]	√			√	√		
[129]		√		√	√		
[95]		√		√	√		
[128]		√		√	√		
[74]	√		√				√
[73]	√		√				√
[75]	√		√				√

there are some works related to many-objective, however, they are investigated in simple environment (i.e., static and non-flexible).

2.4.2 Other Techniques for Evolving Dispatching Rules

Except for GPHH, other techniques have also been applied to evolve dispatching rules. Li et al. [70] tried to use decision tree models to discover previously unknown dispatching rules by applying them directly to production data. However, an implicit assumption is that it is worthwhile to capture the current practices from historical data. Furthermore, this approach does not seek to directly improve any scheduling performance measure. Geiger et al. [32] used a genetic algorithm to discover new dispatching rules in a two-machine flow shop environment. However, no statistical test was performed to support the results. Ingimundardottir et al. [47] introduced a framework to discover dispatching rules by analysing the characteristics of optimal solutions. The learned linear priority dispatching rules outperformed common single priority dispatching rules, with respect to minimum makespan. Obviously, this work has an implicit assumptions that the optimal solutions are available. This is not true in many cases. In general, these techniques are not effective compared with GPHH.

2.5 Summary

This chapter firstly reviewed the main concepts of job shop scheduling, especially dynamic flexible job shop scheduling, and the genetic programming approach. Secondly, existing approaches for JSS are illustrated. Then, the related work about GPHH and other techniques for evolving dispatching rules has been reviewed. It is obvious that genetic programming has become a popular approach to automatically evolving dispatching rules for JSS. However, the most works in this area relate to non-flexible JSS. There are some works on FJSS, but most of them are investigated in static environment with single-objective. Furthermore, only single objective is considered in most works. In addition, the main drawback is that the routing rule is fixed to evolve sequencing rule in most of the FJSS related works. Only one paper tried to evolve routing and sequencing rule simultaneously. Besides, multi-objective researches are very limited in DFJSS.

This thesis will first investigate appropriate representation to evolve routing and sequencing rules at the same time. And then, this thesis will use feature manipulation approaches and surrogates techniques to improve the effectiveness and efficiency of GP. Finally, multi-objective techniques will be investigated to achieve a trade-off among conflicting objectives.

Chapter 3

Preliminary Work

This chapter presents the initial works conducted in investigating GP for DFJSS. In the first work, GP with multi-tree presentation is introduced to handle routing and sequencing simultaneously for solving DFJSS problems. This work aims at investigating appropriate representation of GP to cope with DFJSS problems and the proposed approaches are compared with CCGP [129]. In the second work, surrogate-assisted GP is investigated to reduce the computation time. The rest of this chapter is organised as follows. Section 1 gives the description of proposed GP with multi-tree representation. Then, the proposed surrogate-assisted GP is introduced in section 2. Experiment is designed in section 3 and results with analyses of the proposed approaches are shown in section 4.

3.1 Genetic Programming with Multi-tree Representation

The choice of which representation to use when dealing with a problem using GP is vital. The representation is the crux of the applicable algorithm. There are two main reasons. Firstly, an appropriate representation is definitely a rudimentary factor for an algorithm to build a solution. Secondly, the representation determines the size of the search space and there is a clear trade-off between the complexity of the representation and the ability of GP to explore the search space. These two facts foster the motivation to propose a more suitable representation for DFJSS.

Tree-based GP is a popular way in previous research and multi-tree representation [65] as a special structure has been applied to classifier design [24, 83] and feature manipulation [67]. For DFJSS, a key issue for the tree-based representation of GP is how it can capture both the routing and sequencing rules at the same time. In this work, to address this issue, we proposed to use multi-tree GP (MTGP) to evolve both routing and sequencing rules together.

In multi-tree representation, each individual is represented as a list trees. Taking advantage of this feature to solve DFJSS problems, routing and sequencing rules can be denoted by different trees in one individual. According to this, multi-tree representation naturally lends itself to DFJSS. The pseudo-code of MTGP is given in Algorithm 1.

In this work, we use the multi-tree representation that one individual contains two trees to match our problem. To be specific, the first tree is used to indicate the sequencing rule and the second tree denotes the routing rule. The fitness of one individual depends on the two trees working together.

In the case of multi-tree representation, the evolutionary algorithm must come to a decision as to which trees the genetic operator will be applied.

Algorithm 1: Pseudo-code of MTGP

```
// Initialization
while N_ind < Popsize do
    foreach individual
        Initialize each tree //Randomly initialize each tree by ramp half-and-half
end
// Evolution
while Stopping criteria not met do
    Evaluate the individuals
    Copy the elites to the new population
    Select individuals based on fitness value
    Generate offsprings by applying crossover/mutation/reproduction operators
end
return best individual
```

3.1.1 The Basic Crossover Operator

In this work, only crossover operator is considered because it plays a key role in the evolutionary process with high probability. In classical multi-tree representation, the crossover operator is defined to act upon only one tree in an individual at a time. Other trees are unchanged and copied directly from the parents to the offsprings. Crossover operator is limited to a single type of trees at a time in the expectation that this will reduce the extent to which they disrupt "building blocks" of useful code.

3.1.2 The Improved Crossover Operator

However, when coping with DFJSS problem, such a crossover operator, as we mentioned above, has the following issues.

Firstly, the crossover operation only happens between one type of trees of the parents, therefore, the offsprings generated might not be substantially different from their parents. Thus, the population will lose its diversity and the ability of exploration will decrease.

Secondly, the crossover operation cannot improve the diversity of the combinations of routing and sequencing rules. In DFJSS, a good rule cannot be "good" by itself, but should behave well when collaborating with the other rule. Thus, the diversity of combinations is an important factor for achieving good solutions.

In order to overcome these shortcomings and make the algorithm more in line with the properties of DFJSS, a new tree swapping crossover operator is proposed and the corresponding approach is named as sMTGP. Figure 3.1 shows the tree swapping crossover operator, which shares the same process with the classical crossover operator except that the unselected trees (the same type) are also swapped with each other. Figure 3.1 shows two parents ($parent_1$ and $parent_2$) are selected to generate offsprings and the second type (T_2) of trees is selected for crossover. The dotted circles mean that the subtrees are chosen and will be swapped. The standard crossover operator will stop here. But for the tree swapping crossover operator, the other type of trees is also swapped. Thus, two offsprings ($Offspring_1$ and $Offspring_2$) are generated.

This will bring two benefits. The first is that useful blocks are not easily broken. The second is the pairs or combinations of routing and sequencing rules examined in sMTGP are much more diverse. That is to say, the population of sMTGP will become more diverse compared with MTGP. More importantly, this point matches well with the characteristics of the DFJSS.

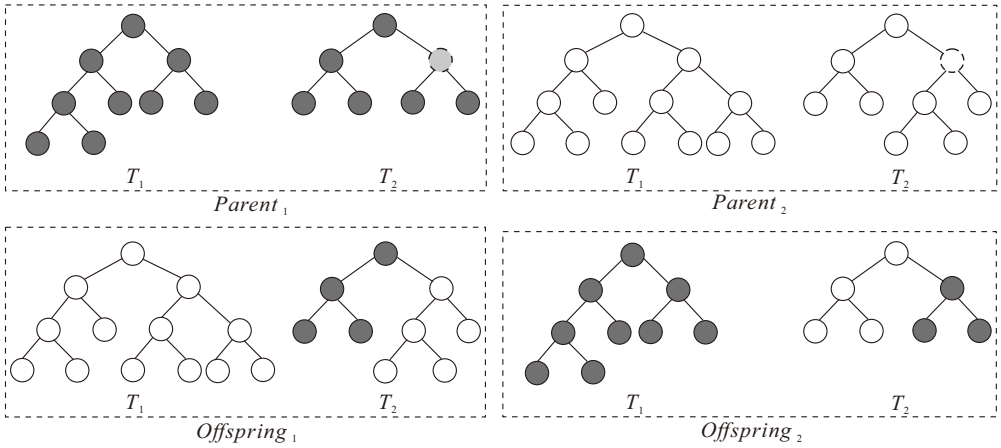


Figure 3.1: Tree swapping crossover operator for multi-tree representation.

3.1.3 Experiment Design

In these two works, we present the results of our experiments obtained by the involved algorithms using three commonly used objectives, namely: (1) max-flowtime, (2) mean-flowtime, and (3) mean-weighted-flowtime.

In order to test the effectiveness and robustness of proposed algorithms, six simulation scenarios based on three objectives and two utilization levels ($3 * 2$) are involved.

Because of the lack of the best known objective value of the instances, benchmark routing and sequencing rules, which have been proven that can get better performance in specific objectives than the counterparts [38], are chosen. The details are shown in Table 3.1.

Table 3.1: The benchmark rules used for different objectives.

Rule type	Objectives	Rule	Description
Routing rule	max-flowtime mean-flowtime mean-weighted-flowtime	LWIQ	Least Work in Queue
Sequencing rule	max-flowtime	FCFS	First Come First Serve
	mean-flowtime	SPT	Shortest Processing Time
	mean-weighted-flowtime	FCFS	First Come First Serve

In the training (test) process, the training (test) performance is the quality of evolved rules. The relative performance ratio, which was defined as the average normalized objective value (f) obtained by evolved rules over the value obtained by benchmark rules, is used to indicate the quality of rules. The equation of normalized objective value is shown as Eq. (1). For the results, the smaller the value, the better.

$$f = \frac{Obj(S(p_r,p_s,I_{train/test}))}{Obj^*(I_{train/test})} * \frac{1}{|I_{train/test}|} \quad (1)$$

In Eq. (1), the evolved routing rule and sequencing rule are denoted by p_r and p_s. The training or test set is represented by $I_{train/test}$. The number of simulations used for each objective in training or test process is represented by $|I_{train/test}|$. $S(p_r, p_s, I_{train})$ is the obtained schedule. The normalized objective value obtained by evolved rules and benchmark rules are denoted by $Obj(S(p_r, p_s, I_{train/test})$ and $Obj^*(I_{train/test})$.

The best pair of rules of the last generation was tested on test data set to measure its performance. The test data set consists of 50 dynamic simulations with different random

seeds. In addition, Wilcoxon signed rank test at the 5% level was used for comparison between the involved algorithms.

Parameter Settings

In our experiment, the terminals in [79] were adopted. The details are shown in Table 3.2. The function is $\{+, -, *, /, max, min\}$, following the setting in [79]. The arithmetic operators take two arguments. The "/" operator is protected division, returning the largest double positive number if dividing by zero. The *max* and *min* functions take two arguments and return the maximum and minimum of their arguments, respectively.

Table 3.2: The terminal set.

	Notation	Description
machine-related	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 on a specified machine
job-related	NPT	Median processing time for the next operation
	OWT	The waiting time of an operation
	WKR	Median amount of work remaining for a job
	NOR	The number of operations remaining for a job
	W	Weight of a job
system-related	TIS	Time in system

The initial population is generated using the ramped-half-and-half method with minimum depth of two and maximum depth of six, and with function and terminal sets as described above. The population size is 1024 and evolution is for 51 generations. The GP trees have a maximum depth of eight. For the genetic operators, the crossover, mutation and reproduction rates are 0.80, 0.15 and 0.05, respectively. The rates of terminal and non-terminal selection are 0.10 and 0.90. Tournament selection with a tournament size of seven is used to select individuals for genetic operators, which is a common setting in previous work [40, 88]. In our experiment, 30 runs are executed, which assures that the results represent the average behaviour and not extreme situations.

Simulation Configuration

For dynamic simulation, commonly used configuration is adopted [39, 79]. In the job shop, there are ten machines, which has been proven to be a good showcase for job shop environment. Job arrival follows poison distribution. All the jobs are given weight 1, 2 or 4, with probability (0.2,0.6,0.2) [114], according to the premise of the 20/60/20 rule which is 20% of jobs are of low importance, 60% of jobs are of average importance, and 20% of jobs are high importance. A warm up period of 1000 jobs is used to reach the steady state of the job shop, and we collect data from the next 5000 arriving jobs. It is noted that new jobs keep arriving in the system until the 6000th job is processed.

The number of operations and the candidate machines of each job varies randomly between 1 and 10 with equal probability. In addition, processing time of each operation will follow uniform discrete distribution between 1 and 99. Two utilization levels are 0.85 and 0.95. It is noted that, in order to improve the generalisation ability of the evolved rules for DFJSS problems, the seeds used to stochastically generate the jobs are rotated in the training process in each generation.

3.1.4 Results and Analyses

As mentioned earlier, the proposed algorithms are compared with CCGP [129] to verify its effectiveness and efficiency. The goal of this work is to evolve routing and sequencing rule at the same time effectively and efficiently in solving DFJSS problems.

Optimisation Performance

In this work, first of all, MTGP and sMTGP are compared with CCGP respectively to measure the feasibility of multi-tree based GP. Then, sMTGP and MTGP are compared to analyse the effectiveness of proposed tree swapping crossover operator.

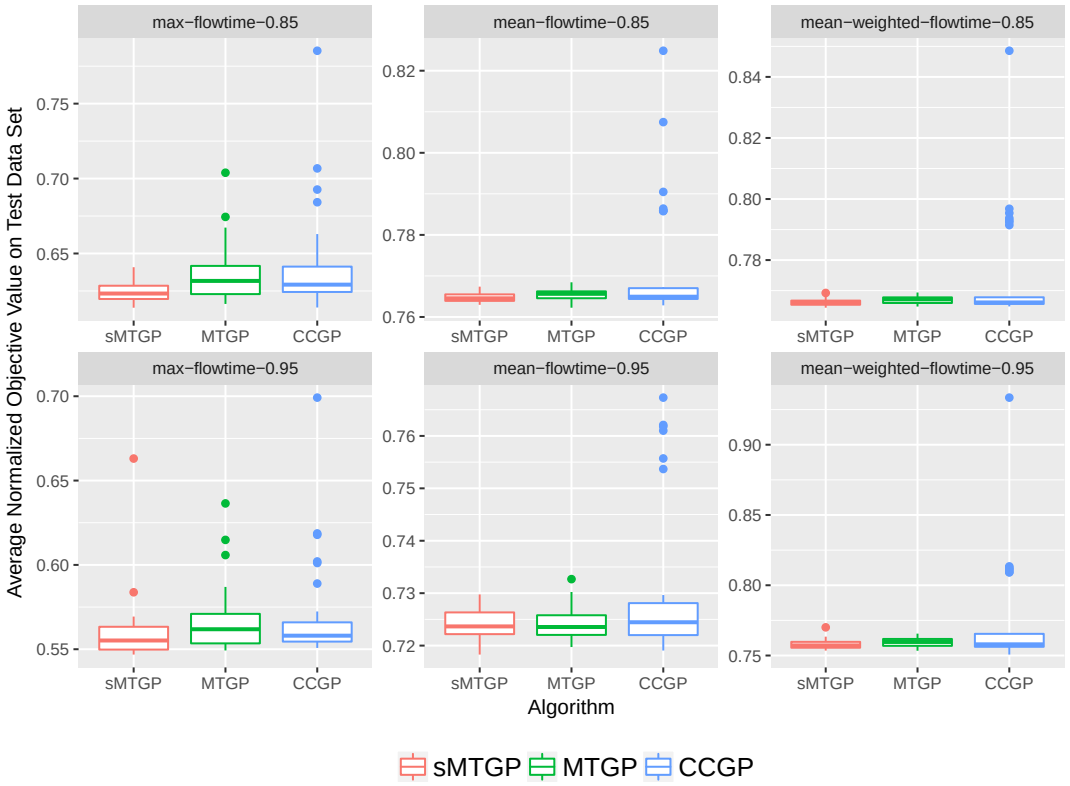


Figure 3.2: The boxplot of average normalized objective value obtained by sMTGP, MTGP and CCGP on test data set.

All the mean value obtained by MTGP and sMTGP are better than CCGP and all the standard deviation value are smaller than the counterparts. Wilcoxon signed rank test results show that sMTGP is significantly better than CCGP only in two scenarios (Max-Flowtime-0.85, Mean-Flowtime-0.85). It is interesting that MTGP got better mean value than CCGP, but none of the instances of MTGP is significantly better than CCGP.

When further looking into the boxplot in Figure 3.2, one can see that CCGP has many more outliers than MTGP and sMTGP. This is because CCGP cannot handle well the interactions between routing and sequencing rules directly, thus can be stuck into poor local optima more often. The reason why there is no statistical significance between MTGP and CCGP is that the two algorithms showed very similar performance except the outliers. Figure 3.2 clearly shows that multi-tree representation managed to dramatically reduce the probability of outliers.

According to these observations, the performance of GP with the multi-tree representation is more stable than GP with cooperative co-evolution. Also, Wilcoxon signed rank test

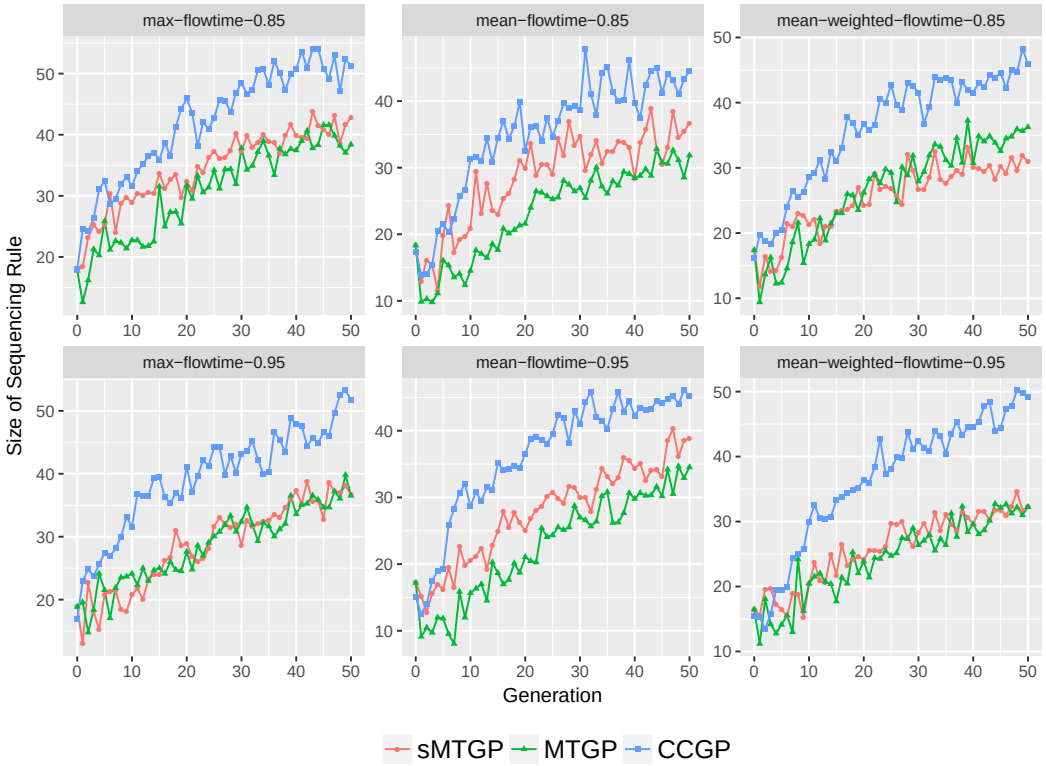


Figure 3.3: The convergence curves of the average best sequencing rule size (30 runs) obtained by sMTGP, MTGP and CCGP at each generation.

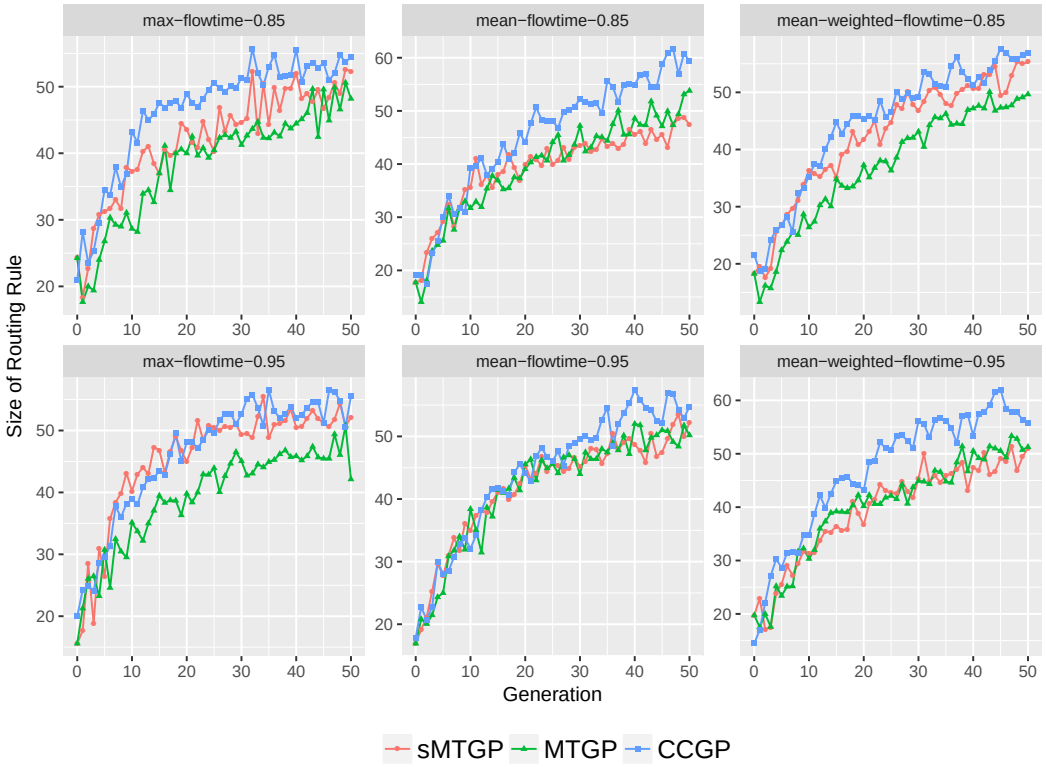


Figure 3.4: The convergence curves of the average best routing rule size (30 runs) obtained by sMTGP, MTGP and CCGP at each generation.

Table 3.3: The average training time over 30 independent runs of the three algorithms.

Index	Scenario	Training Time (seconds)		
		sMTGP	MTGP	CCGP
1	Max-Flowtime-0.85	4459.9	4267.1	4642.8
2	Max-Flowtime-0.95	5057.2	4790.3	5144.9
3	Mean-Flowtime-0.85	4184.5	4278.0	4538.5
4	Mean-Flowtime-0.95	4667.6	4721.3	4849.9
5	Mean-weighted-Flowtime-0.85	4348.1	4181.7	4458.4
6	Mean-weighted-Flowtime-0.95	4585.7	4680.3	4957.0

results show that sMTGP is significantly better than MTGP in four scenarios, which are Max-Flowtime-0.85, Mean-Flowtime-0.85, Mean-weighted-Flowtime-0.85/0.95. It means that the proposed tree swapping crossover operator can effectively improve the performance of MTGP.

Figure 3.3 shows that the sizes of evolved best sequencing rules by sMTGP and MTGP are obviously and dramatically smaller than the best rules evolved by CCGP. Also, Figure 3.4 shows that the best routing rule sizes got by sMTGP and MTGP are smaller than that of CCGP. However, there is not so much difference compared with the changes of sequencing rule sizes. These observations confirm the potential of using multi-tree based GP to achieve smaller size rules.

From Table 3.3, it is clear that sMTGP and MTGP can evolve rules with lower time complexity than CCGP in all scenarios. In addition, for sMTGP, less training time is needed as compared to MTGP in three situations (scenario 3, 4, 6). This is a promising finding that the multi-tree representation is computationally cheaper than cooperative co-evolution.

Overall, MTGP and sMTGP (especially) undoubtedly show better ability to solve DFJSS problems. They can obtain better and smaller rules within a shorter training time.

Further Analysis

In the last section, the rule size relates to the best rule only. In order to explore whether the best rule is smaller by chance or the rules in the whole population generally become smaller, in this section, the average rule sizes in the whole population at each generation were investigated to get a clear vision of the changes of rule sizes.

We took the last scenario (Mean-Weighted-Flowtime-0.95) as an example to further investigate the changes of rule sizes. The details are shown in Figure 3.5 and Figure 3.6.

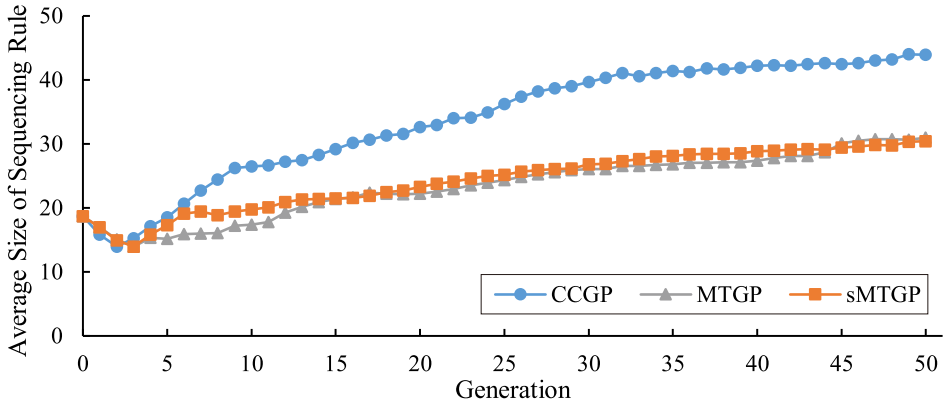


Figure 3.5: The convergence curves of average sequencing rule size (30 runs) obtained by CCGP, MTGP and sMTGP in population at each generation.

As shown in Figure 3.5 and Figure 3.6, at the initial point, for all the three algorithms, the average sizes of both rules are about equal. However, the average sizes obtained by CCGP are larger than others over time. Maybe in multi-tree based GP, effective and smaller rules are more likely to be well preserved because there is at least one rule structure will not be changed by operator at each time during the evolution process. In addition, the average sizes obtained by MTGP and sMTGP show the same trend basically and routing rule sizes are bigger than sequencing rules. This is consistent with the observation in the last section.

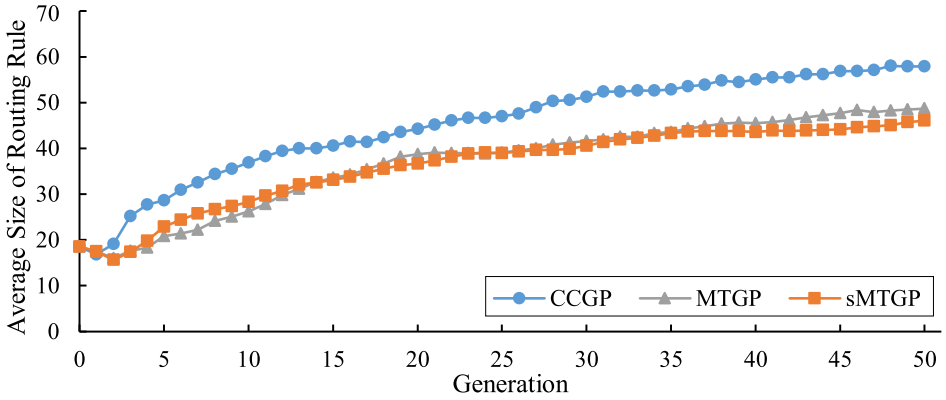


Figure 3.6: The convergence curves of average routing rule size (30 runs) obtained by CCGP, MTGP and sMTGP in population at each generation.

3.2 Surrogate-Assisted Genetic Programming

GP, as a powerful approach, has been widely used for automatically evolving priority rules for solving job shop scheduling problems. However, one of the main drawbacks of GP is the intensive computational requirements. This work aims at investigating appropriate surrogates for GP to reduce its computational cost without sacrificing its performance in solving DFJSS problems.

For surrogates, a common assumption is that higher fidelity models are generally more accurate at the expense of higher computation time. In our research, the fidelity of models depends on the complexity of simulation models and the number of jobs is the key factor in determining their complexities. The simulation models with more jobs will get higher fidelities, but also suffer expensive computation time.

It is worth noticing that approximation error in the surrogates does not always harm. Surrogates may contribute more to the evolutionary search than the original models because of the capable of smoothing the multimodal or noisy landscape of the complex problem.

Naturally, it is better to use simple surrogates in the early stage of the optimisation and increase the quality of the surrogate as the search proceeds. The use of dynamic surrogates is able to improve the search speed.

Our preliminary work shows that the HalfShop surrogate (set the number of jobs and machines to half of the corresponding value in original model) proposed in [94] which is used for DJSS, is not applicable for DFJSS. No prior knowledge is available to select appropriate surrogates for DFJSS problem. So, the key issue is how to design appropriate surrogates reasonably. Based on the assumption that the fidelity of the surrogate and the performance are positively correlated, two kinds of strategies are proposed in this work.

3.2.1 Adaptive Surrogates

In this section, adaptive surrogates are proposed for genetic programming and the corresponding algorithm is named as ASGP. The basic idea is to deliberately enlarge accuracy of the surrogate models by building up a very simple surrogates at the early stage. As the evolutionary optimisation proceeds, the accuracy of the surrogates increases gradually and smoothly expecting that the performance of approximated surrogate models is consistent with the original model.

Let N_{job} and N_{warmup} represent the number of jobs and warmup jobs, respectively. At the ith generation, the number of jobs and warmup jobs in the simulations during the fitness evaluation are denoted as $N_{job,i}$ and $N_{warmup,i}$. The expression of $N_{job,i}$ and $N_{warmup,i}$ are shown as Eq. (2) and Eq. (3), respectively.

$$N_{job,i} = \begin{cases} N_{job} * \frac{1}{maxGen-1} & gen = 0 \\ N_{job} * \frac{Gen}{maxGen-1} & 1 \leq Gen < maxGen \end{cases} \tag{2}$$

$$N_{warmup,i} = \begin{cases} N_{warmup} * \frac{1}{maxGen-1} & gen = 0 \\ N_{warmup} * \frac{Gen}{maxGen-1} & 1 \leq Gen < maxGen \end{cases} \tag{3}$$

In this way, the number of jobs will increase linearly as shown in Figure 3.7. Thus, adaptive surrogates with multi-fidelity models, which range from low-fidelity to high-fidelity detailed surrogates as the generation increases, are created.

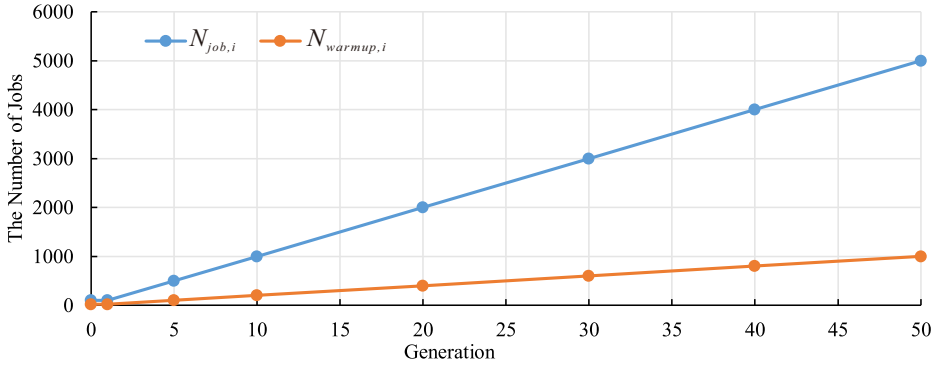


Figure 3.7: The variation trend of the number of jobs and warmup jobs along with generation in adaptive surrogates.

3.2.2 Generation-range-based Surrogates

For the ASGP, at each generation, different surrogate models are applied. In this section, generation-range-based surrogates is proposed for genetic programming (GSGP) to explore whether a fixed interval change can be more efficient.

In this work, the number of jobs and warmup jobs of the original simulation model are set to 5000 and 1000, respectively. We set every ten generations into a range. In each range, the generations will share the same surrogate while in different ranges, the generations will use different surrogates. The setting details of different surrogates used in different generations are shown in Figure 3.8.

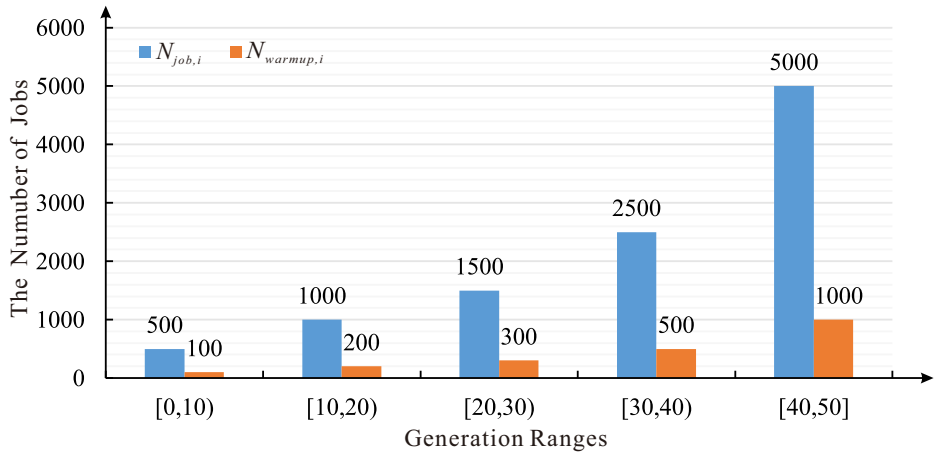


Figure 3.8: The number of jobs and warmup jobs in each generation in generation-range-based surrogates.

3.2.3 Experiment Design

The experiment design here is the same as in section 3.1.3.

3.2.4 Results and Analyses

The surrogate-assisted GP in evolutionary search with respect to the test performance, the training time and the learning process of three algorithms are investigated and discussed. The $(-,+)$ marks show whether our proposed approaches converge significantly better or poorer that the basic approach in Wilcoxon rank sum test $(p \leq 0.05)$, respectively. For the convenience of description, $< obj, uti >$ indicates the simulation scenarios, where obj and uti are the objective and the utilization level.

The goal of the proposed approaches are to evolve dispatching rules in different scenarios more efficient without sacrificing their performance in solving DFJSS problems.

Test Performance of Evolved Rules

Table 3.4 shows the mean and deviation error of ASGP, GSGP and CCGP. Overall, ASGP and GSGP algorithms are no significantly worse than CCGP. The mean value obtained by ASGP are about equal with the value obtained by CCGP in all scenarios. It is noted that ASGP significantly outperforms CCGP in scenario $< tmean, 0.85 >$. This clearly shows the potential of using surrogates to improve the performance of GP. It also indicates that the surrogates (approximation models) may not be always harm.

Table 3.4: The mean and deviation error of the compared algorithms over 30 independent runs for six scenarios.

Index	Scenario	ASGP	GSGP	CCGP
1	$< tmax, 0.85 >$	0.640(0.034)	0.638(0.029)	0.642(0.035)
2	$< tmax, 0.95 >$	0.571(0.023)	0.565(0.018)	0.568(0.030)
3	$< tmean, 0.85 >$	0.772(0.012)(-)	0.768(0.008)	0.772(0.015)
4	$< tmean, 0.95 >$	0.734(0.023)	0.738(0.022)	0.731(0.015)
5	$< twt, 0.85 >$	0.778(0.030)	0.772(0.010)	0.774(0.018)
6	$< twt, 0.95 >$	0.773(0.023)	0.774(0.024)	0.774(0.037)

For GSGP, the mean value obtained are slightly smaller than CCGP in four out of six scenarios. In addition, the variances obtained by GSGP are smaller than CCGP in five out of six scenarios.

Training Time

Table 3.5 shows the computation time (reductions produced by surrogates compared with CCGP) of the three algorithms. Overall, ASGP and GSGP need less computational requirements compared with CCGP. The average reductions produced by ASGP and GSGP are 25.7% and 34.4%, respectively.

Table 3.5: The average training time (reduction) of the compared algorithms over 30 independent runs for six scenarios.

Index	Scenario	Training Time (seconds)		
		ASGP	GSGP	CCGP
1	$< tmax, 0.85 >$	3399.8 (26.8%)	2969.9 (36.0%)	4642.8
2	$< tmax, 0.95 >$	3743.6 (27.2%)	3326.2 (35.3%)	5144.9
3	$< tmean, 0.85 >$	3302.5 (27.2%)	2935.0 (35.3%)	4538.5
4	$< tmean, 0.95 >$	3635.3 (25.0%)	3220.2 (33.6%)	4849.9
5	$< twt, 0.85 >$	3436.2 (22.9%)	3004.4 (32.6%)	4458.4
6	$< twt, 0.95 >$	3725.7 (24.8%)	3282.7 (33.8%)	4957.0

The experimental results have confirmed that ASGP can reduce the computation time by at least 22.9% in six scenarios. In both scenario 2 and scenario 3, the computation time are reduced the most (27.2%). For GSGP, it is obvious that it can reduce more computation time (from 32.6% to 36.0%) than ASGP (from 22.9% to 27.2%). It is not surprising because the average fidelity of ASGP is higher than GSGP. In addition, the computation time are reduced the most (36.0%) in scenario 1 while the least (32.6%) in scenario 5.

Insight the Learning Process

Figure 3.9 shows the convergence curves of evolutionary processes of ASGP, GSGP and CCGP in the training process. The lines in Figure 3.9 are the average normalized objective value from 30 independent runs. Although all GP methods start with the same population, the starting points are different because they use different surrogates. That is, CCGP get the value from surrogates with higher fidelities while ASGP and GSGP get the value from surrogates with lower fidelities.

It is noted that both ASGP and GSGP have higher fluctuations in all scenarios than CCGP, especially at the early stage of evolutionary process. For ASGP, the fidelities of surrogate models change more smoothly to handle the learning process gradually. It is expected to meet the need of training. It is interesting that Figure 3.9 shows that ASGP and GSGP have basically the same trends in six scenarios. This indicates that the predefined ranges and settings of simulations in GSGP are representative for the learning process. In addition, after generation 40, ASGP and GSGP can achieve almost the same learning ability as CCGP, although they use surrogates with lower fidelities at previous generations.

Figure 3.10 shows the convergence curves of the normalized objective value obtained by ASGP, GSGP and CCGP. It is obvious that CCGP can improve much faster at the beginning of the evolution in six scenarios. This benefits from the precise search with full simulations at the expense of computation time. However, after generation 10 approximately, the test performance between these three algorithms does not differ obviously.

Overall, taking the computational cost and test performance into consideration, the proposed algorithms are more promising than CCGP.

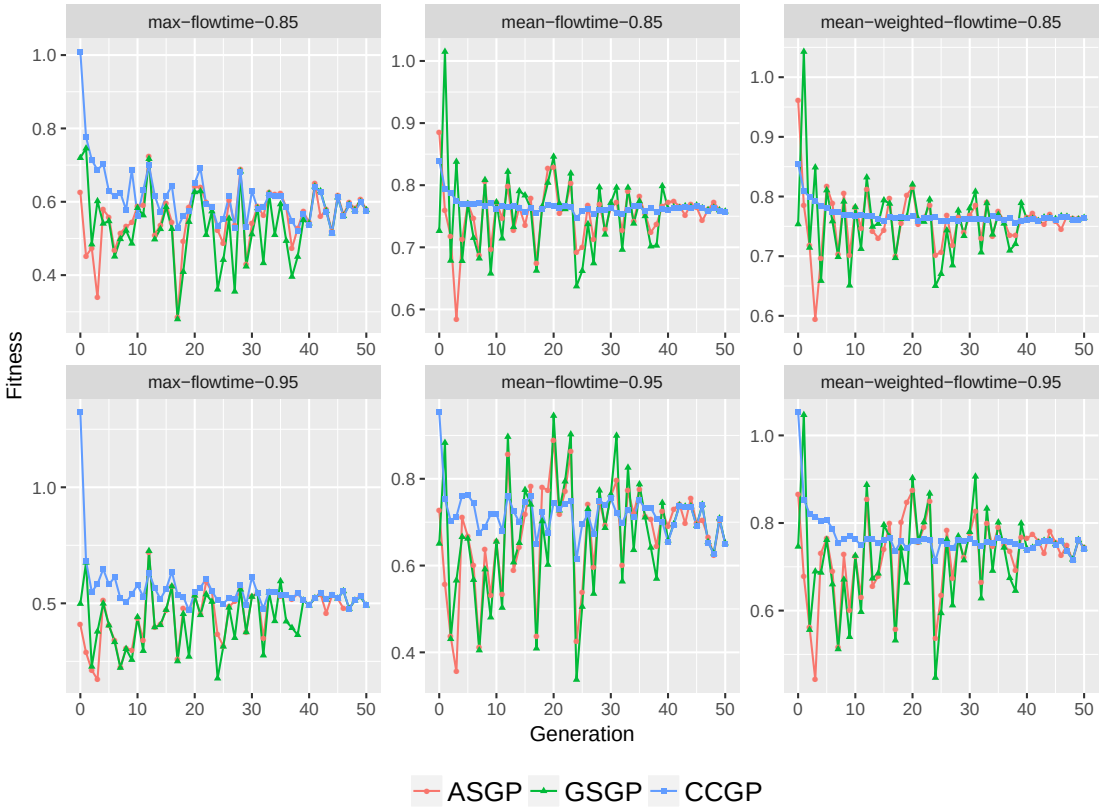


Figure 3.9: The convergence curves of the fitness value obtained by ASGP, GSGP and CCGP in the training processes.

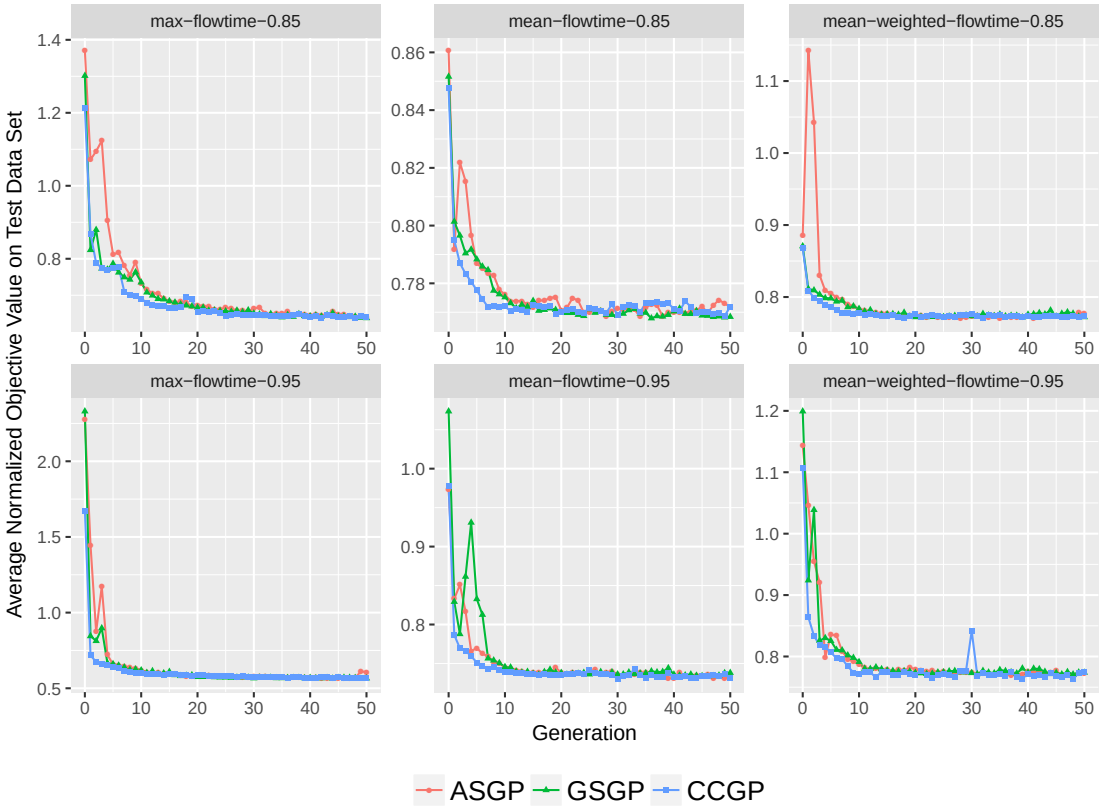


Figure 3.10: The convergence curves of the normalized objective value obtained by ASGP, GSGP and CCGP in test processes.

3.3 Summary

In this chapter, two approaches were developed. The first approach tried to evolve routing and sequencing rules based on GP with multi-tree representation simultaneously, which is one of the very first piece of work in this field. From the experimental results, we got some interesting findings. Firstly, in addition to performance, both the routing and sequencing rules evolved by MTGP and sMTGP are much smaller than the rules built by CCGP, which provides valuable study materials for analysing the rules. And also, MTGP and sMTGP take less training time. This is an important merit because required-time consuming training is a big limitation for genetic programming. Secondly, the proposed tree swapping crossover operator can enhance the ability of MTGP from the perspective of performance, rule size and training time in general. Thirdly, for average normalized objective values on test data set, there are more outliers obtained by CCGP. That is to say, the assumption in CCGP that routing and sequencing rules are independent and can be involved separately, might be not true. This indicates that when we evolve two rules at the same time, we would better to take the interaction into consideration.

In the second approach, in order to tackle the intensive computational requirements of genetic programming approach, this work proposed two different strategies of surrogates for genetic programming to automatically design dispatching rules for dynamic flexible job shop scheduling. It is a preliminary attempt to apply surrogates in the DFJSS problem. The result show that both the adaptive surrogates and the generation-range-based surrogates managed to reduce computation time without deteriorating the quality of the evolved rules and have the potential to get more promising dispatching rules.

In general, this chapter is the preliminary work that investigated different representation and incorporated surrogates to solve DFJSS problems. Some interesting observations have been analysis in this chapter. However, there are still many problems that need to be further investigated. It has been shown in the first work that the average rule size in the whole population becomes smaller with multi-tree representation. The reason will be further explored in the future. Also, it is worth interpreting and analysing the evolved rules to obtain further useful patterns. Also, it is important to further investigate different strategies surrogates to accelerate the effectiveness and efficiency. In the second work, only the level of problem approximation was applied. We will consider function approximation and evolutionary approximation to improve the effectiveness and efficiency of the GP approach.

Chapter 4

Proposed Contributions and Project Plan

The previous chapters have shown the main tasks and preparatory solutions for the tasks in this thesis. In this chapter, the proposed contributions in this thesis and project plan are presented to shape the directions for this thesis.

4.1 Proposed Contributions

This thesis will contribute to the fields of dynamic flexible job shop scheduling and genetic programming based on hyper-heuristic. The major contributions are listed as follows.

(a) This thesis will show how new GP approaches with different representations and new features with global respective, such as look-forward features, look backward features and estimator features, will improve performance for DFJSS. The novel GP approach with multi-tree representation is expected to evolve routing rule and sequencing rule simultaneously for DFJSS and achieve better performance than the state-of-the-art approach for DFJSS in terms of the *effectiveness* and *efficiency*. The designed grammar guided GP will be firstly used in DFJSS both for evolving routing and sequencing rules simultaneously and improving the *interpretability* of evolved rules. Finally, a comprehensive investigation of features for DFJSS, especially for routing, will be studied. The designed new features can provide more comprehensive search space to help GP find better solutions.

(b) This thesis will show how feature manipulation (feature selection and feature construction) can be used to solve the DFJSS problems and how the learning strategies (online learning and offline learning) can affect the performance of GP. The research in this field is very limited and this research will promote the research in this area. The proposed novel feature manipulation approaches and different learning strategies will help GP to explore more with the informative features. They are expected to use valuable information to improve the ability of GP to evolve dispatching rule *effectively* and *efficiently*. The search in this area can promote the development of terminal control strategies in the evolutionary process for GP and make its evolutionary process more flexible.

(c) This thesis will propose different types of surrogates based on problem approximation, function approximation and evolutionary approximation to improve the efficiency of

GP, and develop model management strategies to handle the surrogates appropriately in DFJSS. This research is a first attempt to combine advanced surrogates techniques with DFJSS. The surrogate-assisted GP is expected to evolve dispatching rules *efficiently* without scarifying its performance. This research will provide an efficient means of handling the complex JSS applications plagued with increasing high computation time. It will further strengthen the advantages of GP and promote its application to DFJSS.

(d) This thesis will investigate multi-objective techniques in dynamic flexible job shop scheduling. To our best of knowledge, no work has been reported in this area. Firstly, multi-objective techniques will be used to achieve a trade-off among different objectives. Secondly, the interpretability of evolved rules will be considered as one objective in the evolutionary process to balance performance and interpretability of evolved rules. This is very important factor for real-world application and a hot topic in JSS, because people would like to know how the evolved rule works. This work will give us more inspiration on how to balance the performance and interpretability, thus how to get more *interpretable* rules. Finally, the preference of users will be investigated to guide the algorithm to focus on the most interesting areas of the Pateto-optimal front. This work will make the search more targeted rather than searching for the whole pareto-optimal front, thus to find more *effective* schedules for different users.

4.2 Overview of Project Plan

The initial plan for this PhD project includes six main phases as shown in Table 4.1. The first phase has been completed and part of the rest phases have been done partly during the provisional registration period (12 months). The major research objectives will be covered from phase 2 to phase 6. The last phase involves writing the final thesis.

Table 4.1: Phases of project plan.

Phase	Description	Duration (Months)
1	Reviewing literature, overall design and writing the proposal	12 (complete)
2	Investigating new representations of GP to handle the DFJSS problems	4 (partly)
3	Developing feature manipulation for GP that can enhance the exploration and exploitation of GP	5
4	Proposing surrogates with different strategies for GP to improve its efficiency	5 (partly)
5	Investigating the measurements of interpretability and proposing approaches with multi-objective techniques to achieve a trade-off among different objectives	6
6	Writing the thesis	4

4.3 Project Timeline

A more detailed descriptions of the above plan are shown in Table 4.2 to get a specific plan that can help monitor the progress of this project.

Table 4.2: Project timeline for the next 24 months.

		Time in Months											
Phase	**Task**	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**	**22**	**24**
n/a	Update the literature review	×	×	×	×	×	×	×	×	×	×	×	×
2	Investigating grammar-based representation of GP to handle the DFJSS problems	×											
2	Designing look forward/backward terminals and terminals as estimators with global perspective		×										
3	Developing the feature selection mechanism for DFJSS			×									
3	Developing the feature construction mechanism for DFJSS			×	×								
3	Proposing an online learning strategy with feature selection for DFJSS					×							
4	Proposing surrogates based on function approximation for GP					×	×						
4	Proposing surrogates based on evolutionary approximation for GP						×						
4	Investigating model management for surrogates for GP							×					
5	Investigating multi-objective to achieve a trade-off among different objectives								×				
5	Investigating interpretability of evolved rules with multi-objective techniques to get more interpretable rules									×	×		
5	Investigating the preference of users to get more effective schedule for different users									×	×		
5	(optional) Proposing novel many-objective based approach to balance more than three objectives									×			
6	Writing the first draft of the thesis										×		
6	Editing the final draft											×	×

4.4 Thesis Outline

The outline of the final thesis will be written as follows.

- *Chapter 1: Introduction*

  In this chapter, problem statement, research goals, contributions and thesis outline will be presented.

- *Chapter 2: Literature Review*

  Job shop scheduling, especially dynamic flexible job shop scheduling, and the methodologies used for this problem will be described in detail in this chapter. Then, this chapter will review typical related work in hyper-heuristics , with a special focus on genetic programming hyper-heuristic for heuristic generation to generate dispatching rules.

- *Chapter 3: Representation of GP for DFJSS*

  This chapter will proposed new representation (multi-tree based and grammar-based) for GP which allows to evolve routing rule and sequencing rule at the same time for

DFJSS. Different representations of dispatching rules will be analysed to show their advantages and disadvantages and how they can solve the problem. In addition, this chapter will also design new terminals according to the characteristics of DFJSS and investigate their ability to improve the performance of GP.

- *Chapter 4: Evolving Dispatching Rules with Feature Manipulation for DFJSS*

  In this chapter, new feature manipulation (feature selection and feature construction) strategies will be investigated to improve the exploration and exploitation of GP to solve DFJSS problems. Particularly, an online learning of genetic programming with the feature selection approach will be developed.

- *Chapter 5: Surrogate-Assisted GP for DFJSS*

  This chapter will apply surrogates to improve the efficiency of GP without sacrificing its performance. Different strategies of surrogates will be compared in this chapter. Then, model management will also be investigated to further improve the performance of GP with surrogates.

- *Chapter 6: Incorporating Multi-objective for Evolving Dispatching Rules in DFJSS*

  This chapter will investigate multi-objective approaches to balance different targeted objectives and achieve a trade-off between the performance and interpretablity of evolved rules. In addition, the user's preference will also be considered to achieve more effective schedules for different users.

- *Chapter 7: Conclusions and Future Work*

  This chapter will summary the whole works in this thesis and the conclusions and findings from experiment and analyses in previous chapters will also be summarised. In addition, further research directions will be discussed arising from the contributions of this thesis.

4.5 Resources Required

4.5.1 Computing Resources

This research is based on experiment and need to run large computational experiments. In addition, multiple runs of the proposed approaches need to be performed to check the statistical significance of the obtained result. The grid computing facilities in school can meet the requirement and other IT resources in school can benefit this research.

4.5.2 Library Resources

Most of the published papers that relate to this research can be found in the university's electronic resources. Famous and useful textbooks and lectures notes are also available in the university library.

4.5.3 Conference Travel Grant

Some research results will be submitted to the major conferences in this area. Thus, grant from the university is needed to support the conference travels.

Bibliography

[1] AARTS, E., AARTS, E. H., AND LENSTRA, J. K. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[2] APPLEGATE, D., AND COOK, W. A computational study of the job-shop scheduling problem. *ORSA Journal on Computing 3*, 2 (1991), 149–156.

[3] BADER-EL-DEN, M., POLI, R., AND FATIMA, S. Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework. *Memetic Computing 1*, 3 (2009), 205.

[4] BANZHAF, W., NORDIN, P., KELLER, R. E., AND FRANCONE, F. D. *Genetic programming: an introduction*, vol. 1. Morgan Kaufmann San Francisco, 1998.

[5] BAYKASOĞLU, A., AND KARASLAN, F. S. Solving comprehensive dynamic job shop scheduling problem by using a grasp-based approach. *International Journal of Production Research 55*, 11 (2017), 3308–3325.

[6] BERTSEKAS, D. P., BERTSEKAS, D. P., BERTSEKAS, D. P., AND BERTSEKAS, D. P. *Dynamic programming and optimal control*, vol. 1. Athena Scientific Belmont, MA, 2005.

[7] BIRD, S., AND LI, X. Adaptively choosing niching parameters in a PSO. In *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006* (2006), pp. 3–10.

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

[9] BOKHORST, J. A., NOMDEN, G., AND SLOMP, J. Performance evaluation of family-based dispatching in small manufacturing cells. *International Journal of Production Research 46*, 22 (2008), 6305–6321.

[10] BRANKE, J., NGUYEN, S., PICKARDT, C. W., AND ZHANG, M. Automated design of production scheduling heuristics: A review. *IEEE Transactions on Evolutionary Computation 20*, 1 (2016), 110–124.

[11] BRUCKER, P., AND SCHLIE, R. Job-shop scheduling with multi-purpose machines. *Computing 45*, 4 (1990), 369–375.

[12] BURKE, E. K., HYDE, M., KENDALL, G., OCHOA, G., ÖZCAN, E., AND WOODWARD, J. R. A classification of hyper-heuristic approaches. In *Handbook of metaheuristics*. Springer, 2010, pp. 449–468.

[13] BURKE, E. K., HYDE, M., KENDALL, G., AND WOODWARD, J. A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics. *IEEE Transactions on Evolutionary Computation 14*, 6 (2010), 942–958.

[14] BURKE, E. K., HYDE, M. R., AND KENDALL, G. Evolving bin packing heuristics with genetic programming. In *Parallel Problem Solving from Nature-PPSN IX*. Springer, 2006, pp. 860–869.

[15] 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.

[16] BURKE, E. K., HYDE, M. R., KENDALL, G., AND WOODWARD, J. Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation* (2007), ACM, pp. 1559–1565.

[17] CARLIER, J., AND PINSON, É. An algorithm for solving the job-shop problem. *Management Science 35*, 2 (1989), 164–176.

[18] CARLIER, J., AND PINSON, E. Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research 78*, 2 (1994), 146–161.

[19] CHEN, H., CHU, C., AND PROTH, J.-M. An improvement of the lagrangean relaxation approach for job shop scheduling: a dynamic programming method. *IEEE Transactions on Robotics and Automation 14*, 5 (1998), 786–795.

[20] CHEN, H., IHLOW, J., AND LEHMANN, C. A genetic algorithm for flexible job-shop scheduling. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on* (1999), vol. 2, IEEE, pp. 1120–1125.

[21] CHENG, R., GEN, M., AND TSUJIMURA, Y. A tutorial survey of job-shop scheduling problems using genetic algorithms, part ii: hybrid genetic search strategies. *Computers & Industrial Engineering 36*, 2 (1999), 343–364.

[22] CHIANG, T.-C., AND FU, L.-C. Using dispatching rules for job shop scheduling with due date-based objectives. *International Journal of Production Research 45*, 14 (2007), 3245–3262.

[23] CHONG, C. S., SIVAKUMAR, A. I., LOW, M. Y. H., AND GAY, K. L. A bee colony optimization algorithm to job shop scheduling. In *Proceedings of the 38th Conference on Winter Simulation* (2006), Winter Simulation Conference, pp. 1954–1961.

[24] CORDELIA, L., DE STEFANO, C., FONTANELLA, F., AND MARCELLI, A. Genetic programming for generating prototypes in classification problems. In *2005 IEEE Congress Evolutionary Computation* (2005), vol. 2, pp. 1149–1155.

[25] DEB, K., AND JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Transation Evolutionary Computation 18*, 4 (2014), 577–601.

[26] DELL'AMICO, M., AND TRUBIAN, M. Applying tabu search to the job-shop scheduling problem. *Annals of Operations research 41*, 3 (1993), 231–252.

[27] DIMOPOULOS, C., AND ZALZALA, A. M. Investigating the use of genetic programming for a classic one-machine scheduling problem. *Advances in Engineering Software 32*, 6 (2001), 489–498.

[28] DOMINIC, P. D., KALIYAMOORTHY, S., AND KUMAR, M. S. Efficient dispatching rules for dynamic job shop scheduling. *The International Journal of Advanced Manufacturing Technology 24*, 1-2 (2004), 70–75.

[29] DURASEVIC, M., JAKOBOVIC, D., AND KNEZEVIC, K. Adaptive scheduling on unrelated machines with genetic programming. *Appl. Soft Comput. 48* (2016), 419–430.

[30] EGUCHI, T., OBA, F., AND TOYOOKA, S. A robust scheduling rule using a neural network in dynamically changing job-shop environments. *International Journal of Manufacturing Technology and Management 14*, 3-4 (2008), 266–288.

[31] GEIGER, C. D., AND UZSOY, R. Learning effective dispatching rules for batch processor scheduling. *International Journal of Production Research 46*, 6 (2008), 1431–1454.

[32] GEIGER, C. D., UZSOY, R., AND AYTUG, H. Autonomous learning of effective dispatch policies for flowshop scheduling problems. In *IIE Annual Conference. Proceedings* (2003), Institute of Industrial and Systems Engineers (IISE), p. 1.

[33] GEIGER, C. D., UZSOY, R., AND AYTUĞ, H. Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. *Journal of Scheduling 9*, 1 (2006), 7–34.

[34] GOLDBERG, D. E., AND HOLLAND, J. H. Genetic algorithms and machine learning. *Machine Learning 3* (1988), 95–99.

[35] GOMES, M. C., BARBOSA-PÓVOA, A. P., AND NOVAIS, A. Q. Reactive scheduling in a make-to-order flexible job shop with re-entrant process and assembly: a mathematical programming approach. *International Journal of Production Research 51*, 17 (2013), 5120–5141.

[36] GONÇALVES, J. F., DE MAGALHÃES MENDES, J. J., AND RESENDE, M. G. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of Operational Research 167*, 1 (2005), 77–95.

[37] GRUAU, F. On using syntactic constraints with genetic programming. In *Advances in Genetic Programming* (1996), MIT Press, pp. 377–394.

[38] HAUPT, R. A survey of priority rule-based scheduling. *Operations-Research-Spektrum 11*, 1 (1989), 3–16.

[39] HILDEBRANDT, T., AND BRANKE, J. On using surrogates with genetic programming. *Evolutionary Computation 23*, 3 (2015), 343–367.

[40] HILDEBRANDT, T., HEGER, J., AND SCHOLZ-REITER, B. Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation* (2010), ACM, pp. 257–264.

[41] HO, N. B., AND TAY, J. C. Evolving dispatching rules for solving the flexible job-shop problem. In *Congress on Evolutionary Computation* (2005), pp. 2848–2855.

[42] HOAI, N. X., MCKAY, R. I., AND ABBASS, H. A. Tree adjoining grammars, language bias, and genetic programming. In *European Conference on Genetic Programming* (2003), Springer, pp. 335–344.

[43] HUNT, R. *Genetic Programming Hyper-heuristics for Job Shop Scheduling*. Victoria University of Wellington, 2016.

[44] HUNT, R., JOHNSTON, M., AND ZHANG, M. Evolving less-myopic scheduling rules for dynamic job shop scheduling with genetic programming. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (2014), ACM, pp. 927–934.

[45] HUNT, R., JOHNSTON, M. R., AND ZHANG, M. *Evolving dispatching rules with greater understandability for dynamic job shop scheduling*. Citeseer, 2016.

[46] HURINK, J., JURISCH, B., AND THOLE, M. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum 15*, 4 (1994), 205–215.

[47] INGIMUNDARDOTTIR, H., AND RUNARSSON, T. P. Supervised learning linear priority dispatch rules for job-shop scheduling. In *International Conference on Learning and Intelligent Optimization* (2011), Springer, pp. 263–277.

[48] IWASAKI, Y., SUZUKI, I., YAMAMOTO, M., AND FURUKAWA, M. Job-shop scheduling approach to order-picking problem. *Transactions of the Institute of Systems, Control and Information Engineers 26*, 3 (2013), 103–109.

[49] JAKOBOVIĆ, D., AND BUDIN, L. Dynamic scheduling with genetic programming. In *European Conference on Genetic Programming* (2006), Springer, pp. 73–84.

[50] JAKOBOVIĆ, D., JELENKOVIĆ, L., AND BUDIN, L. Genetic programming heuristics for multiple machine scheduling. In *European Conference on Genetic Programming* (2007), Springer, pp. 321–330.

[51] JAYAMOHAN, M., AND RAJENDRAN, C. New dispatching rules for shop scheduling: a step forward. *International Journal of Production Research 38*, 3 (2000), 563–586.

[52] JAYAMOHAN, M., AND RAJENDRAN, C. Development and analysis of cost-based dispatching rules for job shop scheduling. *European Journal of Operational Research 157*, 2 (2004), 307–321.

[53] JIA, H., NEE, A. Y., FUH, J. Y., AND ZHANG, Y. A modified genetic algorithm for distributed scheduling problems. *Journal of Intelligent Manufacturing 14*, 3-4 (2003), 351–362.

[54] JIN, Y. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computation 9*, 1 (2005), 3–12.

[55] JIN, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation 1*, 2 (2011), 61–70.

[56] KACEM, I., HAMMADI, S., AND BORNE, P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 32*, 1 (2002), 1–13.

[57] KANET, J. J., AND LI, X. A weighted modified due date rule for sequencing to minimize weighted tardiness. *Journal of Scheduling 7*, 4 (2004), 261–276.

[58] KARUNAKARAN, D., MEI, Y., CHEN, G., AND ZHANG, M. Dynamic job shop scheduling under uncertainty using genetic programming. In *Intelligent and Evolutionary Systems*. Springer, 2017, pp. 195–210.

[59] KARUNAKARAN, D., MEI, Y., CHEN, G., AND ZHANG, M. Evolving dispatching rules for dynamic job shop scheduling with uncertain processing times. In *Evolutionary Computation (CEC), 2017 IEEE Congress on* (2017), IEEE, pp. 364–371.

[60] KARUNAKARAN, D., MEI, Y., CHEN, G., AND ZHANG, M. Toward evolving dispatching rules for dynamic job shop scheduling under uncertainty. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017* (2017), pp. 282–289.

[61] KARUNAKARAN, D., MEI, Y., CHEN, G., AND ZHANG, M. Sampling heuristics for multi-objective dynamic job shop scheduling using island based parallel genetic programming. In *Parallel Problem Solving from Nature - PPSN XV - 15th International Conference, Coimbra, Portugal, September 8-12, 2018, Proceedings, Part II* (2018), pp. 347–359.

[62] KOULINAS, G., KOTSIKAS, L., AND ANAGNOSTOPOULOS, K. A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. *Information Sciences 277* (2014), 680–693.

[63] KOZA, J. R. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*, vol. 34. Stanford University, Department of Computer Science Stanford, CA, 1990.

[64] KOZA, J. R. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing 4*, 2 (1994), 87–112.

[65] LANGDON, W. B. *Genetic programming and data structures: genetic programming + data structures = automatic programming!*, vol. 1. Springer, 2012.

[66] LAWLER, E. L., AND WOOD, D. E. Branch-and-bound methods: A survey. *Operations Research 14*, 4 (1966), 699–719.

[67] LENSEN, A., XUE, B., AND ZHANG, M. Generating redundant features with unsupervised multi-tree genetic programming. In *European Conference on Genetic Programming* (2018), Springer, pp. 84–100.

[68] LI, B., LI, J., TANG, K., AND YAO, X. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR) 48*, 1 (2015), 13.

[69] LI, X., EPITROPAKIS, M. G., DEB, K., AND ENGELBRECHT, A. P. Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Transation Evolutionary Computation 21*, 4 (2017), 518–538.

[70] LI, X., AND OLAFSSON, S. Discovering dispatching rules using data mining. *Journal of Scheduling 8*, 6 (2005), 515–527.

[71] LUKE, S. Essentials of metaheuristics. lulu, 2013. *Available for free at http://cs. gmu. edu/~ sean/book/metaheuristics*.

[72] MALVE, S., AND UZSOY, R. A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. *Computers & Operations Research 34*, 10 (2007), 3016–3028.

[73] MASOOD, A., CHEN, G., MEI, Y., AND ZHANG, M. Reference point adaption method for genetic programming hyper-heuristic in many-objective job shop scheduling. In *Evolutionary Computation in Combinatorial Optimization - 18th European Conference, EvoCOP 2018, Parma, Italy, April 4-6, 2018, Proceedings* (2018), pp. 116–131.

[74] MASOOD, A., MEI, Y., CHEN, G., AND ZHANG, M. Many-objective genetic programming for job-shop scheduling. In *IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016* (2016), pp. 209–216.

[75] MASOOD, A., MEI, Y., CHEN, G., AND ZHANG, M. A pso-based reference point adaption method for genetic programming hyper-heuristic in many-objective job shop scheduling. In *Artificial Life and Computational Intelligence - Third Australasian Conference, ACALCI 2017, Geelong, VIC, Australia, January 31 - February 2, 2017, Proceedings* (2017), pp. 326–338.

[76] MCKAY, R. I., HOAI, N. X., WHIGHAM, P. A., SHAN, Y., AND O'NEILL, M. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines 11*, 3-4 (2010), 365–396.

[77] 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*, 5 (2017), 339–353.

[78] MEI, Y., NGUYEN, S., AND ZHANG, M. Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling. In *Simulated Evolution and Learning - 11th International Conference, SEAL 2017, Shenzhen, China, November 10-13, 2017, Proceedings* (2017), pp. 435–447.

[79] MEI, Y., NGUYEN, S., AND ZHANG, M. Evolving time-invariant dispatching rules in job shop scheduling with genetic programming. In *European Conference on Genetic Programming* (2017), Springer, pp. 147–163.

[80] MEI, Y., AND ZHANG, M. A comprehensive analysis on reusability of gp-evolved job shop dispatching rules. In *CEC* (2016), pp. 3590–3597.

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

[82] MONTANA, D. J. Strongly typed genetic programming. *Evolutionary Computation 3*, 2 (1995), 199–230.

[83] MUNI, D. P., PAL, N. R., AND DAS, J. A novel approach to design classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation 8*, 2 (2004), 183–196.

[84] NGUYEN, S. Automatic design of dispatching rules for job shop scheduling with genetic programming.

[85] NGUYEN, S. A learning and optimizing system for order acceptance and scheduling. *The International Journal of Advanced Manufacturing Technology 86*, 5-8 (2016), 2021–2036.

[86] NGUYEN, S., MEI, Y., XUE, B., AND ZHANG, M. A hybrid genetic programming algorithm for automated design of dispatching rules. *Evolutionary Computation* (2018), 1–31.

[87] 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. In *IEEE Congress on Evolutionary Computation* (2012), pp. 1–8.

[88] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Transactions on Evolutionary Computation 17*, 5 (2013), 621–639.

[89] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Learning iterative dispatching rules for job shop scheduling with genetic programming. *The International Journal of Advanced Manufacturing Technology 67*, 1-4 (2013), 85–100.

[90] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Learning reusable initial solutions for multi-objective order acceptance and scheduling problems with genetic programming. In *Genetic Programming - 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings* (2013), pp. 157–168.

[91] 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*, 2 (2014), 193–208.

[92] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Selection schemes in surrogate-assisted genetic programming for job shop scheduling. In *Asia-Pacific Conference on Simulated Evolution and Learning* (2014), Springer, pp. 656–667.

[93] NGUYEN, S., ZHANG, M., AND TAN, K. C. Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015* (2015), pp. 2781–2788.

[94] 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.

[95] NGUYEN, S., ZHANG, M., AND TAN, K. C. Adaptive charting genetic programming for dynamic flexible job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, Kyoto, Japan, July 15-19, 2018* (2018), pp. 1159–1166.

[96] NGUYEN, S. B. S., AND ZHANG, M. A hybrid discrete particle swarm optimisation method for grid computation scheduling. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 483–490.

[97] NIE, L., SHAO, X., GAO, L., AND LI, W. Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems. *The International Journal of Advanced Manufacturing Technology 50*, 5-8 (2010), 729–747.

[98] NOWICKI, E., AND SMUTNICKI, C. A fast taboo search algorithm for the job shop problem. *Management Science 42*, 6 (1996), 797–813.

[99] OKTAVIANDRI, M., HASSAN, A., AND SHAHAROUN, A. M. Decision support tool for job shop scheduling with job cancellation. *International Conference on Engineering of Tarumanagara (ICET)* (2013).

[100] OLAFSSON, S., AND LI, X. Learning effective new single machine dispatching rules from optimal scheduling data. *International Journal of Production Economics 128*, 1 (2010), 118–126.

[101] OLTEAN, M., AND GROSAN, C. A comparison of several linear genetic programming techniques. *Complex Systems 14*, 4 (2003), 285–314.

[102] O'NEILL, M., AND RYAN, C. Grammatical evolution: Evolutionary automatic programming in a arbitrary language, volume 4 of genetic programming, 2003.

[103] PAPADIMITRIOU, C. H., AND STEIGLITZ, K. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[104] PARK, J., MEI, Y., CHEN, G., AND ZHANG, M. Niching genetic programming based hyper-heuristic approach to dynamic job shop scheduling: an investigation into distance metrics. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion* (2016), ACM, pp. 109–110.

[105] PARK, J., MEI, Y., NGUYEN, S., CHEN, G., JOHNSTON, M., AND ZHANG, M. Genetic programming based hyper-heuristics for dynamic job shop scheduling: cooperative coevolutionary approaches. In *European Conference on Genetic Programming* (2016), Springer, pp. 115–132.

[106] PARK, J., MEI, Y., NGUYEN, S., CHEN, G., AND ZHANG, M. Investigating the generality of genetic programming based hyper-heuristic approach to dynamic job shop scheduling with machine breakdown. In *Australasian Conference on Artificial Life and Computational Intelligence* (2017), Springer, pp. 301–313.

[107] PARK, J., MEI, Y., NGUYEN, S., CHEN, G., AND ZHANG, M. Investigating a machine breakdown genetic programming approach for dynamic job shop scheduling. In *European Conference on Genetic Programming* (2018), Springer, pp. 253–270.

[108] PARK, J., MEI, Y., NGUYEN, S., CHEN, G., AND ZHANG, M. An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling. *Appl. Soft Comput. 63* (2018), 72–86.

[109] PEARL, J. Heuristics: intelligent search strategies for computer problem solving.

[110] PEZZELLA, F., MORGANTI, G., AND CIASCHETTI, G. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research 35*, 10 (2008), 3202–3212.

[111] PICKARDT, C. W., HILDEBRANDT, T., BRANKE, J., HEGER, J., AND SCHOLZ-REITER, B. Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal of Production Economics 145*, 1 (2013), 67–77.

[112] PILLAY, N. An analysis of representations for hyper-heuristics for the uncapacitated examination timetabling problem in a genetic programming system. In *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology* (2008), ACM, pp. 188–192.

[113] PILLAY, N., AND BANZHAF, W. A genetic programming approach to the generation of hyper-heuristics for the uncapacitated examination timetabling problem. In *Portuguese Conference on Artificial Intelligence* (2007), Springer, pp. 223–234.

[114] PINEDO, M., AND SINGER, M. A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. *Naval Research Logistics (NRL) 46*, 1 (1999), 1–17.

[115] POLI, R., LANGDON, W. B., MCPHEE, N. F., AND KOZA, J. R. *A field guide to genetic programming*. Lulu. com, 2008.

[116] RILEY, M., MEI, Y., AND ZHANG, M. Improving job shop dispatching rules via terminal weighting and adaptive mutation in genetic programming. In *Evolutionary Computation (CEC), 2016 IEEE Congress on* (2016), IEEE, pp. 3362–3369.

[117] ROSS, P. Hyper-heuristics. In *Search methodologies*. Springer, 2005, pp. 529–556.

[118] SABUNCUOGLU, I., AND BAYIZ, M. Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research 126*, 3 (2000), 567–586.

[119] SELS, V., GHEYSEN, N., AND VANHOUCKE, M. 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*, 15 (2012), 4255–4270.

[120] SHAHZAD, A., AND MEBARKI, N. Learning dispatching rules for scheduling: A synergistic view comprising decision trees, tabu search and simulation. *Computers 5*, 1 (2016), 3.

[121] SHIUE, Y.-R. Data-mining-based dynamic dispatching rule selection mechanism for shop floor control systems using a support vector machine approach. *International Journal of Production Research 47*, 13 (2009), 3669–3690.

[122] TAY, J. C., AND HO, N. B. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering 54*, 3 (2008), 453–473.

[123] VAN LAARHOVEN, P. J., AND AARTS, E. H. Simulated annealing. In *Simulated annealing: Theory and applications*. Springer, 1987, pp. 7–15.

[124] VEPSALAINEN, A. P., AND MORTON, T. E. Priority rules for job shops with weighted tardiness costs. *Management Science 33*, 8 (1987), 1035–1047.

[125] WECKMAN, G. R., GANDURI, C. V., AND KOONCE, D. A. A neural network job-shop scheduler. *Journal of Intelligent Manufacturing 19*, 2 (2008), 191–201.

[126] XIONG, J., XING, L.-N., AND CHEN, Y.-W. Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns. *International Journal of Production Economics 141*, 1 (2013), 112–126.

[127] YIN, W.-J., LIU, M., AND WU, C. Learning single-machine scheduling heuristics subject to machine breakdowns with genetic programming. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on* (2003), vol. 2, IEEE, pp. 1050–1055.

[128] YSKA, D., MEI, Y., AND ZHANG, M. Feature construction in genetic programming hyper-heuristic for dynamic flexible job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2018, Kyoto, Japan, July 15-19, 2018* (2018), pp. 149–150.

[129] YSKA, D., MEI, Y., AND ZHANG, M. Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling. In *European Conference on Genetic Programming* (2018), Springer, pp. 306–321.

[130] ZHANG, Q., ZHOU, A., ZHAO, S., SUGANTHAN, P. N., LIU, W., AND TIWARI, S. Multiobjective optimization test instances for the cec 2009 special session and competition. *University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report 264* (2008).

[131] ZHAO, H. A multi-objective genetic programming approach to developing pareto optimal decision trees. *Decision Support Systems 43*, 3 (2007), 809–826.

[132] ZITZLER, E., THIELE, L., LAUMANNS, M., FONSECA, C. M., AND DA FONSECA, V. G. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transation Evolutionary Computation 7*, 2 (2003), 117–132.