RESEARCH ON AUTOMATIC GENERATION OF DISPATCHING RULES USING GENETIC PROGRAMMING FOR JOB SHOP SCHEDULING PROBLEMS

SHADY AMGAD AHMED AHMED SALAMA

(Degree)
博士（工学）

(Date of Degree)
2022-09-25

(Date of Publication)
2023-09-01

(Resource Type)
doctoral thesis

(Report Number)
甲第8466号

(URL)
https://hdl.handle.net/20.500.14094/0100477892

DOCTORAL DISSERTATION

RESEARCH ON AUTOMATIC GENERATION OF DISPATCHING RULES USING GENETIC PROGRAMMING FOR JOB SHOP SCHEDULING PROBLEMS (ジョブショップスケジューリング問題に対する GENETIC PROGRAMMING を用いたディスパッチングルールの自動生成に関する研究)

By

SHADY AMGAD AHMED AHMED SALAMA

July 2022

Department of Systems Science

Graduate School of System Informatics

Kobe University, Japan

DEDICATION

"This is in memory of my dad who always believed in me.

You are gone but your belief in me has made this journey possible.

As you look down from heaven, I hope you are proud of your son"

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Professor Toshiya Kaihara, for his patience, guidance, and support. He has been an ideal teacher, and mentor who kindly provided knowledge and expertise in writing and editing this thesis. His encouraging words, thoughtful, and detailed feedback have been very important to me. Moreover, this endeavour would not have been possible without the support, valuable suggestions and guidance of Associate Professor Nobutada Fujii. I am also indebted to the continuous assistance of Assistant Professor, Daisuke Kokuryo, who provided invaluable comments on my analysis and writing as well as prompt responses to my late night and early morning emails. His constructive comments and insights helped improve the quality of the published papers and this thesis.

I would also like to express my deepest appreciation to Professor Takashi Kamihigashi for giving me sufficient time and a comfortable working environment while working on this thesis. I am also extremely grateful to Japanese government for granting me the Monbukagakusho scholarship, both as a research student and a doctoral student.

There are many people that I would like to thank. I am so grateful for the time these people have given me. My sincere thanks go to Mrs. Mayuko Sakamoto for her patience and presence in resolving challenges I encountered that made my experience in Japan easier and more enjoyable. Also, I express special gratitude to my dear friend Nursultan Nikhanbayev for the selfless help he has willingly given since my admission to CS21 Lab that has helped me quickly adjust to my new life in Japan. Many thanks go to Dr. Imen for her continuous encouragement when I felt frustrated and unmotivated. Also, it is my pleasure to thank my friends Mohamed, Mina, and Emeline whose constant support helped me to be a better version of me. My accomplishments are because they believed in me. Lastly, my family deserves endless gratitude. I am grateful to my mother and sister for their unconditional and endless support and encouragement throughout my PhD study.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This dissertation focuses on the automatic generation of high-quality dispatching rules in compact structures and low computational requirements using the Genetic Programming (GP) to solve static and dynamic Job Shop Scheduling Problems (JSSPs). Precisely, the main objective is to reduce the computational burden required by the GP algorithm to evolve high-quality dispatching rules for both static and dynamic JSSPs. Five approaches are proposed in this thesis to address the limitations of the conventional research. Two of these approaches are developed to generate scheduling rules for the static JSSPs, whereas the other three approaches deal with the dynamic JSSPs.

Regarding the static JSSPs, two main limitations have been reported in the literature. The first limitation is premature convergence caused by low diversity among GP individuals that leads to low solution quality, whereas the second one is the high computational costs of GP approaches due to significant growth in the size of generated rules without a tangible return in fitness values, known as the bloat effect. Consequently, a distance metric is introduced to measure the genotypic similarity between the GP individuals and the best-evolved rule in this thesis. The proposed metric overcomes the limitations of the current metrics by considering the interaction effect between nodes and their parents, does not require additional simulation runs, and gives higher priority to the nodes closest to the root node. The aim is to represent population diversity in a numerical format that can be optimized and thus improve the exploration ability of the GP algorithm by avoiding early convergence. Therefore, a multi-objective GP framework is proposed by integrating Non-dominated Sorting Genetic Algorithm II (NSGA-II) with the GP algorithm to optimize three objectives simultaneously. The three considered objectives are diversity values estimated using the proposed metric, rule length, and solution quality. To assess the effectiveness of the proposed distance metric and multi-objective GP framework, two algorithms are developed and compared with three algorithms from the literature across ten static JSSP instances using makespan and mean tardiness as objective functions. Experimental results show the effectiveness of the proposed methods in generating a diverse population of high-quality rules with smaller sizes in a shorter computational budget compared with the conventional methods.

Regarding the dynamic JSSPs, two major limitations have been reported in the literature. The first limitation is similar to the static problems which is the high

computational time of the GP algorithm due to the bloat effect. In contrast to the static problems where benchmark instances are used for fitness assessment, a Discrete Event Simulation (DES) model is the most common approach for the dynamic JSSPs. Therefore, the second limitation is the high fitness evaluation costs to assess the fitness values of evolved rules using a DES model. In order to address the first limitation, this thesis proposes a feature selection approach to reduce the size of evolved rules. The proposed approach uses a probabilistic selection scheme to estimate the weight of given terminals instead of the binary discrimination usually used in conventional methods. In addition, the proposed approach does not require pilot GP runs as the conventional methods. Because it uses the evolutionary information collected from previous generations in estimating the weights of the terminals in the next generation in an online manner. The proposed approach (PGP) is compared with three GP algorithms and 30 manually-made rules from the literature under different job shop configurations and scheduling objectives, including total weighted tardiness, mean tardiness, and mean flow time. Experimentally obtained results demonstrate that the proposed approach outperforms the other conventional methods in generating more compact rules in a shorter computational time.

Gene Expression Programming (GEP) algorithm is a modified version of the tree-based GP algorithm to evolve dispatching rules using a fixed-linear representation that is less susceptible to the bloat effect. Therefore, this thesis modifies the feature selection approach proposed for the GP algorithm to be applicable to the GEP algorithm for the dynamic JSSPs. The aim is to evaluate the effect of imposing an additional constraint on the size of evolved rules in the case of the contained GP representation. The proposed approach adds two main points to the existing literature. First, it is the first attempt to control the bloat effect for constrained GP representations in the literature on the automated design of scheduling rules. Second, it increases the likelihood of using the proposed approach in more complex manufacturing environments due to the significant reduction in training time. The proposed algorithm is compared with three algorithms from the literature using three objective functions namely total weighted tardiness, mean tardiness, and mean flow time. Experimental results confirm the ability of the proposed algorithm in evolving rules with smaller sizes in a shorter computational time without sacrificing performance. The weight of terminals across generations of the proposed algorithm is compared with the weights obtained using the PGP algorithm. Results

demonstrate the ability of the proposed feature selection approach to identify the same set of critical terminals regardless of the evolutionary algorithm used.

Finally, a surrogate assisted GEP approach is introduced to reduce the time of expensive fitness assessments of dispatching rules generated for dynamic JSSPs. Reducing fitness assessment times significantly speeds up the most computationally demanding step of the GP algorithm. The proposed approach extends the conventional methods through three main contributions. First, it is the first attempt to use machine learning to abstract a DES model of DJSSPs. Second, it reduces fitness evaluation time without significantly affecting prediction accuracy. Third, it is independent of the structure of evolved rules, and thus it can be adopted with other hyper-heuristic approaches. Three surrogate models are developed by integrating the proposed approach with three simplified models from the literature. The proposed surrogates are compared with their counterparts from the literature using mean tardiness and mean flow time objective functions. Experimental results prove that the proposed surrogates have significantly lower computational costs with a neglectable loss in prediction accuracy across different training and testing scenarios.

It is verified that the proposed approaches significantly reduced the computational time of the GP algorithm to automatically evolve high-quality scheduling rules in compact structures compared to conventional methods. In addition, the performance of the proposed approaches is evaluated under different job shop environments including static and dynamic instances using a combination of the most common scheduling objectives such as makespan, mean flow time, mean tardiness, and total weighted tardiness. The behaviour of the best-evolved rules for each objective function is also analysed and several useful insights are gained. Finally, experimental results demonstrate that the proposed approaches enhance the performance of various stages of the GP algorithm starting from representation, then selection, until fitness evaluations.

Chapter 1. INTRODUCTION

This chapter begins by providing a background on job shop scheduling problems from an academic and practical perspective. Then, the current solution approaches in the literature, and their capabilities and limitations for solving job shop scheduling problems are presented. Moreover, broad overviews are given in the field of the automated design of scheduling rules using hyper-heuristic techniques with a major focus on genetic programming methods. Literature limitations, research objectives, and major contributions of this thesis are also shown in subsequent sections. Finally, the organization of this thesis is illustrated.

1.1 Background

1.1.1 Job Shop Scheduling Problem

Industrial facilities strive to cope with unexpected market fluctuations, meet customers' requirements, and utilize available resources in the best possible way. Most manufacturers today have started with so-called “job shops” which are a type of small manufacturing systems where customized products are made with a relatively small production sequence (Pinedo, 2012). Typically, job shops make a variety of custom or semi-custom parts for other businesses in small to medium-sized orders, for instance, the automotive industry, semiconductor manufacturing facilities, and many others. Although job shops increase the production flexibility in making a large variety of customized products to meet customer demands, it is very challenging to efficiently schedule the production processes due to high product variability and non-standard production flow (Jones et al., 1998). Resulting in a decrease in the levels of utilization

of production resources, an increase in the lead time of jobs, and an increase in production costs. Therefore, solving production scheduling problems can help decision-makers reduce inventory costs, increase throughput, and improve resource usage (Jones et al., 1998). Over the past 60 years, the Job Shop Scheduling Problem (JSSP) has been studied extensively in academia and industry because of its broad real-world applications to manufacturing and cloud computing (Nguyen et al., 2013a). From the academic point of view, most JSSPs are NP-hard problems. Therefore, the computational time required to solve JSSPs increases exponentially with the increase in the size of the problem (Pinedo, 2012). On the other hand, from a practical perspective, there are many manufacturing facilities around the world that share the same characteristics of the job shop models including semiconductor manufacturers and automobile assembly lines (Jones et al., 1998). Therefore, developing effective models for JSSPs may not only increase throughput and decrease costs in job shops, but also in many related industries resulting in a significant impact on the global economy.

As shown in Figure 1.1, the job shop consists of a number of jobs (tasks) that need to be processed by a limited number of machines (resources). Each job has a specific set of operations that must be processed according to a set of technical and precedence constraints (arrows with the same colour as their job). The objective of the JSSP is to achieve a schedule of jobs that optimizes some predefined objectives related to productivity (reducing completion time) or the level of customer satisfaction (meeting due dates) (Pinedo, 2012). JSSPs can be categorized into two main classes of problems depending on whether the operational information is available beforehand. They are Static JSSPs (SJSSPs) and Dynamic JSSPs (DJSSPs). In static job shop scheduling, all jobs are ready to process at time zero. Also, all the information related to all jobs and machines on the shop floor is available when making a schedule. On the other hand, in dynamic job shop scheduling, jobs constantly arrive following deterministic or stochastic arrival patterns, and no operational information is known before the arrival of the job (Jakobović and Budin, 2006). Although it is inevitable to avoid unexpected events during the production process, static scheduling is more widely covered in the literature as compared with dynamic scheduling. Stochastic nature is always inherent in real job shops, for example, the processing time of operations varies due to different skill levels among operators, urgent jobs can arrive at any moment, and machines may suddenly breakdown (Ouelhadj and Petrovic, 2009).

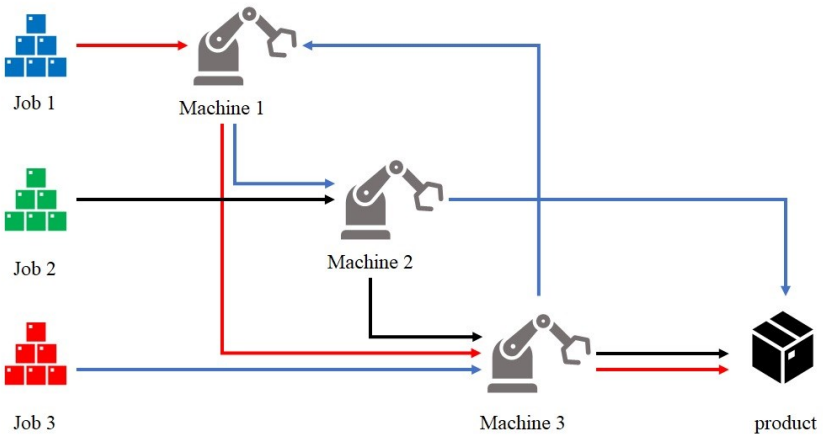


Figure 1.1: Example of a job shop scheduling problem

1.1.2 Solution Approaches

Due to the availability of knowledge in SJSSPs, exact optimization approaches have been proposed to achieve optimal solutions such as integer linear programming (Simon and Takefuji, 1988), branch-and-bound (Lawler and Wood, 1966), Lagrangian relaxation (Kaskavelis and Caramanis, 1998), and dynamic programming (Gromicho et al., 2012). However, the use of exact methods has been limited to solving small SJSSP instances due to the exponential increase in computational cost with increasing problem size (NP-hard problem). Therefore, approximate methods known as heuristics have been developed to get satisfactory solutions in an acceptable computational budget. Approximate methods can be classified into improvement and construction heuristics. Although improvement heuristics such as genetic algorithms (Park et al., 2003), simulated annealing (Akram et al., 2016), and ant colony algorithms (Flórez et al., 2013) do not guarantee optimality, they are able to find quasi-optimal solutions for static JSSPs within a reasonable computational time. In addition, improvement heuristics are not suitable for handling dynamic events because it would be computationally expensive to frequently modify the obtained schedule (running the heuristic again with the new information) to any change that occurs in the system. In contrast, construction heuristics such as dispatching rules are one of the most common optimization approaches used to solve DJSSPs (Ouelhadj and Petrovic, 2009). Moreover, Aytug et al. categorized existing strategies for solving DJSSPs into three classes: completely reactive approaches, robust pro-active approaches, and predictive–reactive approaches (Aytug et al., 2005). The authors reported that dispatching rules are the most popular completely reactive approach used in many real-life production systems.

Dispatching rules are mathematical functions of attributes of the job shop and machines used to prioritize all jobs awaiting processing on a given machine. In other words, a dispatching rule determines which job must be processed next when a machine becomes idle. The main reasons behind the popularity of dispatching rules are their flexibility to incorporate domain knowledge, ease of implementation, scalability to solve large problem instances, low computational cost, and prompt response to dynamic events (Nguyen et al., 2017a). Moreover, dispatching rules have been integrated with other optimization techniques in order to generate effective initial solutions. Many efficient manually designed dispatching rules covering a wide variety of job shop environments and objectives have been proposed in the literature. More details are available in several surveys (Baker, 1984; Blackstone et al., 1982; Dominic et al., 2004). In addition, there are several comparative studies in the literature to analyse the performance of traditional scheduling rules under different manufacturing conditions, for example (Mizrak and Bayhan, 2006) and (Sels et al., 2012a). Three main conclusions were drawn from these studies as follows.

1) Scheduling rules are problem-specific solution methods developed to deal with specific job shop settings under a certain objective function. Therefore, they perform poorly on other system configurations or performance measures.
2) Dispatching rules which combine multiple attributes together (composite rules) have better performance compared with simple rules with only one attribute.
3) Manual design of high-quality rules that usually include a large number of attributes in complex formulas is a challenging task. Because it requires a significant amount of time and code effort, great domain knowledge, and extensive empirical testing.

1.1.3 Automated Design of Dispatching Rules Using Genetic Programming

In order to meet the limitations of manually designed dispatching rules, several researchers have suggested taking advantage of advances in computational power and machine learning methods to automate the process of generating heuristics for hard optimization problems known as "*hyper-heuristics*". In other words, hyper-heuristics are high-level search methodologies that explore the search space of low-level heuristics rather than the search space of solutions to the underlying problem (Drake et al., 2020). The objectives to be achieved through this approach are as follows.

1) Raise the level of generality by discovering the right heuristic for a particular problem instead of solving a single problem instance directly.
2) Reduce the time and effort needed by experts to design efficient heuristics for different problem instances (job shop settings).
3) Gain useful insights by exploring widely diverse, undiscovered high-quality heuristics.

Burke et al. proposed a classification of hyper-heuristic approaches based on their search mechanism including heuristic *selection* and heuristic *generation* methods (Burke et al., 2013). A brief discussion of hyper-heuristic applications covering a wide range of scheduling and combinatorial optimisation problems was also presented. Heuristic selection methods seek to choose a set of low-level heuristics for different problem instances. Low-level heuristics can be simple operators (neighbourhood moves or basic local search operations), metaheuristics, or even hyper-heuristics. On the other hand, the goal of heuristic generation methods is to develop new high-level heuristics by making use of the components (features or operations used in existing heuristics) of known heuristics. In addition, hyper-heuristics can be categorized into *supervised* and *unsupervised* methods based on the machine learning technique used. Regarding the automatic design of scheduling rules, Genetic Programming (GP) has been shown to be a promising unsupervised heuristic generation approach that dominates other hyper-heuristic methods (Branke et al., 2015).

GP is a type of evolutionary computation method (a subset of machine learning) derived from the model of biological evolution and its core mechanisms. GP algorithm has been used in many areas, such as discovering the functional relationship between features (symbolic regression) (Amir Haeri et al., 2017), grouping data into specific classes (classification) (Jabeen and Baig, 2010), software engineering (Afzal and Torkar, 2011), and even aiding in the design of antennas and electrical circuits (Lohn et al., 2005). Regarding the use of the GP algorithm for the automatic generation of scheduling heuristics, (Miyashita, 2000) is expected to be the first study that proposed the GP approach to evolve dispatching rules for a JSSP using a multi-agent model, where each agent dispatches the operations on the resources under its control. Since then, there has been a growing interest among researchers in using the GP algorithm for evolving scheduling rules. Because GP does not depend on any assumptions and can be easily extended to deal with various production scheduling problems (Branke et al., 2016a).

The GP framework for the automated design of scheduling rules consists of two main modules, the GP reasoning module and the fitness evaluation module, as shown in Figure 1.2. The GP reasoning module is responsible for generating a candidate population of dispatching rules whose fitness values have to be estimated using the fitness evaluation module. The GP algorithm generates dispatching rules using predefined sets of terminals and functions, and a specific representation. The terminal set includes relevant jobs, machines, and shop floor attributes (features). Typically, most of these attributes (features) are derived from manually designed rules in the literature based on the nature of the scheduling problem under investigation. Also, the functions set consists of a set of arithmetic functions such as addition, subtraction, multiplication, and division, as well as some logical operators such as minimum, maximum, IF conditions, etc. In terms of the GP representation, an expression tree structure is the standard representation used in many previous studies (Đurasević et al., 2016; Hildebrandt et al., 2010a; Shady et al., 2020a). To ensure the validity of the generated rules and they are syntactically correct, certain grammar has to be imposed. Grammar defines how the individual components (functions and terminals) can be grouped to yield valid mathematical functions (dispatching rules) (Branke et al., 2016a).

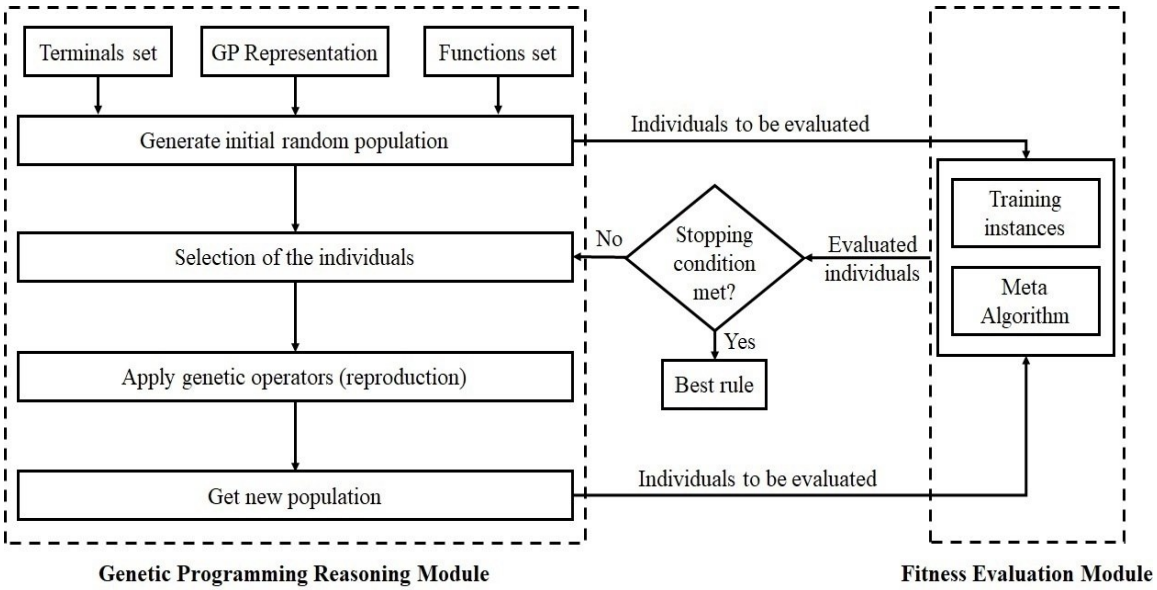


Figure 1.2: GP framework for the automated design of dispatching rules

Figure 1.3 shows a simple grammar for constructing priority functions using three terminals (PT: Processing Time, DD: Due Date, and CT: Current Time) and three arithmetic functions. Also, an example of a GP individual that can be generated using this grammar is shown in Figure 1.3 in both expression tree and mathematical forms.

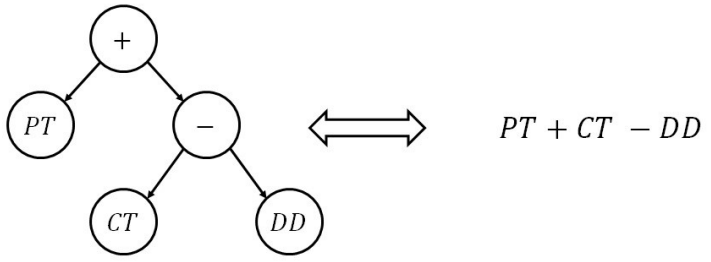
$< start > ::= < node >$

$< node > ::= < function >< node >< node > | < terminal >$

$< function > ::= + | - | \times$

$< function > ::= PT \mid DD \mid CT$

Simple GP grammar



$PT + CT - DD$

Expression Tree **Mathematical format**

Figure 1.3: Example of a simple grammar and a random GP individual

The fitness values of the GP rules are estimated in the performance evaluation model that consists of two main components, training instances, and a meta-algorithm. In the case of static scheduling problems, a set of static training instances obtained from real-world situations is used for fitness evaluation. There is a wide range of benchmark static job shop instances in the literature including OR-library (Beasley, 1990), and randomly generated instances of varying levels of difficulty (instance size) (Storer et al., 1992; Taillard, 1993; Yamada and Nakano, 1992). In contrast, a discrete event simulation model is the most common simulation technique used to estimate the solution quality of scheduling rules (Nguyen et al., 2014a). Moreover, a meta-algorithm must be defined that specifies the job shop logic and constraints for a given scheduling problem. It also specifies how a particular dispatching rule will be used to create a valid schedule (Shady et al., 2020b). After all GP individuals are evaluated, the GP reasoning mechanism selects the best individuals (parents) using some selection techniques such as the roulette method (Holland, 1992), tournament selection (Blickle and Thiele, 1995), etc. to form the mating pool.

Afterward, genetic operators are applied to generate a new population of dispatching rules (offspring). The standard genetic operators used in the GP literature are subtree crossover and mutation operators (Willis et al., 1997). The crossover operator combines the genetic information of two parents to evolve a new offspring, whereas the mutation operator introduces new random information in one parent to get a new rule. The aforementioned steps are considered as one evolutionary iteration (generation). The

population evolves over a predefined number of generations depending on the available computational time. Finally, if the stopping criteria are satisfied, then the algorithm terminates and the best rule is returned; otherwise, another evolutionary iteration begins by following the same steps (Tay and Ho, 2008).

The GP algorithm offers several major advantages compared with other hyper-heuristics such as decision trees, logistic regression, and artificial neural networks, which can be mentioned as follows (Branke et al., 2016a; Nguyen et al., 2017a).

1) **Variable-length encoding representation:** The main difference between the GP algorithm and other optimization algorithms, such as a genetic algorithm that uses the same evolutionary strategies, is the representation of solutions. The common representation used in the GP algorithm is the tree structure with a variable depth, as opposed to the genetic algorithm that uses a fixed string of numbers to represent a solution (J. R. Koza, 1994a). This flexible representation is able to simultaneously explore the structure and corresponding parameters of a heuristic, thereby covering a larger area of the heuristic search area (Nguyen et al., 2017a). Another key benefit of the GP variable representation is that the best scheduling rule (structure and contents) for a given scheduling problem is usually not known in advance, and therefore it is not reasonable to impose a fixed structure on all generated rules (Hildebrandt et al., 2010a).

2) **Availability of multi-objective optimization techniques:** Although the GP algorithm has a different encoding scheme compared with other evolutionary algorithms, most of the current multi-objective optimization techniques available in the evolutionary computation literature can be easily integrated with the GP approaches. For instance, Tay and Ho provided the first work that aimed to solve flexible multi-objective DJSSP with respect to minimizing makespan, mean tardiness, and mean flow time objectives (Tay and Ho, 2008). In this study, the objective function is constructed by combining the objectives into a linear weighted sum in which all the objectives have the same priority. Moreover, Nguyen et al. combined the GP approach with NSGA-II and SPEA2 methods to evolve scheduling rules for multi-objective DJSSP under a single simulation scenario (Nguyen et al., 2015a). Finally, the authors of one study (Masood et al., 2016) have integrated the NSGA-III algorithm, which is one of the latest and well-known multi-objective optimization methods, with the GP algorithm for

creating dispatching rules under static job shop settings while optimizing five performance measures.

3) **The obtained results can be analysed and partially interpreted:** Typically, GP individuals are represented using the expression tress structure composed of leaf nodes (features) and internal nodes (arithmetic functions and logic operators). The tree structure is one of the most intuitive ways to represent mathematical functions. Therefore, there is no need to use sophisticated decoding methods to convert GP individuals into a human-readable format (Branke et al., 2016a). The evolved rules can also be directly provided to a discrete event simulation model of a given scheduling problem for fitness assessment without any pre-processing.

4) **An increasing number of GP articles for production scheduling problems and promising results have been reported in the existing literature**: Nguyen et al. provided a comprehensive review of existing studies from 2000 to 2017 on using GP for automated design of production scheduling rules (Nguyen et al., 2017a). They reported that although there were only four papers between 2000 and 2004 on this topic, there has been significant growth in the number of studies since 2010, reaching 69 papers between 2010 and 2017. Recently, a book has been published reporting several successful applications of GP approaches in developing high-quality dispatching rules that outperform common human-made rules across a wide variety of job shop scheduling environments and objectives (F. Zhang et al., 2021d).

1.2 Limitations of Existing Studies

Although the dispatching rules evolved using the genetic programming algorithm have obtained better performance compared with manually made rules in the literature, the use of the GP to automatically generate scheduling rules is relatively new. Therefore, there are many limitations and research opportunities in existing GP approaches. As shown in Figure 1.4, the current limitations can be categorized into two key classes including limitations of using the GP approach in static scheduling problems, and dynamic scheduling problems. Although the two classes share the same problem with the high computational costs of GP, they differ with respect to population diversity and the time required for fitness assessments.

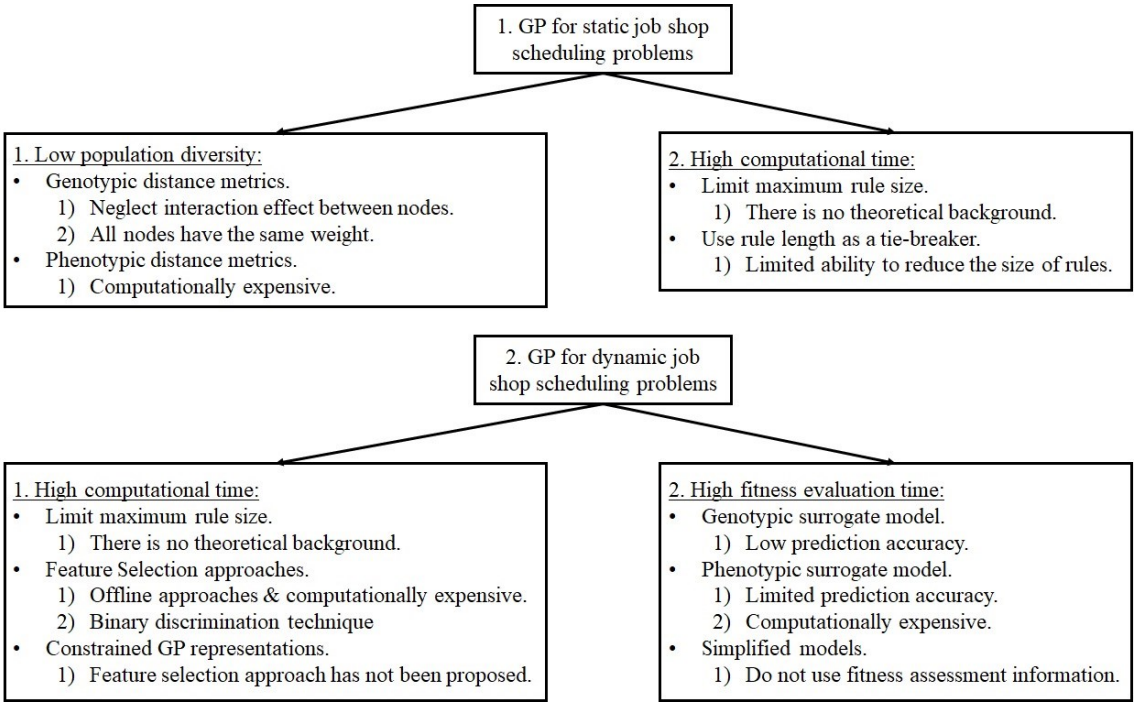


Figure 1.4: Challenges of using the GP approach for static and dynamic JSSPs, current solution approaches (bullets), and their limitations (numbering)

1) The limitations of automatically generating dispatching rules in SJSSPs using the GP approaches are as follows.
   a) The first challenge is the premature convergence (low solution quality) caused by the **low diversity** among GP individuals. Because all operational information is available in static scheduling problems and no unexpected events occur during the scheduling horizon, thereby the scheduling problem does not change during the evolution generations. Consequently, a GP individual that gets a high fitness value in one generation will achieve the same high performance in the upcoming generations. Thus, this rule has a higher probability of survival in the following generations and other individuals will copy its genetic information. Resulting in a well-known phenomenon in the evolutionary computation literature called "*premature convergence*" (Pandey et al., 2014). Premature convergence is defined as the condition in which the GP individuals converge very early to a suboptimal region due to a loss of diversity within the population leading to low-quality solutions. In order to increase diversity between individuals, several distance metrics have been developed. The type of metrics that measures similarity (distance) between individuals based on their syntactic structure is called "*genotypic*" metrics, whereas "*phenotypic*" metrics assess similarity based on the actual performance of compared rules (Burke et al., 2002).

There are two limitations in the existing genotypic metrics as follows.

- They only consider the position of the nodes in GP individuals while neglecting the interaction effect between a specific node and its parents.
- Most distance metrics assume that all nodes have the same weight, which is not the case in scheduling rules where nodes closer to the root have a greater influence than nodes farther away.

Regarding phenotypic metrics,

- They are computationally expensive compared with genotypic metrics because they require fitness evaluations of individuals in measuring the distance between them.

b) The second challenge is the **high computational costs** of GP approaches due to the unjustified, significant growth in the size of generated rules across generations without a tangible return in fitness values. This is a common behaviour commonly observed in GP algorithms and other variable-length genomes methods and is known as the "*bloat effect*" (Luke and Panait, 2006). The bloating effect negatively affects GP searching ability in three ways.

- It shows down the search process by wasting computing resources in evaluating large individuals (complex mathematical functions) with many redundant elements.
- It reduces the probability that genetic operators will alter important parts of evolved rules.
- It reduces the possibility of evolving rules being interpreted by decision-makers, and thus reduces the chances of their use in industry (Mori et al., 2008).

Therefore, two main methods are commonly used in the literature to impair the bloating effect in the GP approaches. The first method is to specify the maximum allowable size of evolved rules using the maximum tree depth or the maximum number of nodes at each rule. The main drawback of this approach is the lack of theoretical background (trial and error is the only technique available) on how to determine the maximum size of evolved rules because the size of the best base is not known in advance (Nguyen et al., 2017a). The second method is to consider the size of evolved rules as an objective that can be optimized using multi-objective optimization approaches. Generally, this approach indirectly optimizes the size of the rules, that is, if there are two rules

with the same fitness values in the selection stage, then the GP algorithm selects the rule with a smaller size (tiebreaker) (Burks and Punch, 2015). The main limitation of this approach is that in the case where the GP population is very diverse (large range of fitness values), it will have no or minimal effect in reducing the size of evolving bases.

2) The limitations of automatically generating dispatching rules in DJSSPs using the GP approaches are as follows.
   a) The first challenge is similar to that in the static scheduling problems which is the **high computational time** of the GP approaches due to the bloating effect caused by relatively different causes. In contrast to the case in SJSSPs, the GP population in DJSSPs is highly diverse due to the influence of dynamic events. Therefore, the fitness value of specific rule changes from one generation to the next due to the use of a new random seed in each generation, which changes the configuration of the job shop under study (processing times, due dates, job arrivals, etc.) (Hildebrandt et al., 2010a). This reduces the benefit that can be obtained from the use of the bloating control methods used in the static methods, although the maximum depth of the tree is frequently used (Branke et al., 2016a). Moreover, due to the large variability among evolved rules with different tree structures and contents, it is difficult to distinguish between significant and irrelevant terminals. Therefore, two approaches have been proposed to reduce the size of evolved rules which are *feature selection methods*, and *constrained GP representations*. Although the existing feature selection methods have been successfully reduced the size of GP rules in several studies without compromising the quality of generated rules, they suffer from two major drawbacks (Mei et al., 2016, 2017a; F. Zhang et al., 2021a).
      - Most of the current methods are offline feature selection methods which means they require multiple expensive GP runs (pre-processing) to determine the important terminals to be used in subsequent GP runs (F. Zhang et al., 2021a). Therefore, they have huge overhead computational costs. In addition, the use of an attribute vector to represent the importance of each terminal on its corresponding rule is a promising approach to reducing the size of the GP rules (Nguyen et al., 2018a). Its main limitation is that attribute vectors do not provide accurate information about their priority functions because they do not consider

situations in which a particular attribute might not be present in the priority function (Shady et al., 2021a).

- The existing feature selection methods use a binary discrimination method, i.e., inclusion or exclusion of a particular feature from the terminal set, which ignores the relative importance of the respective terminals at different stages of the GP run (Nguyen et al., 2018a).

Regarding the use of constrained GP representations to control bloating effect, the Gene Expression Programming (GEP) algorithm introduced in (Ferreira, 2001) has been used as an alternative to the GP algorithm when the size of evolved rules is significantly important. GEP algorithm uses fixed-length linear strings (chromosomes) to represent expression trees of various shapes and sizes, thereby the bloating effect is not as pronounced as in the GP algorithm. However, the rules evolved using the GEP approach might contain irrelevant terminals because there is no direct way to eliminate their occurrence. To the best of our knowledge, there is no feature selection approach proposed in the literature for the GEP representation.

b) The second challenge in the automatic generation of scheduling rules in DJSSPs is the **high fitness evaluation time** that adds to the overall computational costs of the GP algorithm. Discrete Event Simulation (DES) models of the dynamic job shops are usually developed to introduce stochastic variables such as job arrivals, due dates, processing times, etc. in an easier and more flexible way as compared with analytical methods. However, DES models used in the case of DJSSPs have much greater computational needs in contrast to static instances used in SJSSPs. Therefore, computationally cheaper fitness evaluation models known as "*surrogates*" have been introduced to reduce the fitness evaluation time of DES models. Two surrogate models have been proposed to overcome this challenge in the field of automated design of dispatching rules for DJSSPs using GP. The literature surrogate models are:

    - Phenotypic characterization model (Hildebrandt and Branke, 2015a).
    - Simplified models (Nguyen et al., 2017d).

Regarding the phenotypic characterization model, the surrogate model is based on a decision vector that estimates the fitness of a given rule by using the same fitness value of the most similar rules generated in the previous generation. One drawback is that the dimensions of the decision vectors must be large

enough to adequately distinguish between evolving rules. Another limitation of this surrogate model is that the prediction accuracy is very low, as reported in (Nguyen et al., 2017d). In contrast, the simplified surrogate models do not use a decision vector or any performance-related information from previous generations. The main idea of these models in reducing computational time is to use smaller versions of the actual job shop understudy, i.e., a smaller number of jobs and machines. Although the simplified models achieved high prediction accuracy with a significant reduction in computational costs, they do not use any information collected during simulation evaluation.

1.3 Research Objectives

The overall objective of this thesis: is to automatically generate high-quality dispatching rules in concise structures with low computational costs for the static and dynamic job shop scheduling problems by enhancing the performance of the GP algorithm. Although dispatching rules are widely used in real-world applications, it has been noted that the main limitation on the widespread use of GP approaches to automatically evolve rules is the high computational burden, which always takes hours or even days of training time as well as the complexity in understanding the behaviour of evolved rules. The size of GP rules is not only one of the main reasons for the high computational time of GP approaches, but also has a direct impact on interpretability because complex (large) rules are more challenging to interpret by decision-makers than simple (short) rules. Thus, these rules are less likely to be adopted in the industry. It is noteworthy that all GP approaches proposed in this thesis were developed and evaluated in a specific problem domain of the GP algorithm, which is the automatic generation of scheduling rules, and thus other GP applications are out of the scope of this work.

As shown in the previous section, the performance of the GP algorithm in generating high-quality dispatching rules relies heavily on the problem under investigation. In other words, the GP limitations in the field of automatic generation of scheduling rules are different in the case of SJSSPs than in the case of DJSSPs. Therefore, the overall research objective can be divided into five sub-objectives. In addition, these objectives are categorized into two major groups in the same manner as used in the literature limitations section, as shown in Figure 1.5:

1. Objectives related to the automatic generation of dispatching rules in SJSSPs using the GP approach.

2. Objectives related to the automatic generation of dispatching rules in DJSSPs using the GP approach.

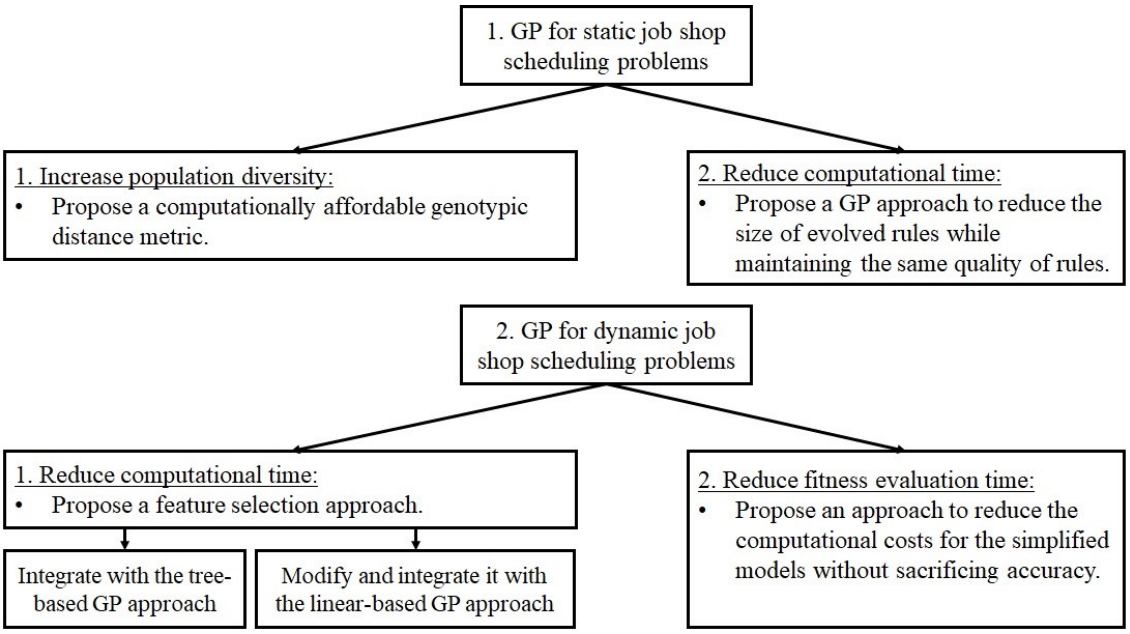


Figure 1.5: Research objectives that are addressed in this thesis with the proposed solution approach for each objective.

Thus, the details of the objectives addressed in this thesis are presented as follows.

1) Objectives related to the automatic generation of dispatching rules in **SJSSPs** using the GP approach:
   a) The first objective is to **increase diversity** among GP rules to overcome the GP premature conversion, and thus increase the quality of evolved rules. This objective consists of two main challenges. The first challenge is how to measure the similarity of GP individuals and represent them in numerical form. In other words, a distance metric should be developed that captures the main characteristics of generated rules and brings out the similarities between them in an easy-to-optimize format. Second, the proposed distance metric needs to be computationally efficient so as not to overburden the GP algorithm. Therefore, this thesis develops a new distance metric to estimate the similarity between the evolved rules considering these two challenges. The proposed distance metric relies on the genotypic similarity between GP rules and the best rule evolved so far, thereby it has lower computational costs compared with literature phenotypic distance metrics. Also, it addresses the limitations in the existing genotypic metrics by considering the interaction effect between nodes and their parents, as well as giving a higher weight to nodes near the root than to more

distant nodes. The distance metric estimates similarity between two individuals with a numerical value between 0 and 1, where "0" indicates that the two rules are completely different and "1" indicates they are identical. Therefore, in order to increase population diversity, the similarity values of GP individuals have to be maximized, which is addressed in the following objective.

b) The second objective is to **reduce the computational time** of the GP algorithm by reducing the size of evolved rules. Controlling the bloating effect will also increase the understandability of evolved rules, which increases the likelihood that the best-generated rules will be used in real-world scheduling problems. There is a major challenge in restricting the size of generated rules without negatively affecting their performance. Specifically, when the size of evolved rules alone is considered as an objective, then smaller rules will be generated (selected through generations) even if they have poor performance. In contrast, if the solution quality alone is considered as an objective, then high-quality rules will be generated without any limitations regarding their size. Therefore, a multi-objective GP framework is proposed for SJSSPs to simultaneously optimize diversity value (using the proposed distance metric), solution quality, and size of generated rule. The idea behind considering the three objectives simultaneously with the same weight is due to the conflicting nature between the solution quality and rule size, and between diversity value and rule size (small rules have a smaller diversity range compared with large rules). Consequently, rules with higher diversity values, smaller sizes, and higher performance have a higher probability of surviving across generations. It is worth noting to consider diversity value and solution quality at the same time when selecting rules helps in balancing the exploration and exploitation ability of the GP algorithm (better search space exploration).

2) Objectives related to the automatic generation of dispatching rules in **DJSSPs** using the GP approach:

a) The first objective is to **reduce the computational time** needed for the tree-based GP approach. Although the tree representation helps the GP algorithm to obtain dispatching rules of different sizes and contents, it is the most susceptible representation to the bloating effect. As mentioned in the literature limitations section 1.2, maximum tree depth is the most widely used method to control bloating in the existing studies although it cannot guarantee the quality of evolved rules. In contrast, feature selection using attribute vectors proposed in

(Nguyen et al., 2018a) and constrained GP representation used in (Ozturk et al., 2019) achieved promising results regarding the performance and size of evolved rules. Consequently, this thesis aims to enhance the performance of these two promising methods. Regarding feature selection, the limitations in the existing feature selection methods stated previously are addressed in the proposed approach as follows.

- This thesis develops an online feature selection method for the GP algorithm that uses the evolutionary information generated in previous generations to select important terminals in an adaptive manner.
- The proposed feature selection method does not require any GP runs, and uses a probabilistic discrimination scheme (probability of selecting a specific terminal) to choose terminals instead of the binary method used in the literature.

Regarding the constrained GP representations, this challenge is considered in more detail in the following research objective.

b) The second objective is to **reduce the computational time** of the GP algorithm in the case of using a GP constrained representation. In contrast to the first objective, where the tree-based (variable-length representation) GP approach is used, the fixed-linear representation of the GEP algorithm is used. Changing the encoding scheme of individuals not only affects the tendency of the GP individuals to increase in size, but also determines the set of genetic operators that can be used based on the chosen representation (Ferreira, 2001). Therefore, the feature selection approach that gets good results in the case of the tree-based GP algorithm does not guarantee that similar performance will be achieved in the case of the GEP algorithm; i.e., better, similar, or worse results might be obtained. Therefore, this objective can be broken down into the following research questions.

- How to propose a feature selection method applicable to the GEP representation? To the best of our knowledge, there is no feature selection method in the literature for the GEP algorithm.
- Is it effective to integrate feature section methods with a fixed-length GP representation such as the GEP algorithm? and if yes, would they negatively affect the GEP exploration ability as it is already restricted?

In order to answer these questions, this thesis modifies the feature selection approach proposed for the GP algorithm to be applicable to the GEP algorithm. Afterward, the performance of integrating the modified feature selection approach with the GEP algorithm is assessed across different DJSSP instances and objective functions.

c) The third objective is to **reduce the fitness evaluation time** of generated individuals. The main reason behind the high computational costs is the use of DES models to imitate the behaviour of the dynamic job shop understudy. Therefore, this objective is concerned with overcoming the following challenges.
   - Propose a surrogate model to reduce the simulation length (runtime) of DES models used in fitness assessment without miss-ranking the performance of evolved rules.
   - The proposed model has to be independent of the structure of the GP evolved rules in order to be applicable to other GP approaches.

   The simplified models proposed in (Nguyen et al., 2017d) are the latest models used to reduce fitness assessment in the field of automated design of dispatching rules while maintaining high prediction accuracy. Therefore, this thesis proposes an approach to reduce the computational costs of the simplified models without sacrificing accuracy. Specifically, the proposed approach aims to reduce the simulation length of the simplified models by collecting fitness-related information during evaluating the performance of a small set of GP rules (training rules). Then, a machine learning model is trained using the collected information to develop a simple function (surrogate model) that will replace the excluded simulation interval. Finally, the remaining rules are evaluated using the simplified model with a shortened simulation length and the developed surrogate model.

1.4 Major Contributions

In order to address the five aforementioned sub-objectives in the previous section, this thesis offers five major contributions to the current literature on the automatic generation of scheduling rules using the GP algorithm for JSSPs as follows.

1) In order to increase the diversity of generated rules that positively affect the quality of their solutions in the case of SJSSPs, this thesis introduces a distance

metric to measure the similarity between GP individuals based on their genotypic characterization. The proposed metric differs from the similarity metrics in the literature as follows:

a) It considers not only the positions of the nodes but also the interaction effect between the nodes and their parents in estimating similarity values.
b) It is a genotypic-based similarity measure, thereby there is no need to estimate fitness values for evolved rules using expensive simulation runs as for phenotypic metrics.
c) It prioritizes the weight of the nodes closest to the root node compared with those farthest to match the nature of the rules used in JSSPs.

2) For the sake of creating high-quality dispatching rules for SJSSPs in a smaller computational budget and concise structures, this thesis proposes a multi-objective framework by integrating NSGA-II with the GP algorithm to simultaneously optimize diversity value, rule length, and solution quality in static job shop settings. The proposed approach helps the GP algorithm avoid premature convergence and reduce the size of evolved rules while maintaining high fitness values. In contrast to the existing methods, the proposed approach has the following advantages.
    a) The framework considers the size of GP rules as a direct objective to be optimized rather than using it as a "tiebreaker" when two rules have the same fitness values. This reduces the size of generated rules which has a significant impact on reducing computational time and increasing the interpretability of the best rules.
    b) The framework does not depend on the system settings of the job shop understudy or the performance measures to be improved.
3) In order to increase the interpretability of evolved rules and speed up the process of the automatic generation of scheduling rules, this thesis proposes a feature selection approach to reduce the size of GP rules and reduce GP computational costs by modifying the attribute vector proposed in (Nguyen et al., 2018a). The proposed feature selection approach extends the GP literature by offering the following advantages.
    a) The attribute vector is strictly linked to its corresponding rule which helps in gathering useful information from complex tree structures. In addition, any change that might occur in the attribute vector of a specific rule will have a direct impact on the performance of the rule.

b) The attribute vector uses an adaptive selection scheme to estimate the probability of selecting a particular terminal instead of the binary discrimination usually used in literature methods.
c) The feature selection approach uses the evolutionary information collected from previous generations using attribute vectors of rules to estimate the weights of the terminals in the next generation.

4) In order to check the applicability of the proposed approach when using fixed representation to encode dispatching rules, this thesis modifies the feature selection approach proposed for the GP algorithm to be applicable to the GEP algorithm where evolved rules are represented using linear chromosomes. The integration of the proposed feature selection approach with the GEP algorithm adds the following points to the existing literature.
   a) The use of feature selection to control bloating in constrained GP representations has not yet been reported in the literature on the automated design of scheduling rules.
   b) It increases the likelihood of using the proposed approach in more complex manufacturing environments due to the significant reduction in training time resulting from both restricted search space and the ability to select important features only.
5) In order to speed up the automatic generation of dispatching rules, this thesis proposes three surrogate models to reduce the time needed for fitness assessment which is the most computationally demanding step. The surrogate models aim to replace part of the simulation length of an expensive DES model with a simple mathematical function. This significantly reduces computational times due to the large number of generated rules that must be evaluated. Consequently, the proposed surrogate models achieve the following advantages.
   a) They are built on the simplified models that get higher prediction accuracy compared with the phenotypic model (Nguyen et al., 2017d).
   b) They have a low computational budget and the same prediction accuracy compared with the simplified models (Nguyen et al., 2017d).
   c) They only depend on the behaviour of the DES model, and therefore they can be used with algorithms other than GP or GEP algorithms.
   d) It is the first attempt to use machine learning to abstract a DES model of DJSSPs. Therefore, there is a wide range of research opportunities to develop other machine learning techniques using the same concept.

1.5 Thesis Outline

The remainder of this thesis is organized as shown in Figure 1.6. Chapter 2 provides a literature review of related work where problem domains are highlighted in blue, and literature limitations are highlighted in red. Chapter 3 is the only chapter focusing on the SJSSPs. The major contributions of this thesis are presented in four chapters from Chapter 3 to Chapter 6 where the proposed solution approaches are highlighted in green. Chapter 7 presents the conclusions and future directions for research. An overview of each chapter is presented as follows.

Chapter 2 provides a review of production optimization techniques with an emphasis on job shop scheduling. In addition, a detailed explanation of several solution approaches that have been proposed for different job shop scheduling environments is given, with the advantages and limitations of each method. The basic concepts of dispatching rules, hyper-heuristics, and automated design of scheduling heuristics are illustrated. Moreover, a review of current research articles related to the use of the GP algorithm to automatically generate dispatching rules is presented. Finally, the limitations of the existing literature are discussed in detail.

Chapter 3 covers the proposed distance metric for measuring similarities between dispatching rules in case of static job shop scheduling problems. The multi-objective GP framework is proposed to optimize the fitness, diversity, and size of the evolved rules. Finally, the performance of the proposed framework is evaluated across multiple instances (different sizes and difficulties) of SJSSP and two objective functions.

Chapter 4 proposes a feature selection approach for identifying important terminals and excluding insignificant ones during GP runs. Existing articles that have developed feature selection methods in the field of automated design of dispatching rules are also briefly discussed. Moreover, the limitations of current feature selection methods are analysed. Finally, the performance of integrating the proposed feature selection with the GP algorithm is shown in terms of computational time, rule length, and solution quality.

Chapter 5 build on the work of Chapter 4 by modifying the feature selection approach proposed for the tree-based GP algorithm to be applicable to the fixed-length GEP algorithm. The effectiveness of the proposed approach is evaluated under different dynamic job shop scheduling instances and three scheduling objective functions. Moreover, the results obtained using the GEP feature selection approach are compared

with those obtained using the literature approaches with respect to computational time, the average size of evolved rules, and solution quality.

Chapter 6 proposes a surrogate assisted GEP approach to reduce the evaluation time needed to estimate the performance of evolved rules. In addition, current surrogate models from the literature are illustrated. Finally, the proposed surrogates are compared with their counterparts from the literature regarding computational time and prediction accuracy.

Chapter 7 provides a summary of the findings of this thesis as well as a list of possible future research directions.

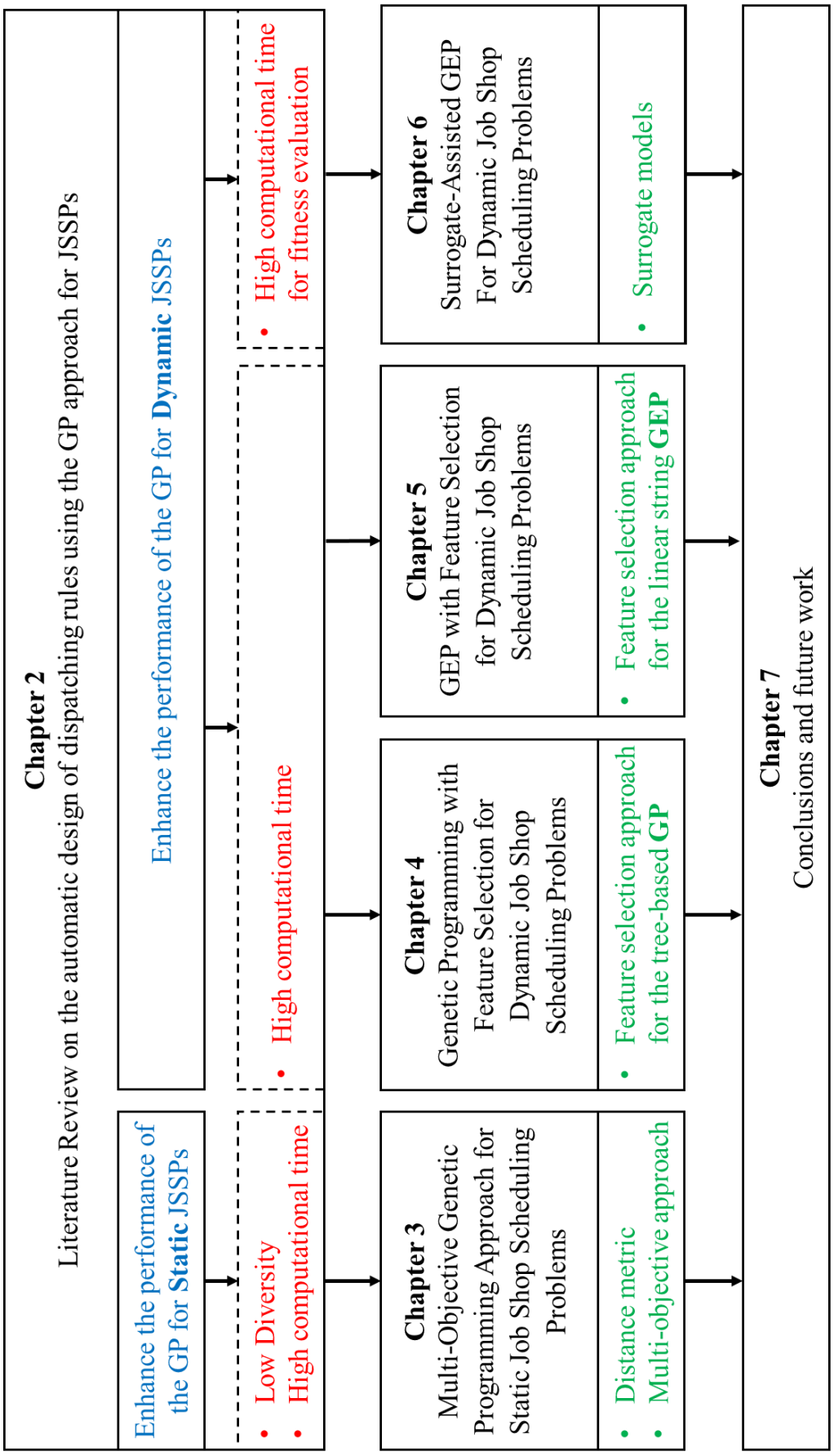


Figure 1.6: Structure of this thesis including problem domain, literature limitations, and proposed approaches used in each chapter

Chapter 2. LITERATURE REVIEW

This chapter begins by introducing the impact of the manufacturing sector on the global economy. The definition of the production planning and control concept and its main components are given, with the main focus on production scheduling activities. Then, job shop scheduling problems are described, including terminologies, notations, problem formulation, classes of schedules, etc. Moreover, a review of solution approaches is provided for both static and dynamic job shop settings, in which a deeper analysis is made of evolutionary algorithms and dispatching rules. Then, an overview of the types of hyper-heuristics used to select or generate heuristics is provided. This chapter also presents the main components of the GP algorithm, such as representation, evaluation, selection, genetic operators, and pseudocode of a basic GP algorithm. Afterward, the related work to the automatic generation of dispatching rules using the GP algorithm under different scheduling environments is covered. In addition, the limitations of the current literature are discussed, including premature convergence, GP bloating effect, feature selection methods, and high computational time.

2.1 Introduction

Manufacturing facilities have a significant impact on the national economies of countries, including gross domestic product, meaningful return on investment, employment rate, the link between manufacturing and innovation, and national security (Wang, 2018). Also, decision-makers seek to quickly adjust production systems to adapt to fluctuations in market demand, individual customization, a globalized market, and environmental pressures. Therefore, production planning is a necessity for manufacturing facilities to make the production process as efficient as possible

according to customers and organizational needs. Production planning can be defined as an administrative procedure that takes place within a manufacturing company to ensure that enough raw materials, workers, and other resources are available to generate final products according to the predetermined schedule (Kiran, 2019). Production planning activities are categorized into three major planning levels based on the length of the planning horizon as follows (Maravelias and Sung, 2009; Stadtler et al., 2015).

1) **Long-term planning.** Strategic decisions are made at this level which lay the foundation for the future development of the manufacturing organization over several years. Typically, decisions included in long-term planning are plant location and layout, product development, process development, equipment planning, material handling, employee welfare, etc.
2) **Mid-term planning.** It establishes a framework for routine operations, including basic quantities and timings for the flows and resources in a specific manufacturing system. The planning horizon extends from 6 to 24 months. Material requirement planning, production planning, and distribution planning are some examples of the tactical decisions involved in mid-term planning.
3) **Short-term planning:** All actions must be specified as comprehensive tasks for immediate execution and control. As a result, short-term planning models need the greatest level of precision as they have a great influence on the actual performance. The planning horizon might range from a few days to three months. Operational tasks include material control, quality control, machine loading, **production scheduling**, transport planning, etc.

Over the last decades, short-term production scheduling has been studied from various perspectives to develop efficient methods for a variety of manufacturing environments to deliver on-demand products to customers in a cost-effective manner (Dolgui et al., 2019). Production scheduling is a complex decision-making procedure for assigning tasks to a limited number of production resources with the goal of optimizing one or multiple objectives under constraints related to processes, resources, and system settings. In 1954, Johnson proposed a two-machine flow shop scheduling model with the objective of minimizing makespan (Johnson, 1954). This model is considered the first work in the scheduling literature. Since then, multiple scheduling approaches have been suggested for different manufacturing paradigms including centralized scheduling, distributed scheduling (Toptal and Sabuncuoglu, 2010),

decentralized scheduling (Minguillon and Lanza, 2019), and cloud manufacturing scheduling (Liu et al., 2019).

Job shop scheduling problems typically belong to centralized scheduling environments. As shown in Figure 2.1, centralized scheduling approaches can be classified based on five major factors (Jiang et al., 2021). They are the production environment, processing characteristics of operations, resource constraints, objective function, and system configuration. Regarding resource constraints, one-piece refers to a manufacturing system where products can flow one by one through each step of the process. Batch, on the other hand, refers to manufacturing methods in which products are moved from one phase to the next in groups (batches). The scheduling problems addressed in this thesis include one-piece resource constraints. In addition, the other factors are explained in more detail in the upcoming sections.

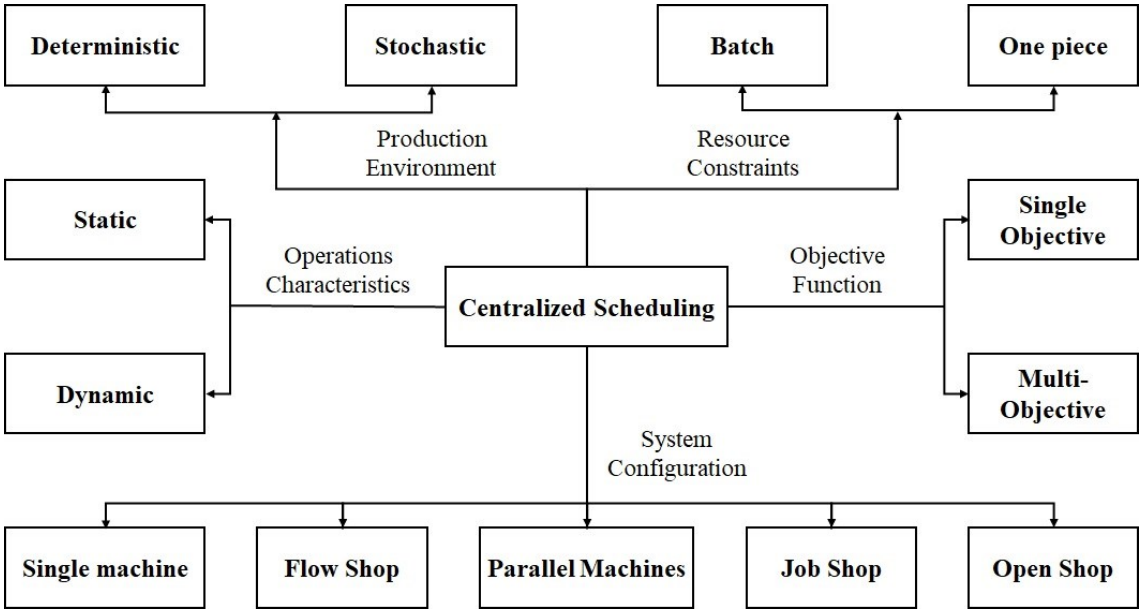


Figure 2.1: Classification of centralised scheduling

2.2 Job Shop Scheduling

2.2.1 Problem Description

The notations used to describe general JSSP instances are as follows.

- There are a number of M machines on the shop floor that need to process N jobs.
- Each job j that arrives on the job shop floor has a set of N_j operations that have to be processed in the same sequence to complete the job.
- The due date assigned to job j is indicated as d_j.

- Each operation o_{ij} for a job j, which is the i^{th} operation of job j, has a processing time $p(o_{ij})$, which determines how long the operation needs to be processed by a particular machine $m(o_{ij})$.
- The time when an operation o_{ij} of a job j is ready to be processed on a given machine is referred to as operation ready time $r(o_{ij})$, which is the completion time of its preceding operation $(o_{i-1,j})$.
- The release time of job j denoted as r_j is equal to the ready time of its first operation $r(o_{1j})$.
- The completion time of job j, when all operations finish processing, is referred to as C_j.
- The weight of job j denoted as w_j represents the importance of the job in goals related to customer satisfaction such as weighted tardiness objective.

The model parameters that remain constant from the moment a job j arrives on the shop floor are r_j, N_j, $p(o_{ij})$, $m(o_{ij})$, d_j, and w_j. In contrast, model variables that change depending on scheduling decisions made during the production process are $r(o_{ij})$, and C_j. In order to obtain valid schedules for JSSPs, there are many constraints that have to be taken into account as follows.

- Operation $o_{i-1,j}$ must finish processing before operation o_{ij} can start processing.
- No job can be processed on more than one machine at any time.
- Only one job can be processed by a machine at a time.
- No job can begin processing before its arrival in the shop.
- A job cannot be processed before it arrives on the shop floor.
- Job routing has to be maintained during the planning horizon and no alternative routes are allowed.

In the absence of precedence relationships between operations, that is, operations can be processed by machines in an arbitrary order, this problem is known in the scheduling literature as an "*open shop*" scheduling problem. In contrast, if all jobs follow exactly the same machinery sequence, then the problem is called a "*flow shop*" scheduling problem. In addition, in a parallel machine environment or "*flexible job shop*", there is a collection of work centres where there are a number of machines in each centre that can process a given operation.

There are two main classes of job shop scheduling problems, *static* and *dynamic* problems. In *static* scheduling problems, the set of jobs on the shop floor does not change over time. In contrast, new jobs can arrive on the shop floor at any time point during the production process in *dynamic* scheduling problems. Another key factor to consider in JSSPs is the uncertainty of the processing data, such as release times, processing times, due dates, etc. If all the processing information is known in advance before the production begins, then it is a *deterministic* scheduling problem, whereas information uncertainty is inherent in *stochastic* scheduling problems. Specifically, when a job arrives on the shop floor in either a static or dynamic pattern, it joins the queue of the first machine, in its route, to process the first operation with a deterministic or stochastic processing time. When the first operation is completed, the job moves to the next machine in its operation sequence. After the last operation in a given job is processed, the job exits the shop floor, and its completion time is updated. When all the required number of jobs are processed, a schedule is obtained that determines the start and finish time of each operation for all jobs. In order to assess the quality of the obtained schedule, several objective functions have been developed and frequently used in previous studies (Nguyen et al., 2015a; Shady et al., 2021b; Tay and Ho, 2008) as shown in Table 2.1, where f_j denotes the flow time of a job j. Set $\mathbb{T}$ represents the collection of tardy jobs; and $\mathbb{C}$ stands for the set of completed jobs.

Table 2.1 Objective functions for job shop scheduling problems

Objective function	Equation
Makespan	$C_{max} = max_{j\in\mathbb{C}}\{C_j\}$
Mean Flowtime	$F = \frac{\sum_{j\in\mathbb{C}} f_j}{\lvert\mathbb{C}\rvert}$
Maximum Flowtime	$F_{max} = max_{j\in\mathbb{C}}\{f_j\}$
Mean Tardiness	$T = \frac{\sum_{j\in\mathbb{T}}(C_j - d_j)}{\lvert\mathbb{T}\rvert}, \quad \mathbb{T} = \{j \in \mathbb{C}: C_j - d_j > 0\}$
Max Tardiness	$T_{max} = max_{j\in\mathbb{T}}\{C_j - d_j\}$
Percentage of Tardy Jobs	$\%T = 100 \times \frac{\lvert\mathbb{T}\rvert}{\lvert\mathbb{C}\rvert}$
Total Weighted Tardiness	$TWT = \sum_{j\in\mathbb{T}} w_j (C_j - d_j)$

2.2.2 Classes of Schedules

Because there are an infinite number of ways to insert idle time in a given schedule, there are an infinite number of potential schedules that can be generated for any job shop scheduling problem. In other words, when a machine becomes idle at a certain time (after the current process is finished), the scheduling algorithm must decide which job to process. Therefore, there are three basic classes of schedules as follows (Pinedo, 2012).

1) **Active schedule.** It is a feasible schedule where it is not possible to create another schedule where at least one operation completes earlier than the original schedule while having at least one operation that finishes later than the original schedule. In other words, an active schedule cannot be modified to make one process finish early without making another process finish later.
2) **Semi-active schedule.** It is not possible to make an operation finish processing earlier without changing the order of processing operations on any of the machines on the shop floor.
3) **Non-delay schedule.** It is a feasible schedule where no machine is kept idle while there is a job waiting in the machine queue. That is, a machine is allowed to be idle if there are no waiting jobs, otherwise, the machine starts processing one of the waiting jobs based on the scheduling algorithm immediately upon completion of the current job.

Giffler and Thompson have proven that the optimal solution for job shop scheduling in the case of minimizing the makespan objective must be an active schedule (Giffler and Thompson, 1960). Figure 2.2 presents a generalized procedure to create active, non-delay, or hybrid (active and non-delay) schedules for job shop scheduling problems with a predefined scheduling rule (Nguyen et al., 2017a). This algorithm has been commonly used in the scheduling literature to deal with a variety of production scheduling problems. The algorithm begins by defining a set of unscheduled operations (Ω) that are ready to be processed. Then, the algorithm determines the operation (σ^*) with the earliest completion time $S(m^*)$ and its corresponding machine (m^*). Accordingly, a set of all operations with a ready time smaller than $S(m^*) + \alpha(t(\Omega) - S(m^*))$ is created. The α parameter is a non-delay factor $\alpha \in [0, 1]$ to control the look-ahead ability of the procedure i.e., it restricts operations included in the set Ω'. In other words, if $\alpha = 0$, only non-delay schedules can be created (operations that are waiting at the queue of machine m^*), whereas if $\alpha = 1$, the algorithm creates an active schedule by considering

all operations ready to be processed (join the machine's queue) before the earliest completion time of m^*. On the other hand, if the non-delay factor has a value between 0 and 1, then hybrid schedules are created of both active and non-delay schedules.

1: Initialize a set of unscheduled operations $\Omega \leftarrow \{o_{1,1}, o_{2,1}, \dots, o_{N,1}\}$
2: Let $\mathrm{t}(\Omega) = min_{\sigma \in \Omega}\{\max\{r(\sigma), U_{m(\sigma)}\} + p(\sigma)\}$
3: Let σ^* be the operation that minimum is achieved, $m^* = m(\sigma^*)$, and $\Omega^* = \{\sigma \in \Omega \mid m(\sigma) = m^*\}$
4: Let $\mathrm{S}(m^*) = \max\{min_{\sigma \in \Omega^*}\{r(\sigma)\}, U_{m^*}\}$
5: Let $\Omega' = \{\sigma \in \Omega^* \mid r(\sigma) \leq \mathrm{S}(m^*) + \alpha(t(\Omega) - S(m^*))\}$
6: Apply dispatching rule on Ω' to find the next operation σ' to be scheduled on m^*
7: Remove σ' from Ω' and add its successor operation into Ω (if available)
8: If Ω is not empty, return to step 2.

Figure 2.2: Generalized schedule construction algorithm

According to (Ouelhadj and Petrovic, 2009), the real-time events that can occur in dynamic scheduling problems can be classified into two major categories as follows.

1) **Resource-related events:** Including unforeseen events that will influence the performance of production resources, such as machine breakdowns, maintenance, changes in job setup times, defective materials, delays in arrival, shortages of materials, etc.
2) **Job-related events:** Including dynamic events affecting jobs, whether these events occur due to a change in jobs' properties, for example, processing times, number of operations, due dates, etc., or a change in the number of jobs expected to arrive on the shop floor during processing, for example, the arrival of urgent jobs, cancellation of jobs, early or late arrival of jobs, etc.

This thesis focuses on the job-related dynamic events, including dynamic arrival of jobs, stochastic processing times, the number of operations at each job, and jobs' due dates are stochastic.

2.2.3 Solution Approaches for JSSPs

This section gives an overview of solution approaches that have been proposed in the scheduling literature based on the job shop environment, including static and dynamic scheduling problems. As shown in Figure 2.3, there are five major solution approaches in the scheduling literature. These approaches are exact methods (mathematical programming methods), approximate methods including heuristics and meta-heuristic algorithms, simulation methods, and artificial intelligence methods (Jiang et al., 2021). Exact methods such as mixed-integer programming, lagrangian relaxation, branch and bound, and dynamic programming are the most traditional optimization methods proposed in the scheduling literature in order to obtain optimal schedules for static problems. Typically, these methods rely on excessive assumptions to reduce the complexity (simplification) of the underlying scheduling problem. Also, they are computationally impractical for large problem instances because most scheduling problems are NP-hard (Pinedo, 2012).

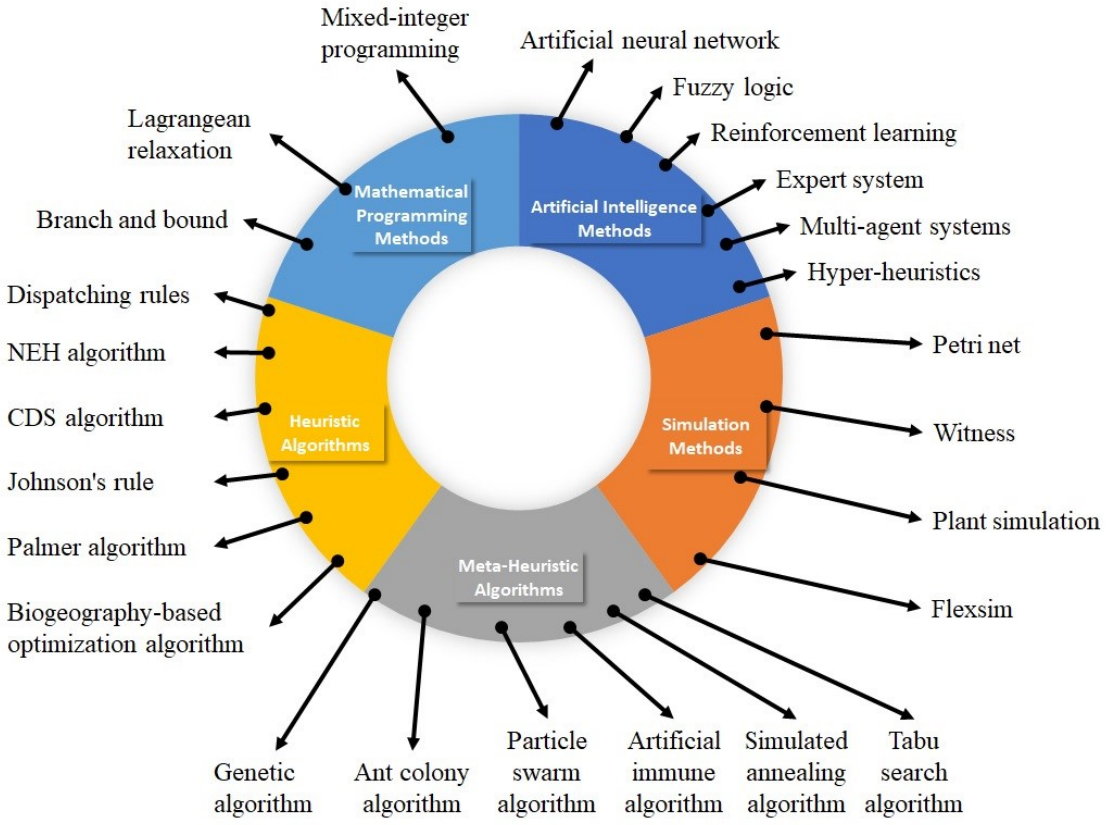


Figure 2.3: Solution approaches for scheduling problems

Approximate approaches have been developed to overcome the computational burden of exact methods i.e., approximate methods aimed at obtaining good enough solutions in an acceptable computational budget. Heuristic and meta-heuristic

algorithms are the main categories of approximate methods, although their nature is different (Gao et al., 2020). Heuristic algorithms such as dispatching rules (Sels et al., 2012a), NEH algorithm (Liu et al., 2012), Johnson's rule (Johnson, 1954), etc. are problem-dependent methods developed to solve specific problems and usually include domain knowledge in their searching mechanism. Therefore, Heuristic algorithms are not applicable to other problem domains without significant modification. In contrast, meta-heuristic algorithms, e.g., genetic algorithm, tabu search algorithm, simulated annealing algorithm, etc., are problem-independent methods that can be used for a wide variety of problems without the need for significant modification (Onar et al., 2016).

Simulation models have been widely used in dynamic scheduling to imitate stochastic variables that occur in real-world problems because it is very challenging to model them using analytical approaches (Ramasesh, 1990). Typically, simulation models are combined with other optimization techniques either exact or approximate to generate solutions that are evaluated using simulation models. Several simulations software have been proposed with user-friendly interfaces to facilitate the modeling of manufacturing systems under different production conditions such as Witness (Waller, 2012), Plant simulation (Bangsow, 2020), and Flexsim (Nordgren, 2002).

The use of artificial intelligence methods in the field of production planning and scheduling has been very active since the 1960s (Çaliş and Bulkan, 2015). Since then, artificial intelligence techniques have become powerful solution methods for many combinatorial optimization problems including job shop scheduling. Frequently used artificial intelligence methods include artificial neural networks (Weckman et al., 2008), fuzzy logic (Bilkay et al., 2004), expert systems, multi-agent systems (Kouider and Bouzouia, 2012), and hyper-heuristics (Burke et al., 2013). Most of these methods are specially developed to solve dynamic scheduling problems with random job arrivals and machine breakdowns as reported in (Çaliş and Bulkan, 2015; Mohan et al., 2019).

1) Approximate Methods for Scheduling Problems

This section provides more details on the most common *heuristics* and *meta-heuristic* algorithms used to solve both static and dynamic JSSPs. Regarding *heuristic* methods, Johnson's rule is used to minimize the completion time (makespan objective function) for a set of jobs that has to be processed on a two-machine flow shop (Johnson, 1954). Johnson's rule can achieve the optimal solution if some conditions are met including job processing times are known and constant, job priorities are not taken

into account, and all jobs must follow the same sequence of the two machines. In a follow-up study (Jackson, 1956), Jackson's algorithm uses Johnson's rule to minimize the makespan objective function in the static two-machine scheduling problem. This algorithm can also obtain an optimal solution in polynomial time $O(n \times log(n))$ if the same conditions are satisfied. Afterward, Palmer's algorithm was introduced in (Palmer, 1965) for static scheduling problems where there are more than two machines, multiple jobs, and Jonson's conditions are not met. Campbell Dudek Smith (CDS) algorithm (Campbell et al., 1970) uses Johnson's algorithm at each iteration to obtain optimal or near-optimal schedules in terms of minimum completion time for multiple jobs, and multiple machine scheduling problems. Also, Nawaz Enscore Ham (NEH) Algorithm has been proposed in (Nawaz et al., 1983) and is known as the insertion algorithm to reduce makespan objective for multiple jobs, and multiple machines flow shops. If the number of machines on the shop floor greatly exceeds the number of jobs, then the CDS algorithm is expected to outperform the NEH algorithm because the effectiveness of the former depends on the number of machines and the latter depends on the number of jobs. The aforementioned algorithms follow a set of rules for determining the sequence of jobs on the available machines i.e., they prioritize the set of jobs waiting to be processed on a machine based on specific characteristics. This idea has influenced researchers to design scheduling heuristics known as "*dispatching rules*" to create schedules for different job shop settings and objective functions. For many decades, dispatching rules have been extensively studied in the scheduling literature leading to a large number of dispatching rules developed for both static and dynamic problems. Although dispatching rules is one of the approximate methods initially proposed for static problems, they are more commonly used in dynamic environments than other solution approaches as reported in (Dominic et al., 2004). Since Chapters 4, 5, and 6 focus primarily on dynamic settings, dispatching rules are converted in more detail in the next section.

In the past two decades, research on the development of meta-heuristic algorithms has been very active due to the high computational costs of exact methods and the inability of heuristic methods to adapt to environments other than for which they were developed (Garey et al., 1976). Meta-heuristics are higher-level problem independent methods that guide the search process to find near-optimal solutions for optimization problems. Metaheuristic methods are classified into two categories, *local*, and *global* search-based algorithms as shown in Figure 2.4. Local search-based methods can find

the optimal solution for a specific area of the search space (local optima), or the global optima where there is no local optima or the area being searched contains it.

Local search methods start with a complete candidate solution (schedule) and iteratively make small changes (neighbourhood exploration) until there is an improvement in the quality of the solution, and then they takes it as a new solution (schedule) (Hussain et al., 2019). These methods include simulated annealing algorithm, tabu search, iterated local search, variable neighbourhood search, and Greedy Randomized Adaptive Search Procedure (GRASP). Teramoto et al. proposed a scheduling method based on a simulated annealing algorithm for JSSP with the aim of reducing the average flowtime (Teramoto et al., 2020). They developed two methods for limiting the neighbourhood of a solution in order to overcome the drawback of simulated annealing, where finding good solutions largely depends on the quality of the initial solution. The proposed methods obtained higher probabilities of finding effective solutions compared with the standard SA algorithm due to its ability to avoid updating a solution in the wrong direction. The two identical parallel-machine scheduling problems found in many real-world industries are investigated in (Xu et al., 2019). The authors used Iterated Local Search (ILS) and a Tabu search method to find a near-optimal solution to the problem, with the goal of optimizing the maximum total completion time for each machine, that is, increasing the level of machine utilization and decreasing the overall waiting time for jobs. The computational results on random instances showed that the proposed Tabu search algorithm outperforms two existing (SPT and RSPT) algorithms and the ILS method in instances with a small-to-medium number of jobs. In contrast, the ILS method performs better than the two SPT and RSPT algorithms as well as the Tabu search algorithm in instances with a large number of jobs. Zandieh and Adibi introduced a scheduling approach based on a variable neighbourhood search for dynamic job shop scheduling problems that take into account random job arrivals and machine failures (Zandieh and Adibi, 2010). An artificial neural network was also used to update the parameters of the variable neighbourhood search at any rescheduling point according to the problem state. The proposed method was compared with the shortest processing time, first in first out, and last in first out dispatching rules commonly used in the dynamic scheduling literature to optimize the mean flowtime objective function. The results demonstrated the efficiency of the proposed method in a variety of job shop conditions.

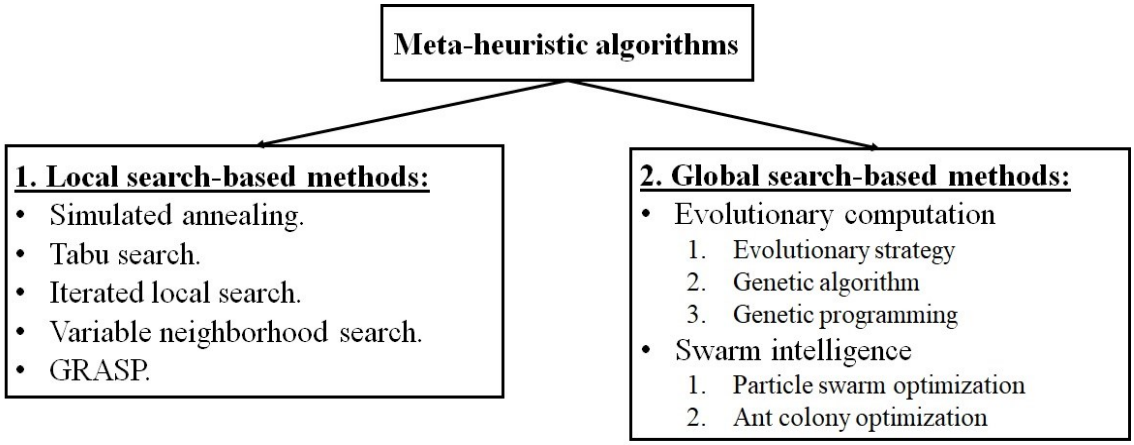


Figure 2.4: Meta-heuristic algorithms for scheduling problem

Global search methods are typically used on relatively complex problems when little information is known about the objective function response surface structure, or when there are many local optima in the function (Hussain et al., 2019). Global search meta-heuristics are categorized into two main categories, *evolutionary computation*, and *swarm intelligence*. Evolutionary strategies, genetic algorithm, GP are the most common evolutionary computational methods used in the scheduling literature. Horng et al. proposed an evolutionary algorithm by embedding an evolutionary strategy in ordinal optimization to get an acceptable schedule for stochastic job shop scheduling problems with the objective of minimizing the expected sum of storage expenses and tardiness penalties (Horng et al., 2012). Experimental results from comparing the proposed approach with five dispatching rules demonstrated the efficiency of the approach in achieving sufficiently good schedules in terms of solution quality and computational efficiency. Zhou et al. developed a hybrid heuristic genetic algorithm approach to improve the efficiency of the traditional genetic algorithm in reducing the maximum completion time of job shop scheduling problems (Zhou et al., 2001). The proposed approach integrated SPT and MWKR scheduling rules and neighbourhood search technique into the genetic evolution process to improve the solution performance. The achieved results demonstrated the superiority of the method over literature methods including guided biology search, simulated annealing, and traditional genetic algorithm. Since the focus of this thesis is the automatic generation of dispatching rules using GP methods, the following sections are devoted to a detailed explanation of dispatching rules, the GP mechanism, related studies, and current challenges.

Swarm intelligence is a branch of evolutionary computing that makes use of the collective behaviour of decentralized and self-organized systems represented in a swarm or flow of organisms (Kennedy, 2006). Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) are the two most prominent methods of swarm intelligence. Wang et. al proposed an improved particle swarm optimization algorithm for DJSSPs with random job arrivals (Wang et al., 2019). The improvement strategies include a modified decoding scheme, a population initialization approach, and a novel particle movement method. They used a mixed integer programming model to generate an initial schedule while minimizing three objective functions. When a new job arrives on the job floor, the proposed PSO algorithm reschedules the new jobs to maintain the performance and stability of the job shop schedule. Results showed that the improved PSO algorithm has high performance compared with five variants of the standard PSO algorithm and three meta-heuristics from the literature. Korytkowski et. al. developed an approach by integrating a simulation model of a dynamic job shop with a heuristic based on ant colony optimization to minimize four objective functions including mean flow time, max flow time, mean tardiness, and max tardiness (Korytkowski et al., 2013). The discrete event simulation model was used to evaluate the local fitness function of ants, whereas the ACO algorithm was used to find a good assignment of multi-attribute dispatching rules for each machine rather than using a single dispatching rule for all machines. The results obtained using a case study demonstrated the ability of the proposed approach to obtain better results than the case where only one dispatching rule was used in an entire system.

2) Frequently Used Techniques for Dynamic Scheduling

According to the literature review of dynamic scheduling in manufacturing systems proposed in (Ouelhadj and Petrovic, 2009), the solution approaches used for dynamic scheduling problems can be classified into three major groups which are discussed in detail below.

1) Completely reactive scheduling

No predetermined schedule is generated, and scheduling decisions are carried out in real-time during processing. In other words, scheduling decisions are made as soon as a specific machine becomes idle (Pinedo, 2012). Due to the short reaction time (prompt response) required in completely reactive scheduling, dispatching rules are the most widely used approach in practice. Dispatching rules, or scheduling policies,

are used to select the next job to be processed (highest priority) from the set of awaiting jobs when a machine becomes free. A large number of dispatching rules have been proposed in the literature that can be classified on the basis of structure into four categories (Jones et al., 1998) as shown in Figure 2.5.

- Simple priority rules: It includes rules which are primarily developed using job-related information. Examples of simple rules are those based on processing times (SPT: shortest processing time), due dates (EDD: earliest due date), arrival times (FIFO: first in first out), etc.
- Combinations of rules: they are used depending on the current situation on the shop floor. An example of a combination of rules is the case where the SPT rule is used until the number of jobs in a machine's queue exceeds 5, and then the FIFO rule is used. This prevents jobs with large processing times from being queued for long periods.
- Weighted priority indices: These rules usually have more than one piece of information (attribute) in a linear combination for creating schedules. Each attribute included has a weight (coefficient) that reflects its relative importance. A common example of this kind of rule is as follows: $rule = 2 \times processing\ time + work\ in\ next\ queue + next\ processing\ time.$
- Composite dispatching rules: Scheduling information is combined into more sophisticated structures rather than a linear combination used in weighted priority indices. GP methods are usually used to automatically generate this kind of dispatching rule as they usually show robust and high performance under different scheduling problems (Nguyen et al., 2017a).

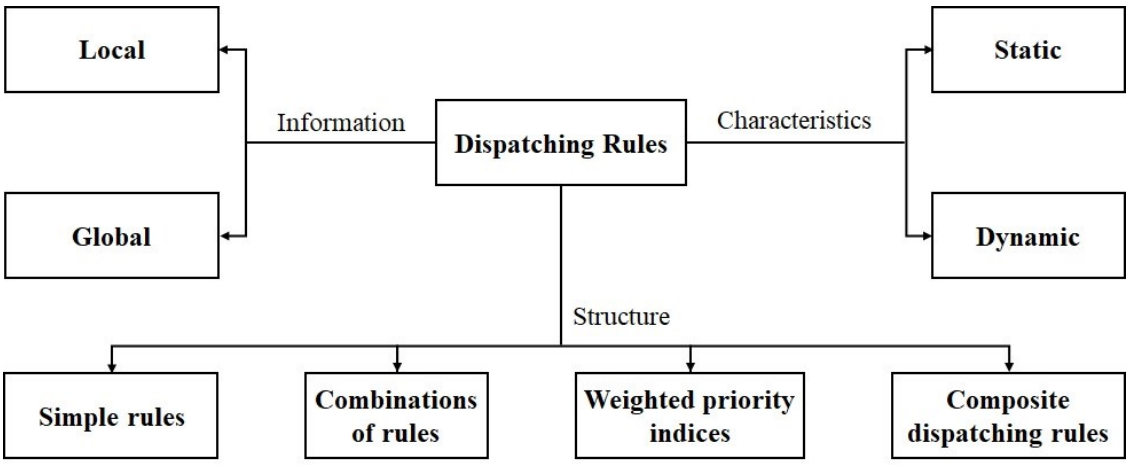


Figure 2.5: Classification of dispatching rules

Dispatching rules can also be classified according to their characteristics as *static* or *dynamic*. Static rules do not depend on the current time on the shop floor, e.g., SPT, FIFO, EDD, etc. In contrast, dynamic rules are time-dependent, which means that their performance is affected based on the current time, for example, minimum slack, critical ratio, cost over time, etc. Another classification is given in (Pinedo, 2012) where dispatching rules are categorized based on the information, used to construct them, into *local* and *global* rules. Local rules use the processing information available at the machine to choose the next job for processing, such as SPT, FIFO, etc. On the other hand, global rules use processing information from other machines, such as work in the next queue, apparent tardiness cost, etc. Several comparative studies have been conducted in the literature to evaluate the performance of existing dispatching rules under different job shop settings and objective measures (Holthaus and Rajendran, 2000; Rajendran and Holthaus, 1999; Sels et al., 2012a). Three major limitations of human-made dispatching rules found in the literature are frequently reported as described below.

- The performance of dispatching rules is highly dependent on the scheduling problem (system settings and processing characteristics) and the objective function to be optimized.
- The more information about the processing and the considered objective function is included in the dispatch rules, the higher their performance under different system settings.
- Although composite dispatching rules have better performance compared with other types of scheduling rules, the manual design of these rules is a very challenging task.

2) Predictive-reactive scheduling

An initial schedule is generated before processing begins using the available information. Then, when unexpected events occur during the actual processing, the generated schedule is modified in response to the real-time events. Most of the predictive–reactive scheduling approaches in the literature rely on simple schedule modification, known as "*rescheduling*", considering only job shop efficiency. The goal of these approaches is to minimize deviation from the initial schedule because significant changes might seriously affect other planning activities, resulting in poor schedule performance. Several predictive–reactive scheduling methods have been

proposed in the literature, for example (Adibi et al., 2010; Duenas and Petrovic, 2008; Yang and Geunes, 2008).

3) Robust pro-active scheduling

Robust pro-active scheduling approaches attempt to predict unforeseen events that might occur during processing, thus generating predictive schedules (robust schedules) that can reduce the negative impact caused by these dynamic events (Al-Hinai and ElMekkawy, 2011). This is accomplished by allocating idle time between jobs processing so that jobs are less likely to be disrupted by new incoming jobs or machine failures (Mehta and Uzsoy, 1998). A limited number of research articles have been published on the generation of robust schedules because predictability metrics are difficult to define and there is no universal definition of schedule robustness (Ouelhadj and Petrovic, 2009).

2.3 Hyper-heuristics

Despite the fact that heuristic and meta-heuristic approaches have been successfully applied in solving real-world computational search problems, there are still certain challenges in applying them to new problems or even new instances of the same problems (Burke et al., 2013). The reasons behind this are as follows:

a) There is a large range of parameters or algorithm options involved when using these methods.
b) Developing and maintaining problem-specific methods (heuristics) is a costly process in terms of time and effort.
c) There are no clear guidelines on how to choose them based on the problem understudy and why different heuristics work efficiently, or not, in different problem instances.

Therefore, many scholars have suggested using artificial intelligence techniques to develop algorithms that are more applicable than many existing research approaches in the literature, which are known as "*hyper-heuristics*". In other words, the goal of using hyper-heuristics is to raise the level of generality with which search algorithms can operate by developing an easy-to-implement system that can operate on a wide range of related problems rather than a single narrow class of problem instances (Burke et al., 2003). A hyper-heuristic framework is a high-level methodology that, given a specific problem instance(s) and a number of low-level heuristics (or their components),

automatically generates an appropriate solution by appropriately combining the components available to solve the given problem(s). The term "hyper-heuristics" was first coined in a conference paper by Cowling et. al in 2001 (Cowling et al., 2000). The idea was further developed in (Cowling et al., 2002) and a genetic algorithm-based hyper-heuristic was applied for scheduling geographically distributed training staff and courses. Since then, numerous articles, tutorials, reviews, and books have been published regarding this emerging search paradigm. For instance, an introduction of hyper-heuristics concepts and a review of related articles are found in (Burke et al., 2003). In addition, Burke et al. proposed a classification of hyper-heuristic approaches based on their search mechanisms (Burke et al., 2013). A brief discussion of hyper-heuristic applications covering a wide range of scheduling and combinatorial optimisation problems was also presented. In a follow-up survey (Drake et al., 2020), the authors expanded the survey on hyper-heuristic approaches published in 2013 by considering recent advances in hyper-heuristic research frameworks, mainly selection-based methods, current research trends, and future research directions.

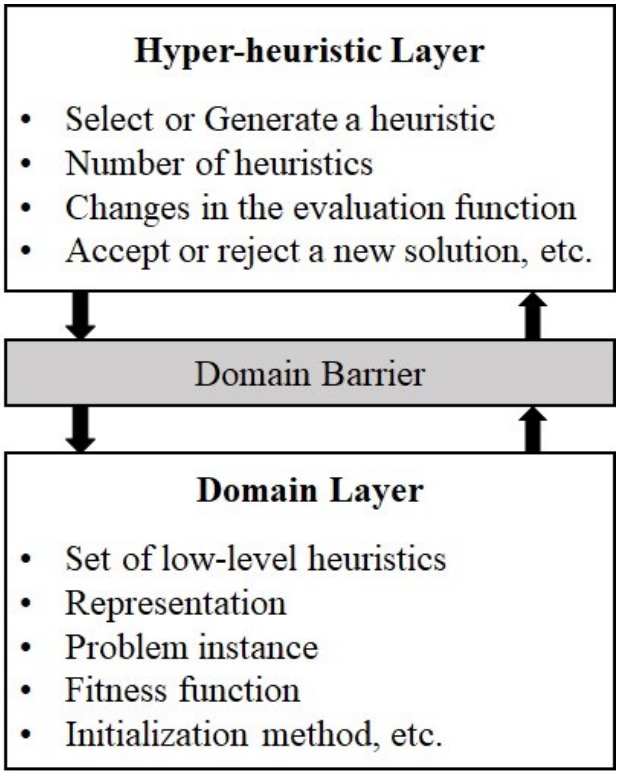


Figure 2.6: General hyper-heuristic framework

Figure 2.6 shows the general framework of the hyper-heuristic approach (Burke et al., 2011). The framework consists of two main layers, *hyper-heuristic* and *domain* layers, separated by *domain barrier* to prevent any knowledge transfer between the

layers. The hyper-heuristic layer is responsible for selecting or generating a number of heuristics from a set of low-level heuristics without having any prior knowledge of the domain in which they are used. The domain layer includes problem-related information such as a set of low-level heuristics, representation, problem instance, fitness function, etc. Finally, the hyper-heuristic method has to accept or reject a solution based on the results of the fitness assessment. Hyper-heuristic approaches can also be classified according to three main criteria, the source of feedback, the type of low-level heuristics, and the nature of the search space, as shown in Figure 2.7 (Burke et al., 2019). According to the source of feedback information, three techniques are available, including *online, offline,* and *no-learning*. In online hyper-heuristics, the learning process takes place while the hyper-heuristic algorithm is solving a given instance of a problem by sending instant feedback regarding performance, for instance, integrating reinforcement learning with heuristic selection methods, and the use of meta-heuristics to explore the search space of heuristics. In contrast, offline learning hyper-heuristics collect performance-related knowledge by evaluating the performance of generated heuristics over a set of training instances that generalize the nature of unseen instances (Swiercz, 2017). Also, there are two types of low-level heuristics existing in the literature including *constructive* and *perturbation* heuristics. Perturbative hyper-heuristics start with complete solutions randomly generated or by following some simple rules and then try to improve solution quality iteratively, whereas constructive hyper-heuristics start with partial candidate solutions and build up a solution gradually (iteratively) during the construction process. Finally, heuristic *selection* and *generation* are the two types of hyper-heuristic methodologies based on the nature of the heuristic search space. Heuristic *selection* approaches aim to select the right low-level heuristic for each problem instance, whereas heuristic *generation* methods seek to combine low-level heuristics to generate "new" heuristics for the problem under investigation.

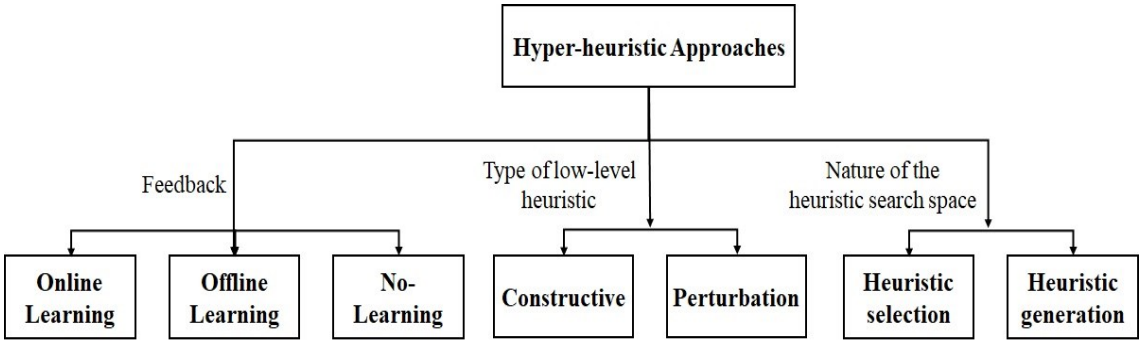


Figure 2.7: Classification of hyper-heuristic approaches

2.4 GP Based Hyper-heuristics

GP is an evolutionary computational method for global optimization inspired by the biological process of Darwinian evolution. In GP, computer programs are automatically generated in different shapes and sizes to solve problems without requiring the user to know or define in advance the shape or structure of the solution (J. R. Koza, 1994a). In other words, since the optimal solution and its shape are usually unknown in most real-world problems, the main advantage of using the GP approach is the variable length of generated solutions which allow a large number of solutions to be explored compared with fixed-length representations. The GP search mechanism can be briefly described in the following steps (O'Neill, 2009).

- A population of programs (solutions) is randomly generated using a specific representation and problem-related components.
- The performance of each program is assessed using a predefined fitness function.
- The fitness value of each program determines its likelihood of surviving and reproducing in the next generation.
- The average performance of generated programs evolves over generations until stopping criteria are met and the best program is returned.

The remainder of this section presents the key concepts of the GP approach with a detailed explanation of each step of the GP evolutionary process.

2.4.1 Representation

The syntax tree is the most popular GP representation (Nguyen et al., 2017a). However, other representations exist in the GP literature such as linear GP (Nie et al., 2013a), cartesian GP (Miller and Harding, 2008), and grammar-based GP (Hunt et al., 2016a). An example of a GP individual in a tree-based representation is shown in Figure 2.8. This tree structure represents the mathematical expression $x + y^2 + 3$. Here the arithmetic operators are $\{\times, +\}$ which is called the *function set*. Each element in the function set requires a finite number of arguments (leaf nodes) with a minimum of one argument, and thus cannot be placed at the leaves of a specific tree. The function set can include other types of functions, including mathematical functions, trigonometric functions, and logarithmic functions, depending on the nature of the problem (Branke et al., 2016a). In contrast, the elements $\{x, y, 3\}$ represent the contents of the *terminal set* and can be located only at the tree leaves (arguments). The variables x and y take

numerical values and represent the input of this expression. In addition, the terminal set might include problem-related features (program's external inputs) and constants that can be predefined or randomly generated. Therefore, the search space consists of all possible combinations created by selecting elements from the terminal (external nodes) and function sets (internal nodes). The set of the functions and terminals together is known as the *primitive set* of a GP. For the sake of convenience, GP individuals are usually expressed in a user-friendly format such as LISP S-expression (Nguyen et al., 2017a) because it is straightforward to see the relationship between sub-trees. For example, the tree expression displayed in Figure 2.8 can be presented using the prefix-notation expression as $(+ x (+ 3 (\times \ y y)))$.

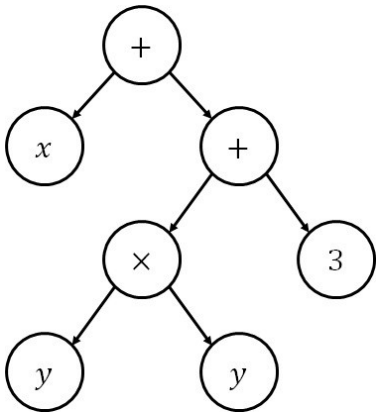


Figure 2.8: GP syntax tree representing the function: $x + y^2 + 3$

There are two requirements that terminal and function sets have to satisfy, namely *sufficiency* and *closure*. The sufficiency characteristic assumes that the contents of the terminal and function sets are sufficient to solve the problem under study. Because usually the optimal solution is not known in advance, it is difficult to ensure that included functions and terminals are enough to solve a given problem. However, the contents of the terminal and function sets are extracted from high-quality solutions that have been achieved in related studies. On the other hand, the closure property means that any function or terminal can be used as an input for any function in the functions set. Therefore, a protected division that returns zero (rather than undefined) in the case of dividing by zero is commonly used in previous studies (Mei et al., 2017a; Nguyen et al., 2018a; Shady et al., 2021b). In order to satisfy the sufficiency property, a large number of terminals have to be included, which increases the size of generated programs. Therefore, fixed length GP representation has been developed to resolve this issue. Also, other GP representations, such as strongly typed GP approaches and grammar-based GP approaches, have been developed to evolve syntactically and

semantically correct programs while satisfying the closure property. A brief description of each of these representations is given below.

- **Gene Expression Programming (GEP)**: has been proposed to overcome the bloating effect that commonly occurs in tree-based GP approaches by using a fixed linear structure in the evolution process. More details of the GEP approach regarding the initialization process, genetic operators, and learning mechanism are described in the next section and Chapter 6.
- **Strongly typed GP:** is an approach to enforcing type constraints where each terminal has a specific data type, and each function accepts arguments of certain types and will return its own type (Hunt et al., 2016a).
- **Grammar-based GP:** imposes some constraints on the structure or contents of generated individuals by using some rules known as "*grammar*" in the initialization process and genetic operators to ensure that the rules are logically correct (McKay et al., 2010).

2.4.2 Population Initialization

Similar to other evolutionary computation methods, the GP approach begins by randomly generating a population of programs. There are a number of different approaches to generating this random initial population. Three main initiation methods have been proposed in the literature for random population initialization. These approaches are *full*, *grow*, and *ramped half-and-half* (John R. Koza, 1994). The maximum depth is predefined for these methods in order to restrict generated individuals from going beyond it. The depth of a node is the number of edges that must be traversed to reach the root (the highest node in the tree with a depth of zero). Therefore, the depth of the tree is the distance from the furthest node (leaf) to the root of the tree. The full method generates full trees where all leaves have the same depth by randomly selecting elements from the function set until the maximum tree depth is achieved. Although the full method initializes programs with the same depth as the maximum depth, this does not mean that these programs have an identical number of nodes. Because the considered functions might require a different number of arguments known as *arity*. In contrast, the grow method generates individuals of various sizes and shapes because nodes are chosen from the primitive set (both function and terminal sets) until the maximum depth is reached. In other words, once the required maximum depth is achieved, only terminals can be chosen, similar to the full method. Consequently,

Koza proposed a combination of the two methods called ramped half-and-half because neither the full nor the grow method obtains a sufficient variety of sizes and shapes on its own (J. R. Koza, 1994a). Using the ramped half-and-half method, half of the initial population is initialized by the full method while the other half is initialized by the growth method. Figure 2.9 shows the pseudocode for a recursive implementation of both the full and grow methods; where $functions$: are the elements in the functions set, $terminals$: are the elements in the terminals set, max_depth: is the maximum allowed depth, method: is either full or grow method, expr: is the generated expression, and $rand()$: returns a random number uniformly distributed between 0 and 1 (O'Neill, 2009).

```
Procedure: generate_rand_expr(functions, terminals, max_depth, method)
1: if max_depth = 0 or (method = grow and rand() < |terminals| / (|terminals|+|functions|))
 2: then
 3:     expr = select_random_element(terminals)
 4: else
 5:     func = select_random_element(functions)
 6:     for i = 1 to arity(function) do
 7:         argument_i = generate_rand_expr(functions, terminals, max_depth – 1, method):
 8:     end for
 9: end if
10: return expr
```

Figure 2.9: Pseudocode for program generation algorithm.

2.4.3 Fitness Evaluation

Fitness assessment is a crucial step in any evolutionary computation algorithm because it represents the performance of generated individuals in a given task. Therefore, the higher the performance of a program in the current generation, the more likely it is to survive and reproduce, thus increasing its contribution to future generations. In other words, fitness functions guide the GP algorithm to promising search regions (high-quality programs) in the search space (Nguyen et al., 2017a). Fitness values are estimated using a fitness function, which can be either *static* or *dynamic* as shown below.

- **Static fitness function:** the fitness function does not change during the run. Therefore, evaluating the fitness value of a rule multiple times using the same static instance will always result in the same value.

- **Dynamic fitness function:** the function changes at any moment during the run. Therefore, the fitness value of a specific program might change depending on the time at which the program was evaluated.

Also, fitness functions might vary greatly depending on the problem domain. Therefore, it is common to evaluate the performance of generated programs across a range of different scenarios, i.e., by running the programs with different inputs. The set of scenarios used during the learning phase of GP is known as the *training set*, whereas the set of unseen scenarios used to evaluate the general performance of the generated programs is known as the *testing set*. The reason why different combinations of scenarios are used is that all possible inputs for a specific task are usually unknown, thus the objective of the learning phase is to generate programs that can perform well on unseen scenarios, i.e., their performance does not deteriorate on unseen scenarios (overfitting). Therefore, two sets of scenarios are used to verify this objective (F. Zhang et al., 2021d). In addition, fitness values can be used as termination criteria, such as predetermining the desired fitness level where the algorithm ends, or the achieved best fitness value does not change for a predetermined number of generations.

2.4.4 Selection

After the evaluation of all the individuals in a particular generation has been completed, some individuals will be selected to be the parents of the next generation. Several selection methods have been proposed in the literature, as shown below.

1) Tournament selection
2) Roulette wheel selection
3) Rank selection
4) Elitism selection

However, tournament selection and roulette wheel selection are the most popular selection methods in the GP literature (Nguyen et al., 2017a). Using the tournament method requires two main steps.

1) A number of programs equal to the tournament size are randomly chosen from the population to form the tournament.
2) The individuals with the best fitness values (lowest value in the case of the minimization problem) will be selected as parents.

The key merit of the wide use of the tournament selection method is the ease of adjusting the selection pressure by changing the tournament size. Specifically, a higher tournament size increases selection pressure, giving high-quality individuals a greater chance of being selected, whereas a smaller tournament size decreases selection pressure, giving a greater chance of poor individuals being selected. In addition, the tournament method can operate on a parallel architecture that reduces the computational time (Blickle, 2000).

The roulette wheel selection method is one of the methods known as *fitness proportionate selection*, where individuals are selected randomly based on a distribution proportional to their fitness values. In other words, individuals with higher fitness values will be selected more often compared with lower individuals. Equation 2.1 shows the probability p_i of selecting an individual i as a parent according to its fitness value f_i, where N represents the number of individuals in the current population (Blickle, 2000). The main limitation of the roulette wheel selection is that it might suffer from premature convergence caused by the loss of population diversity. Because the selection pressure might become very high if there are prominent individuals, and thus better individuals will be preferred over poor individuals. In contrast, if the difference between the fitness values of the individuals is very small, then the roulette wheel selection ability becomes similar to random selection (Yadav and Sohal, 2017).

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.1}$$

Using the rank selection method, individuals are weighted based on their rank in three steps, rather than their fitness values as used in the roulette wheel selection method. First, individuals in a given generation are sorted according to their objective values. Second, the fitness value assigned to each individual depends on their rank. Third, the probability of selecting each individual depends on its fitness value normalized by the total fitness values of the population (uniform selection scheme). Therefore, the rank selection method offers an advantage over the roulette wheel selection method by avoiding premature convergence by using more robust selection pressure. However, it is computationally expensive because all individuals must be sorted based on fitness value. Also, it might experience a slow convergence speed because high-performing individuals do not differentiate sufficiently from others (Kumar, 2012).

Finally, the elitism selection method copies a subset of the fittest individuals from the current generation into the population of the following generation. In other words, the elitism method reduces genetic drift during the evolution process by ensuring that only the best individuals are allowed to pass their traits to the next generation. Although the performance of the GP algorithm can be rapidly increased, the number of selected rules using the elitism method must be carefully determined to avoid losing population diversity and getting trapped in local optima (Du et al., 2018).

2.4.5 Genetic Operators

The rule of genetic operators is the formation of new individuals known as "*offspring*" for the next generation based on selected individuals from the previous generation (parents). Specifically, genetic operators help increase genotypic diversity within a population to efficiently explore different regions of the search space (O'Neill, 2009). Differences between individuals in real life occur for two main reasons: mutation of genes within the individual and recombination of genes through the process of reproduction. Similarly, two genetic operators have been proposed in the tree-based GP literature, following the same concepts that occur in nature (Nguyen et al., 2017a). They are *subtree mutation* and *crossover operators,* which are usually used with predefined probabilities to create the population of the next generation. (Poli et al., 2007)

1) **Subtree mutation operator:** it only requires one parent to generate a new program (offspring) following two steps. First, a random point, known as a *mutation point*, is randomly chosen in the selected parent. Second, a randomly generated subtree replaces the subtree rooted at this point (Poli et al., 2007). An example of the subtree mutation operator is shown in Figure 2.10, where the mutation point is highlighted by a dashed frame. Also, Figure 2.10 (a), Figure 2.10 (b), and Figure 2.10 (c) show the parent program, the randomly generated tree, and the new program generated after mutation, respectively.
2) **Subtree crossover operator:** Two programs (parents) are randomly selected from the current population. Then, a crossover point is randomly chosen for each parent. Two offspring are created by swapping the subtree rooted at the crossover point of the first parent with the subtree rooted at the crossover point of the second parent (Poli et al., 2007). Figure 2.11 shows an example of the subtree crossover operator applied to two parents to create two offspring; swapped subtrees are highlighted in dashed frames. It has been noted that many

crossovers might only swap two leaves (insignificant effect on diversity). Thus, Koza suggested randomly choosing function nodes 90% of the time and leaving nodes 10% of the time to increase population diversity (John R. Koza, 1994).

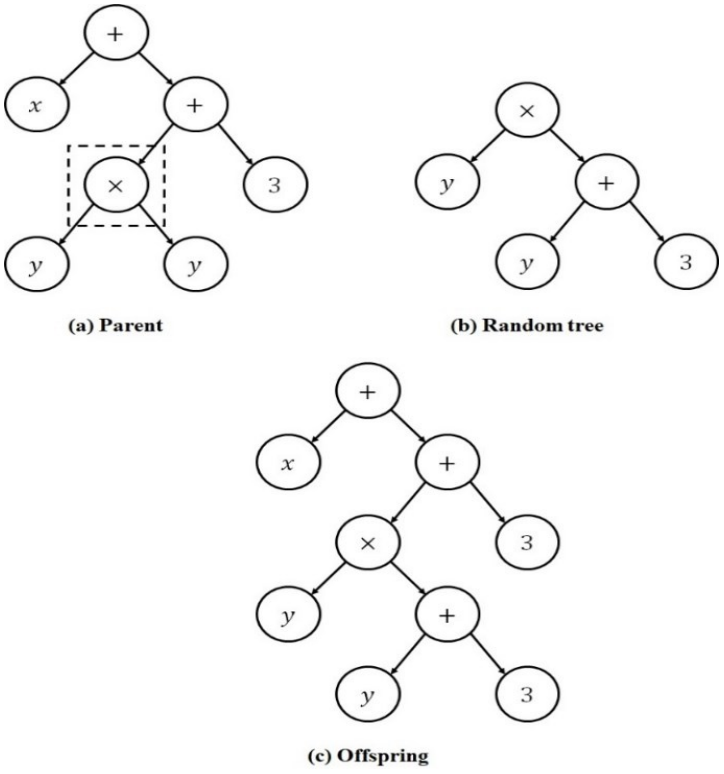


Figure 2.10: Subtree mutation in GP

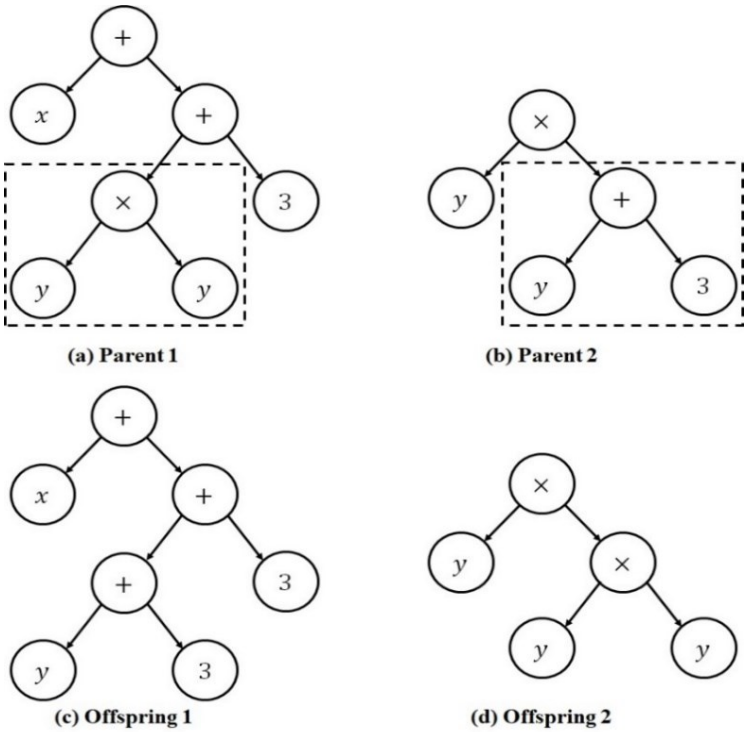


Figure 2.11: Subtree crossover in GP

2.4.6 Standard GP Algorithm

The pseudocode of the standard GP algorithm with a minimization objective function is shown in Figure 2.12. The algorithm returns the best program evolved, i.e., the program p^* that achieved the smallest fitness value $fitness(p^*)$ in the given problem. The algorithm begins by loading either static or dynamic problem data that is required for fitness assessment, function, and terminal sets based on the problem domain. This algorithm follows the four main steps discussed in the previous sections, including population initialization, fitness evaluation, selection, and genetic operators.

- First, a population of candidate programs is randomly created using the predefined function and terminal sets with a size equal to N and an initialization method that can be full, grow, or ramped half-and-half.
- At each generation in the evolutionary process, each program is evaluated using the predefined fitness function. If a specific program p_i has better performance (smaller fitness value in the minimization problems) than the current best program, it will replace the best program evolved so far and the best fitness value will also be updated accordingly, as shown in steps 5–11.
- Afterward, parent programs $P^{parents}$ are selected using one of the previously described selection methods to form the mating pool. Genetic operators are applied to generate new individuals for the next generation according to predefined probabilities until the desired population size is reached, as shown in steps 14–17.
- Finally, if the maximum number of generations $maxGenerations$ is reached, then the algorithm terminates and the best rule is returned; otherwise, another evolutionary iteration begins by following the same steps.

Inputs: problem data, function and terminal sets, and fitness function

Output: the best evolved rule p^*

1: $P \leftarrow \{p_1, p_2, \ldots, p_N\}$

2: **Set** $p^* \leftarrow null$ and $fitness(p^*) \leftarrow= \infty$

3: $generation \leftarrow 0$

4: **while** $generation \leq \max Generations$ **do**

5: **for all** $p_i \in P$ **do**

6: **evaluate** $fitness(p_i)$

7: **if** $fitness(p_i) < fitness(p^*)$ **then**

8: $p^* \leftarrow p_i$

9: $fitness(p^*) \leftarrow fitness(p_i)$

10: **end if**

11: **end for**

12: $p^{new} \leftarrow \{\}$

13: **while** $|p^{new}| < N$ **do**

14: $p_{offspring}$ ← apply genetic operators to selected individuals from P

15: $p^{new} \leftarrow p^{new} + p_{offspring}$

16: **end while** $|p^{new}| < N$ **do**

17: $P \leftarrow p^{new}$

18: $generation \leftarrow generation + 1$

19: **end while**

20: **return** p^*

Figure 2.12: Pseudocode for the standard GP algorithm

2.5 Generating Dispatching Rules Using GP

GP approach is the most popular hyper-heuristic approach in the field of automatic generation of scheduling rules. According to the hyper-heuristic classification given in Figure 2.7, using the GP approach to evolve dispatching rules is a hyper-heuristic *generation* method based on *offline* learning feedback to generate new low-level *constructive* heuristics. Several promising approaches to developing superior dispatching rules as compared with manually designed ones have been proposed in the literature for different job shop environments and manufacturing conditions (Branke et al., 2016a). This section provides a review of literature articles related to the use of the GP approach for scheduling problems, categorized based on the manufacturing

environment, as shown in Figure 2.13. Then, the challenges, their causes, and solution approaches in the current literature are discussed in detail in the second subsection, as illustrated in Figure 2.13, where GP limitations are highlighted in red, the reasons are highlighted in blue, and the solution approaches considered in this thesis are highlighted in green. Finally, a detailed explanation of how this thesis contributes to addressing each of these limitations is also provided.

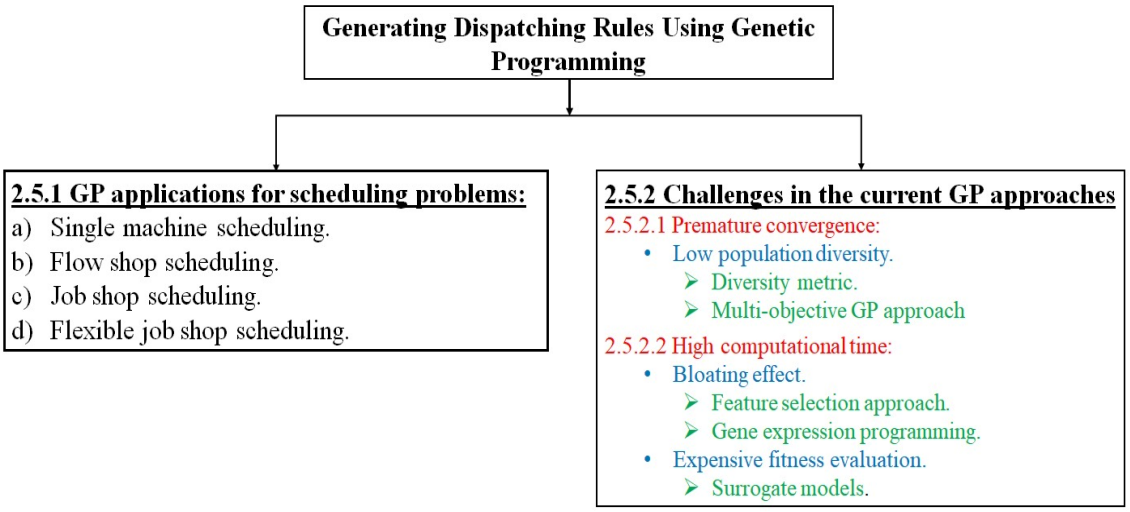


Figure 2.13: Structure of the 5th subsection of section 2.

2.5.1 GP Applications for Scheduling Problems

The GP approach has been used to generate dispatching rules for different manufacturing settings such as single machine, flow shops, job shops, and flexible job shop scheduling environments (Nguyen et al., 2017a), as described below.

a) Single machine scheduling

Yin et al. proposed a GP approach to generate predictive scheduling heuristics in a single-machine environment with stochastic breakdowns while considering two objectives, job tardiness, and stability (Yin et al., 2003). The proposed approach uses bi-tree GP representation to evolve two components: dispatching rules and a function to estimate the idle time that must be inserted before a given job can be processed. Empirical results showed that the proposed approach can efficiently generate high-quality dispatching rules and deal with uncertain perturbations due to the inserted idle time. In the same context, a single machine environment with a total weighted tardiness objective was investigated in (Dimopoulos and Zalzala, 2001). The authors proposed a GP approach to train

dispatching rules to handle different levels of job tardiness and due dates. The results showed that the nine dispatching rules evolved using the proposed approach had high performance in many unseen scenarios. Specifically, the automatically generated dispatching rules were at least as good as the current human-made rules in the literature across different validation instances and tardiness levels.

Geiger et al. proposed a GP approach to generate dispatching rules for a variety of single machine environments under three objective functions, including minimizing total completion time, minimizing maximum lateness, and minimizing total tardiness (Geiger et al., 2006a). The authors integrated the proposed approach with a simulation model (fitness evaluation) to develop a system for evolving dispatching rules called "scheduling rule discovery and parallel learning system". Dispatching rules generated using their system showed better performance compared with a large set of literature rules. Finally, the authors stated that although the man-made rules in the scheduling literature are the results of decades of research, the rules generated using the GP approach had better performance and could be created with a fraction of the time and effort. Therefore, the use of machine learning techniques in production scheduling is a promising research area with many opportunities to explore. Also, Jakobović and Budin developed an interesting GP approach for a dynamic single machine and job shop scheduling with bottleneck estimation capability (Jakobović and Budin, 2006). The proposed approach generates three program trees containing one discrimination function and two dispatching rules. The discrimination function is used to determine whether a given machine is a bottleneck. This function evolves using separate function and terminal sets. Accordingly, if the machine is a bottleneck, then the first dispatching rule is used, otherwise, the second rule is applied. Promising results were obtained by comparing the evolved scheduling rules with the literature ones. Finally, the authors reported that the proposed approach is particularly useful for scheduling environments in which appropriate heuristics are not available or to facilitate the design of high-quality heuristics.

b) Flow shop scheduling

Chen et al. proposed a GP approach to create heuristic rules for a k-stage hybrid flow shop problem where one stage consists of non-identical batch processing machines and the other contains non-identical single processing machines (Chen

et al., 2015). Then, the offline generated rules were selected by ant colony optimization algorithm for each sub-problem in the equipment manufacturing industry including part assignment, part sequencing, and batch formation. The proposed algorithm showed better performance compared with other hyper-heuristics in terms of reducing the total weighted tardiness. In the same vein, Vázquez-Rodríguez and Ochoa proposed a GP framework to generate variants of the Nawaz Enscore Ham heuristic for flow shop scheduling problems under five objectives i.e., one variant of NEH was generated for each objective (Vázquez-Rodríguez and Ochoa, 2011). The objective functions considered were: makespan, total tardiness, total weighted tardiness, the sum of completion time, and the sum of total weighted completion times. Experimental results showed that the proposed approach outperformed the original and randomized versions of the NEH heuristic across several benchmark problems. Shi et al. introduced a scatter programming algorithm to generate composite rules in a hybrid flow shop scheduling problem with the objective of reducing makespan (Shi et al., 2015). The proposed approach included a local improvement method to accelerate convergence. Simulation experiments demonstrated the effectiveness of the proposed approach compared with the standard GP and the scatter programming approaches in generating rules with higher scalability and flexibility.

c) Job shop scheduling

Since dispatching rules typically suffer from a lack of global perspective (do not consider the future state of the job shop when making scheduling decisions), Hunt et al. proposed a GP algorithm to evolve less-myopic dispatching rules for a dynamic ten machines job shop scheduling problem (Hunt et al., 2014a). The proposed approach evolved robust rules regarding the total weighted tardiness objective by incorporating features from the wider shop system. Regarding numerical experiments, it was observed that including global features in the terminal set improved the mean and reduced the standard deviation of the performance of the best-evolved rules. The same findings were also reported in (Shady et al., 2020b), where local and global features were included in the terminal set of the GP approach used in evolving dispatching rules for a dynamic job shop environment. The proposed approach had a lower mean flow time compared with ten man-made rules from the literature under different job shop utilization levels. In a follow-up study (Shady et al., 2020a), the authors

examined the GP approach with global features across a larger range of training and testing scenarios. Two objectives were considered including mean flow time and mean tardiness as well as 12 literature rules. The results showed the GP best-evolved rule had an average relative increase in performance of about 5.85% and 45.56% compared with the best literature rules with respect to mean flowtime and mean tardiness, respectively. In order to tackle the myopic nature of traditional dispatching rules, Nguyen et al. proposed a GP approach to generate a new type of dispatching rules called iterative dispatching rules (Nguyen et al., 2013b). Different from traditional dispatching rules that create one fixed schedule of jobs, these rules include look-ahead strategies to iteratively improve schedules by utilizing the information of scheduled jobs in previous steps. In other words, iterative dispatching rules can correct their behaviour based on their mistakes in previous sequencing decisions. The proposed approach had significantly better performance compared with the standard GP algorithm with makespan and total weighted tardiness as objective functions.

Hunt et al. developed a GP-based hyper-heuristic approach to create scheduling policies for a two-machine job shop scheduling problem in static and dynamic conditions (Hunt et al., 2014b). The static problem was considered to verify the ability of the GP approach to discover optimal scheduling rules. On the other hand, two representations (evolve a single rule for the two machines and evolve a specific rule for each machine) were used in the case of the dynamic problem with the aim of reducing the total weighted tardiness. The results showed that the GP algorithm was able to generate rules that could achieve optimal schedules similar to Jackson's algorithm. Also, the performance of the representations relied heavily on the testing instances (processing time and utilization levels). Since real-life scheduling problems usually encounter multiple conflicting objectives, Nguyen et al. developed four multi-objective GP approaches to automatically generate dispatching rules and Due-Date Assignment Rules (DDARs) in dynamic job shop scheduling problems (Nguyen et al., 2014a). Three multi-objective approaches from the literature were used namely Nondominated Sorting Genetic Algorithm II (NSGA-II), strength Pareto evolutionary algorithm 2, and harmonic distance-based multi-objective evolutionary algorithm. In addition, a new approach called diversified multi-objective cooperative evolution was used. The authors compared the rules

evolved using the four developed approaches with existing rules created from combinations of existing dispatching rules and DDARs with respect to makespan, normalized total weighted tardiness, and mean absolute percentage error. Experimental results showed that the evolved rules have dominated popular literature rules in both training and testing scenarios. Also, the obtained Pareto fronts helped in a better understanding of the trade-off between the objectives.

In the same context, Shady et. al proposed a GP for multi-objective DJJSP considering machine breakdowns (Shady et al., 2021b). The proposed approach combined the standard GP algorithm with NSGA-II to generate dispatching rules that can handle different breakdown situations while reducing job mean flow time and makespan. Regarding numerical experiments, the authors compared the evolved non-dominated rules to 12 literature rules under 10 testing scenarios generated by varying levels of machine breakdown. In addition, the structure of the best-evolved rule and the distributions of attributes across GP generations were analyzed to gain useful insight into the behavior of the evolved rules.

d) Flexible job shop scheduling

The Flexible Job Shop Scheduling Problem (FJSSP) is an extension of the JSSP where there are multiple copies of the most important machines (parallel machines) in order to reduce the waiting time caused by busy machines (bottlenecks) (Chaudhry and Khan, 2016). Therefore, two scheduling decisions have to be performed; finding a suitable machine to process a given operation (routing) and deciding the sequence of awaiting operations at each machine (sequencing). Tay and Ho proposed a GP approach to evolve dispatching rules for solving multi-objective FJJSPs where the weighted sum method was used to construct an objective function by combining makespan, mean tardiness, and mean flow time with the same weight (priority) (Tay and Ho, 2008). In addition, the least waiting time assignment rule was used to select a machine to process a given job. Five composite dispatching rules were evolved by the GP approach. The evolved rules showed higher performance compared with five popular rules from the literature. Hildebrandt et al. (Hildebrandt et al., 2010a) tested the rules generated in (Tay and Ho, 2008) under various objectives. The authors found that these rules performed poorly. In addition, two main reasons for this limited

performance were reported. First, a linear combination of objectives is not appropriate where the weight of each objective is unknown, which is usually the case. Second, the use of the fixed least waiting time assignment rule severely affects scheduling decisions facing sequencing rules.

Therefore, Yska et al. proposed the use of a cooperative co-evolution GP framework to simultaneously evolve both routing and sequencing rules for static and dynamic FJSSPs (Yska et al., 2018). Experimental results demonstrated the effectiveness of the proposed approach in reducing max flow time, mean flow time, and mean weighted flow time objectives compared with the standard GP approach that generated only dispatching rules for both static and dynamic scenarios. In the same context, Xu et al. developed a GP algorithm with delayed routing for solving multi-objective dynamic FJSSPs that optimizes energy efficiency and mean tardiness simultaneously (Xu et al., 2021). Three delayed routing strategies were presented to define the subset of ready jobs for the idle machine in order to make the sequencing decision. Therefore, every routing decision can be delayed which helps in taking the most recent information into consideration before assigning the ready jobs to machines.

2.5.2 Limitations and Corresponding Solution Methods in Related Studies

Although the GP approach has shown promising performance in generating dispatching rules automatically under different manufacturing settings and objective functions as described in the previous section, there are two general challenges in the existing GP approaches (Branke et al., 2016a; Nguyen et al., 2017a). The literature limitations are:

- Premature convergence in the case of static scheduling problems
- High computational time for both static and dynamic scheduling problems.

The following subsections discuss the reasons for these challenges, the advantages and disadvantages of the current solution methods in the literature, and a brief overview of the approaches proposed in this thesis.

1) Premature Convergence due to Low Population Diversity

Because JSSPs are NP-hard optimization problems, thus the fitness landscape might contain more than one optimum. Therefore, there is a higher chance that the GP individuals converge very early to one of these local optima (sub-optimal solutions), known as "*premature convergence*". Premature convergence is a well-known

phenomenon in evolutionary computation that occurs due to the significant loss of diversity among evolved individuals. Since high-quality individuals typically survive to future generations and their traits are shared between individuals with crossover operators at higher rates, thus a significant portion of individuals in future generations (offspring) will have somewhat similar genotypic structures (Ekárt and Németh, 2000). Therefore, it is necessary to maintain high levels of diversity among GP individuals to prevent premature convergence towards the structure of a small number of high-performance rules. Although a mutation genetic operator helps in increasing the genetic diversity between individuals to a certain extent, its effect is limited because of the following reasons (Hughes, 2021).

- The mutation rate, which determines how many individuals should be mutated, is usually much lower than the crossover rate to avoid over-exploration.
- There is no direct method to determine mutation rates because it depends on the problem under investigation, which requires many expensive GP runs.

Instead, distance metrics have been developed to define a straightforward technique to numerically measure the diversity between GP individuals, thereby controlling for desired diversity across generations. From a topological point of view, if two trees are close to each other or identical, one can be easily converted to the other with a few applications of genetic operators (Gustafson and Vanneschi, 2008). Accordingly, to enhance the population diversity, the similarity (distance) between individuals must be evaluated, and individuals with low similarity values must survive across generations.

a) Promoting population diversity

To enhance diversity among GP individuals, two major challenges must be considered:

- How to **measure diversity** between individuals?
- How to **maintain diversity** across evolving generations?

Regarding measuring the diversity between individuals, the distance between GP individuals is determined using *distance metrics* that measure similarity based on the individuals' structure (*genotype*) or performance (*phenotype*) (Burke et al., 2002). Therefore, several distance metrics have been proposed in the literature, such as the number of different structural individuals (genotypes) (Burks and Punch, 2015), fitness values (phenotypes) (Jackson, 2010), edit distances (de Jong et al., 2001), and composite measures (Kelly et al., 2019). In addition, Burke et al. examined the relationship between several measures of diversity in GP and fitness values (Burke et

al., 2004). Moreover, a survey and analysis of different semantic methods used to increase the phenotypic diversity between GP individuals were presented (Vanneschi et al., 2014). The distance metrics in these literature reviews were analysed using standard GP methods, such as artificial ants, even-5-parity, and symbolic regression. To the best of our knowledge, research that addressed the applicability of distance measures in measuring diversity among GP individuals in the JSS domain has not yet been reported. Also, the distance metrics in the GP literature have certain limitations that make them unsuitable for use in generating diverse rules for job shop scheduling problems, as shown below.

- Genotypic distance metrics consider the position of a node while neglecting the interaction effect between the node and its parents.
- Many distance measures assume that all nodes have the same weight, and this is not the case in scheduling rules where nodes near the root have more impact than nodes that are far away.
- The evolution of GP rules is computationally expensive, limiting the use of phenotypic distance measures.

Regarding maintaining diversity across generations, multiple diversity mechanisms are used to reduce the risk of high-performance individuals taking over the entire population before effectively exploring the fitness landscape (Sareni and Krahenbuhl, 1998). Diversity mechanisms can be classified into two groups: *niching* and *non-niching*. The most common niching methods are fitness sharing, clearing, and crowding. Non-niching techniques include methods such as the removal of genotype or phenotype duplicates, incest prevention, and island models (Hughes, 2021). The main intuition behind fitness-sharing methods is to devalue each individual's fitness at a rate that is proportional to the number of identical individuals in the population. Rather than penalizing fitness values, some scholars have suggested using multi-objective methods, where diversity is included as an explicit objective to be optimized along with fitness in the GP algorithm. A multi-objective method called age-fitness Pareto optimization was proposed in (Schmidt and Lipson, 2011) to promote population diversity by simultaneously optimizing the age and fitness of individuals. Similarly, Burks and Punch introduced a genetic diversity technique and employed the Pareto tournament selection method to obtain a set of non-dominated GP individuals (Burks and Punch, 2015). Similarity among individuals was measured using tree fragments, and the

dominance relationship between individuals was determined using both the solution quality and diversity values.

To the best of our knowledge, considering the diversity values among GP individuals as an objective to be optimized has not been addressed in the field of automated design of dispatching rules using the GP approach. Therefore, in the next chapter, a new distance metric is proposed to measure diversity among GP evolved scheduling rules in SJSSPs. The proposed metric prioritizes the nodes closest to the root, takes into account the interaction effect between nodes, and is computationally efficient. In addition, a multi-objective framework is introduced by integrating the GP approach with the NSGA-II algorithm to maintain diversity across generations. Finally, the proposed framework considers the rule size as an objective to be optimized rather than the tie-breaker approach used in literature methods, thus reducing the computational time of the GP algorithm.

2) High Computational Time due to the Bloating Effect and Fitness Evaluation

A well-established phenomenon in GP and other variable-length genome methods is the tendency of evolved programs to grow rapidly over time without a significant return in fitness. This phenomenon is referred to as "*bloat*" (Luke and Panait, 2006). This slows down the search process by wasting computing resources in evaluating large individuals and reduces the likelihood that genetic operators will alter important parts of the evolved programs (Mori et al., 2008). From the practical point of view, larger dispatching rules in complex mathematical structures have high computational costs and are difficult to understand compared with simpler rules. Therefore, when the training time of the GP algorithm becomes more affordable, it will lead to greater adoption in more complex manufacturing environments with fewer assumptions and more objectives to be optimized. Moreover, reducing the size of evolved rules increases the chances of their use in the industry because smaller rules are easier to interpret by decision-makers and to implement in real-world manufacturing systems (Nguyen et al., 2017d; F. Zhang et al., 2021b). The most common way to control bloating is to impose size limits on the evolved programs. The size constraint can be expressed using the maximum allowable tree depth or the maximum number of tree nodes (Crane and McPhee, 2006). Many bloat control methods have been reported in the literature to mitigate the redundant growth of GP individuals, and ten bloat control techniques were compared in (Luke and Panait, 2006). Other methods include code editing, modifying genetic operators, parsimony pressure, and the removal of oversized individuals

(Alfaro-Cid et al., 2010). In addition, the size of the GP individuals is indirectly (tie-breaker) optimized in parallel with fitness and diversity as proposed in (Alfaro-Cid et al., 2010; Burks and Punch, 2015; de Jong et al., 2001).

In terms of using GP to develop scheduling rules, the bloating effect not only greatly increases the computational time, but also evolving rules tend to be too complex for decision-makers to understand, which limits their use in real-world applications (Branke et al., 2016a). Therefore, many scholars have suggested the use of simplification techniques to reduce the complexity and improve the readability of GP rules. These methods are used to remove redundant parts of evolved heuristics using *manual* (Tay and Ho, 2008) or *online* learning (Wong and Zhang, 2006) techniques, thereby making heuristics more compact and easier to interpret. Moreover, since the GP computational time and population diversity in the case of DJSSPs are significantly larger than that of SJSSPs, the proposed multi-objective GP approach is not applicable to the dynamic settings. In other words, integrating a multi-objective approach with the GP algorithm in dynamic settings will increase the computational time without increasing the solution quality of generated rules (prioritise smaller rules even with lower solution quality). Instead, there are two main research themes commonly adopted to mitigate the impact of bloating effect on the GP evolved rules for job shop scheduling problems.

- Feature selection methods have been used in several studies (Mei et al., 2016, 2017a; Shady et al., 2021a) to indirectly control the growth in GP rules by selecting a subset of the most important features and eliminating non-critical features, resulting in a reduction in the size of the generated rules.
- It has been reported that the GEP algorithm generates smaller rules compared with the GP algorithm because it is not strongly affected by the bloat due to its constrained representation (Nguyen et al., 2017a; Nie et al., 2010, 2011, 2013a).

a) Feature Selection

The ability of GP to generate superior rules relies on careful selection of the terminal set that covers the most relevant job, machine, and job shop information (F. Zhang et al., 2021a). However, it is very challenging to ascertain the most important features manually from a large set of system attributes that may encounter interaction effects (Zhang et al., 2019a). Some examples are due-date attributes, which are usually worthless when minimising flowtime-related objectives. The inclusion of irrelevant terminals engenders three main downfalls, as shown below (Mei et al., 2017a).

- It adversely affects GP searching ability by broadening the heuristics search space and thereby hinders the achievement of promising search areas. Precisely, if the tree structure is used with a maximum depth of D and if all functions are binary, then the GP search space size is $|F|^{2^D-1}.|T|^{2^D}$, where F and T respectively stand for the function and terminal sets.
- Generated rules tend to be complex with a large variety of terminals, which hinders the process of understanding how the evolved rules work and how any change will alter the priority value of a given job.
- Longer rules require high computational costs to be evaluated, which can be expected to increase the GP computational burden.

Although GP can perform feature selection automatically, its ability is limited. For example, even the best rules usually include some redundant terminals (Zhang et al., 2019a). The frequency analysis proposed in (Friedlander et al., 2011) is a commonly used feature ranking method. Its main limitation is that the weight of the features can be incorrectly estimated because some terminals might have a higher weight even if they occur more frequently in redundant arithmetic operations than other terminals involved in critical operations (Mei et al., 2017a). Therefore, Mei et al. suggested an offline feature selection method based on the contribution of each terminal to the priority function of the best rules (Mei et al., 2016). This approach regards feature selection as a pre-processing step that requires several computationally expensive GP runs to obtain a variety of good rules for estimating terminals' weights from them. In a later study, Mei et al. proposed an offline feature selection approach for identifying important features of the GP algorithm in dynamic job shop settings (Mei et al., 2017a). The proposed framework consists of three key steps. First, a niching-based search framework is used to extract a diverse set of good rules. Second, a weighted voting method is used to estimate the weight of each terminal and to identify a subset of significant terminals. Finally, the significant terminal set replaces the original terminal set in future GP runs. They reported two limitations associated with the proposed approach. First, although the approach was sufficient to identify a compact set of features, it did not reduce the program size of the evolved rules in later GP runs. Second, the output of feature selection is binary and uses no evolutionary information collected from earlier generations. Zhang et al. proposed a two-stage feature selection framework to evolve routing and sequencing rules for the flexible DJSSP (F. Zhang et al., 2021a). They split the whole GP process into two stages using a predefined checkpoint such as the number

of generations used in their work. For the first stage, they employed a niching method and a surrogate model to identify a set of the most relevant terminals. Then, the new terminal set replaced the original set, which is used in evolving individuals in the generations following the checkpoint's generation. For one study, a new hybrid GP approach based on a new representation, new local search heuristic, and efficient fitness evaluators was developed (Nguyen et al., 2018a). Using this representation, GP rules are defined using two parts: priority functions and attribute vectors. Attribute vectors perform an explicit selection of attributes by deactivating irrelevant terminals, thereby narrowing the search space and improving the interpretability of evolving rules. The authors reported that, although the proposed approach reduced rule length, it obtained more active terminals than the standard GP algorithm did. Earlier studies have revealed some challenges in the current approaches, as described below:

- Most feature selection methods assess the effect of each terminal by the frequency with which it occurs in the best-evolved rules (Riley et al., 2016). The main shortcoming of this technique is that the results may be biased towards irrelevant features because of the occurrence of redundant features.
- Feature selection methods usually adapt offline selection mechanisms (Mei et al., 2016) or a checkpoint to obtain a promising subset of terminals, resulting in additional simulation runs, making the GP algorithm too time-consuming and impractical (F. Zhang et al., 2021a).
- Feature selection methods reported in the literature use a binary discrimination method, i.e., inclusion or exclusion of a feature from the terminal set, which ignores the relative importance of the respective terminals (Mei et al., 2017a).

In order to address the literature limitations, Chapter 4 includes the following contributions.

- A reliable method to estimate the weight of each terminal without being affected by the occurrence of redundant terminals or complex rule structures is developed.
- An online feature selection approach that uses the estimated weights of terminals from earlier generations to guide the search in the current generation is proposed.
- Feature selection probability rather than the inclusion or exclusion method is introduced to provide a broad preference scheme for each feature.

b) Gene Expression Programming

Gene Expression Programming (GEP) algorithm is a GP constrained representation proposed to evolve programs of different sizes and shapes encoded in linear chromosomes of fixed length (Ferreira, 2001). Typically, a GEP chromosome includes one or more genes of equal length connected to each other using a linking function. GEP genes are composed of a head and a tail. The elements in the head are selected from the function set or the terminal set, while those in the tail are only selected from the terminal set (Sabar et al., 2015). To ensure a valid tree expression, it is required that the length of the head h and the tail t satisfy the equation $t = h(n - 1) + 1$, where n is the maximum number of arguments for all functions in the function set. The K-expression is used to decode strings into expression trees. The first element in the gene corresponds to the root of the tree, and each function is attached to as many branches as there are arguments to that function, following a depth-first fashion (Nie et al., 2010). When the last node in a branch is a terminal, the branch stops growing, and thus it is common for GEP genes to have a noncoding region (Ferreira, 2001). A variety of genetic operators have been developed to introduce diversity in GEP individuals. Transposition operators such as Insertion Sequence (IS) transposition, Root IS (RIS) transposition, and gene transposition, have been introduced to select a fragment (successive symbols) and insert it into a specific location in the chromosome. In contrast to the RIS transposition where selected elements must start with a function from the head portion of the GEP individual, IS elements can start with either terminal or function and thus can be selected randomly at any position in the chromosome. Moreover, recombination operators such as one-point, two-point, and gene recombination can be employed to exchange some parts of the chromosomes of two randomly selected parents (Nguyen et al., 2017a).

In (Nie et al., 2013a) and (Zhou et al., 2020a), the authors reported that the linear representation used in the GEP algorithm was able to generate higher quality rules compared with the tree-based GP in much less computational time and smaller sizes. Moreover, the GEP algorithm has been used to evolve scheduling rules under various operational settings, including single machine (Nie et al., 2010), DJSSP (Nie et al., 2011), Flexible Job Shop (FJS) (Nie et al., 2013a), multi-objective FJS (Ozturk et al., 2019), limited buffer spaces (Ozturk et al., 2020), and FJS with setup time (C. Zhang et al., 2021). Since the GEP approach is relatively new compared with the GP algorithm and there are many genetic operators that need fine-tuning before running, the number

of literature articles that have used the GEP for the automatic generation of dispatching rules is very limited. Also, there is no feature selection approach proposed in the GEP existing literature (Nguyen et al., 2017a). Although this is reasonable because the GEP algorithm is less susceptible to the bloating effect, the GEP evolved rules might still include some redundant features. Therefore, the research questions that can be derived from the current GEP literature are as follows.

- Does integrating feature selection methods with the GEP algorithm help reduce computational time? and if yes, will it negatively affect the GEP exploration ability since it is already restricted?
- How can the proposed feature selection method for the tree-based GP approach be modified for use in the GEP algorithm?

In order to find answers to these questions, Chapter 5 includes the following contributions.

- The proposed feature selection approach for the tree-based GP approach is modified to be applicable to the GEP algorithm.
- The performance of the proposed feature selection GEP algorithm is evaluated under different DJSSP instances and objective functions.
- The GP feature selection approach proposed in (Nguyen et al., 2018a) is modified to be applicable to the GEP algorithm to analyse its performance in the case of the GEP algorithm and compare it with the proposed approach.

c) Surrogate Models for Expensive Fitness Evaluation

As mentioned earlier, there are a large number of dispatching rules (in the hundreds) evolve in each generation, and each of these rules needs to be evaluated using a simulation model that imitates a specific scheduling problem. During the evaluation of each rule, thousands of scheduling decisions are made by estimating the priority values of the waiting operations using the given rule (function). Therefore, the most computationally expensive part of using the GP approach to generate dispatching rules for dynamic problems is fitness assessment with simulation (Nguyen et al., 2017a). Several surrogate models have been proposed in the evolutionary computation literature to overcome the computational burden caused by fitness assessments (Jin, 2011).

Surrogate models are computationally affordable approximations of an expensive fitness function derived from statistical or machine learning techniques that are trained using samples of fully evaluated solutions (Jin, 2011). Integration of surrogate models

and evolutionary algorithms aids in the early identification of promising individuals whose actual performance will subsequently be assessed using an expensive fitness function. In addition, poor-quality individuals can be quickly discarded without sacrificing high computational costs (Hildebrandt and Branke, 2015a). Two surrogate models have been proposed in the literature to reduce the fitness evaluation time required for the GP algorithm to evolve rules for DJSSPs as described below.

1) Hildebrandt and Branke proposed a surrogate model based on the phenotypic characterization of evolved rules (Hildebrandt and Branke, 2015a). The proposed model estimates the fitness value of a certain rule by using the fitness of the most similar rule generated in the previous evolutionary generations. Then, the rules with the highest approximate fitness values are selected to form the next generation, and their real fitness values are obtained using the actual simulation model. The results showed that the proposed surrogate assisted GP approach had a higher convergence speed compared with the standard GP approach given the same computational budget.
2) Nguyen et al. developed surrogate models based on simplified versions of the original job shop; i.e., three simplified versions of the actual job shop settings were introduced using a smaller number of machines, operations at each job, and the total number of jobs required (Nguyen et al., 2017d). The results showed that the simplified methods significantly reduce the computational costs of fitness assessment while achieving higher accuracy levels compared with the phenotypic surrogate model.

Although the existing surrogate assisted GP models have shown better performance compared with the standard GP algorithm, there are some limitations that need to be addressed as follows.

1) The decision vector used in the surrogate model proposed in (Hildebrandt and Branke, 2015a) has to be adequately large to differentiate between GP rules, resulting in an increase in computational time. In addition, the prediction accuracy of this surrogate model is significantly lower compared with the simplified model as reported in (Nguyen et al., 2017d).
2) Although the simplified models achieve high prediction accuracy with a significant reduction in computational costs by reducing the job shop complexity, all information obtained during simulation runs is discarded.

In order to address the above limitations, Chapter 6 includes the following contributions.

- Three surrogate models are developed to reduce the computational time required to evaluate the performance of GEP rules. The proposed models are built on the simplified models that have higher prediction accuracy compared with the phenotypic surrogate and do not use any decision vectors.
- The surrogate assisted GEP approach uses the information collected during fitness assessment of a subset of rules, in training a simple mathematical function to replace part of the simulation length (reduce simulation time).
- This is the first attempt to use machine learning methods to abstract a simulation model of DJSSPs, which opens up a wide range of research opportunities to develop other machine learning techniques using the same concept.

Chapter 3. MULTI-OBJECTIVE GENETIC PROGRAMMING APPROACH FOR STATIC JOB SHOP SCHEDULING PROBLEMS

3.1 Introduction

The first objective of this chapter is to propose a distance metric for measuring the similarity between GP individuals to increase the solution quality of evolved rules. Because static job shop scheduling problem instances do not change across evolutionary generations, high-quality rules ensure that the same high performance is obtained in subsequent generations, which is not the case in dynamic settings. Therefore, they have a higher probability to take over the entire population before effectively exploring the fitness landscape (premature convergence) (Nguyen et al., 2017a). Representing the population diversity in numerical formats helps to track and control it across generations. In other words, a higher selection probability can be given to distinct individuals that have low similarity values regardless of their fitness values, and thus can survive for the upcoming generations (Burke et al., 2004). In addition, promoting population diversity enhances the exploration ability of the GP algorithm to explore large regions of the solution's space without being trapped in local optima, which is reflected in the quality of the obtained solutions. However, considering population diversity only as an objective to be optimized leads to higher over-exploration and thus computational budget might run out before the algorithm converges to a good solution. Consequently, the solution quality of GP individuals must be taken into account while optimizing their diversity values.

The second objective is to develop a multi-objective GP framework to simultaneously optimize solution quality, diversity value, and rule length. Controlling the growth in the size of the evolved rules significantly reduces the computation time of the GP algorithm as stated earlier. In addition, the rules in simple structures are easier to interpret by the decision-makers and to implement in real-world manufacturing systems (Branke et al., 2016a). In most multi-objective GP methods in the literature, the size of rules is indirectly considered. Meaning that when the two compared rules have the same fitness value, the smaller rule is chosen in the next generation. The reason is that smaller rules usually include limited information about the job shop floor and thus have low performance compared with larger rules. Also, smaller rules have a smaller range of potential changes (lower diversity) compared with larger rules. Finally, it is clear that optimizing the three objectives simultaneously is suitable due to their conflicting nature.

The remainder of this chapter is organized as follows. Section 3.2 provides a detailed explanation of the proposed distance metric and the multi-objective GP approach. The experimental details are presented in Section 3.3 including comparison design, static job shop scheduling problem instances, and GP parameters. Section 3.4 provides the results in terms of parameter tuning, makespan, and mean tardiness objective functions. Finally, Section 3.5 presents the conclusions of this chapter.

3.2 Proposed Approach

3.2.1 Distance Metric

Because the purpose of developing this new distance metric is to promote diversity among dispatching rules, a behaviour analysis was carried out for the scheduling rules that were evolved using GP. The preliminary runs enabled four observations to be made as follows.

a) The fitness evaluation of rules is computationally expensive, restricting the use of phenotypic distance measures.
b) The structure of GP individuals converges to that of the best rule in a given generation.
c) Not only does the location of the nodes greatly affect their performance, but the interaction between the node and its parents also has a significant impact.
d) Not all nodes have the same weight, because it was noted that nodes near the root had a greater impact on the fitness value of a given rule than distant nodes.

These findings were addressed in the proposed distance metric. To reduce the computational requirements, a genotypic metric is proposed that does not require any additional fitness evaluations. Moreover, instead of evaluating the similarity between every pair of rules in a GP population, the similarity between each rule and the best rule is evaluated, which reduces the similarity calculations from $O(n^2)$ to $O(n)$. Finally, the distance between the two rules is calculated in three steps as shown below:

I. The two trees i and j are brought to the same structure (depth) by adding $null$ nodes.

II. Weighted edge sets S_i and S_j for the two rules are generated using the following equations:

$$S_i = \{w(e_i) \; \forall \; e_i \in E_i\}, S_j = \{w(e_j) \; \forall \; e_j \; \in E_j\} \tag{3.1}$$

where e is an edge in the set of all Edges E in a specific tree, and $w(e)$ is a function that estimates the weight of edge e.

The weight $w(e)$ of an edge e connecting nodes x and y is estimated using the following equation:

$$w(e) = \frac{w_x \; + \; w_y}{2} \tag{3.2}$$

where w_x and w_y represent the weights of nodes x and y respectively, which can be evaluated as follows:

$$w_x = k^{(d_{max} - d_x + 1)}, and \; w_y = k^{(d_{max} - d_y + 1)}$$

where d_{max} is the tree depth, d_x and d_y denote the depth of nodes x and y. k is a constant ≥ 1 to indicate that the difference at depth k in the compared trees is k times more important than the difference at depth $k + 1$.

III. The similarity value S_{ij} between trees i and j are measured using the following equation:

$$S_{ij} = \frac{\sum_{e \in E_i \cap E_j} (w(e_i) \; + \; w(e_j))}{\sum_{e \in E_i \cup E_j} (w(e_i) \; + \; w(e_j))} \tag{3.3}$$

An illustrative example of how the proposed distance metric measures the similarity between two trees is shown in Figure 3.1. The two rules are presented in both the tree structure and mathematical form. Three terminals are presented: processing time PT, work in the next queue $WINQ$, and next processing time NPT. The depth of the first

rule ($d_1 = 2$) is incremented by one using a null node to obtain the same depth as the second rule ($d_2 = 3$). The depth and weight for each node are shown in columns d and w next to the tree structure of rule 2. The weights of the nodes were assigned using $k = 2$. Clearly, the set of common edges includes $+PT, + \times, \times NPT, and \times WINQ$. The similarity value between the two rules is estimated by dividing the sum of the weights of the common edges by the total weight of all the edges in both rules. There is a similarity value of 0.7 between the two rules. From this example, when $k = 2$, the weight of a node is half the weight of its parent node, that is, increasing the depth by one will reduce the edge weight by half. In addition, the location of an edge (right or left) does not affect its weight because edges at the same depth have the same weight regardless of their position. In addition, the similarity value ranges from 0 to 1 and can be evaluated even if the two trees have different depths or dissimilar root nodes. Finally, the proposed metric utilizes the information available in the structure of the rules to be compared; thus, additional fitness evaluations are not required.

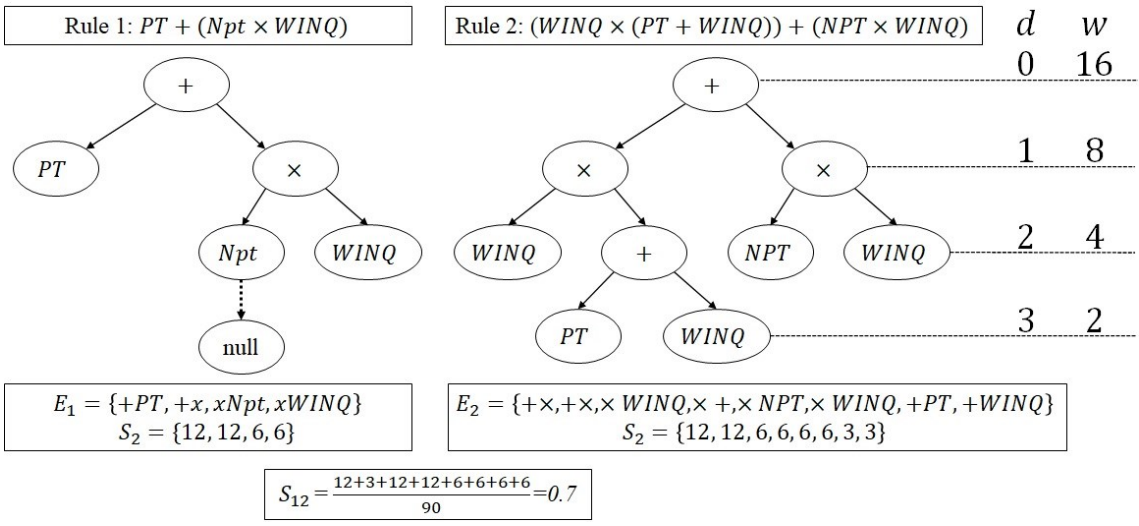


Figure 3.1: Example of the proposed similarity measure.

3.2.2 Multi-objective GP approach

The aim of the proposed approach is to extend the standard GP algorithm (John R. Koza, 1994) by enhancing diversity and reducing the bloat effect in the evolved rules. Non-dominated Sorting Genetic Algorithm-II (NSGA-II) is one of the most popular Pareto-based multi-objective algorithms used to obtain a set of well-spread Pareto-optimal solutions (Deb et al., 2002a). In addition, Zhang et al. (Zhang et al., 2019b) integrated the strategies of NSGA-II and SPEA2 into GP to solve the dynamic flexible JSSP. The results showed that incorporating NSGA-II with GP generated more effective

rules with higher consistency than SPEA2. Therefore, the NSGA-II was integrated with the GP algorithm to simultaneously optimize the three objectives at the same time. The objectives considered were the fitness value, similarity value, and rule depth. The pseudocode for the proposed algorithm is presented in Figure 3.2. The algorithm begins by generating a population of dispatching rules using a predefined set of functions and terminals. Evolutionary generation begins by using each of these rules to construct a schedule for a given JSSP instance. The objective values of the rules are then estimated from the constructed schedules, and the best rule in this generation is updated accordingly. Parents and offspring are combined to ensure elitism, which guarantees that the best individuals secure a place in the next generation. Then, the similarity value between each rule and the best-evolved rule is assessed using the proposed distance metric. NSGA-II is used to select rules with high ranks and smaller crowding distances for new population formation. If the required number of generations is not reached, another evolutionary iteration starts; otherwise, the algorithm terminates, and a set of non-dominated scheduling rules is returned.

Inputs: Job shop scheduling problem instance I
Outputs: The Pareto front of non-dominated rules P^*
Initialize population $P \leftarrow \{X_1, X_2, \dots, X_{popsize}\}$
Set generation $g \leftarrow 0$
while $g < g_{max}$ **do**
  Apply genetic operators to P (generate offspring Q)
  Best rule $X_b \leftarrow 0$
  foreach rule $X_i \in Q$ **do**
    Construct a schedule $\Delta(X_i, I)$
    Calculate the objective value $f(\Delta(X_i, I))$
    if $f(\Delta(X_i, I)) < f(\Delta(X_b, I))$ **do**
      $X_b \leftarrow X_i$
    end if
  end for
  $T \leftarrow P \cup Q$
  foreach rule $X_i \in T$ **do**
    Calculate the similarity value S_{X_i, X_b}
  end for
    $P \leftarrow$ NSGA-II $select(T)$; and $g \leftarrow g + 1$
end while
return the non-dominated individuals $P^* \subseteq P$

Figure 3.2: Pseudocode of the proposed multi-objective GP algorithm

3.3 Experimental Setup

This section presents the experimental setup used to compare the proposed approach with the other methods. In addition, the parameter settings for both the GP algorithms and JSSP instances are shown.

3.3.1 Comparison Design

To assess the effectiveness of the proposed distance metric and multi-objective GP approach, two algorithms are developed. The first algorithm, referred to as "PGP_P," evaluates the ability of the proposed similarity metric to increase the quality of GP rules by promoting population diversity. The second algorithm, denoted as "PGP_N," follows the same steps presented in Figure 3.2 and is used to verify whether the proposed approach can optimize the solution quality, diversity, and size of evolved rules simultaneously. The two proposed algorithms were compared with the following three algorithms in the literature:

i. SGP: Standard GP algorithm (John R. Koza, 1994).
ii. EGP: SGP with Edit-distance metric (de Jong et al., 2001).
iii. MGP: SGP with genetic Marker metric (Burks and Punch, 2015).

The SGP algorithm was developed to evaluate the benefits of increasing the population diversity and controlling the bloating effect compared with the standard version of the GP algorithm. In addition, the EGP and MGP algorithms were considered to enable the proposed algorithm to be compared with other GP algorithms that use different distance metrics and bloat control techniques. The edit distance metric used in the EGP algorithm measures the distance between two individuals as the shortest number of editing operations required to convert one tree to the other. Specifically, the edit metric works in three steps as follows (de Jong et al., 2001).

a) The distance between each individual in the GP population and the best individual found so far is measured.
b) Two overlapping nodes get a distance of 1 when they are different, otherwise (same), the distance is 0.
c) The distance between two trees is the sum of all different nodes divided by the size of the smaller tree for normalizing.

In addition, the authors proposed a multi-objective GP optimization approach to avoid bloat and prompt population diversity. Experimental results showed that the

proposed approach had better performance compared with the standard GP on the 3, 4, and 5-parity problems both with respect to computational time and tree size. On the other hand, the MGP algorithm uses a genetic marker distance metric that estimates the similarity between individuals by comparing a small portion of each tree known as a "genetic marker". In other words, the MGP algorithm enhances diversity by maintaining an appropriate number of unique genetic markers (tree fragments) in the population (Burks and Punch, 2015). The genetic marker distance metric promotes diversity using the following steps.

i. Depth-first traverse is performed for each tree up to a predefined depth (end of the genetic marker).
ii. The resulting Lisp expression becomes the genetic marker.
iii. The density of a genetic marker is used to determine the prevalence of a specific genetic marker in a population.

Finally, the MGP algorithm outperformed three literature algorithms on benchmark problems with respect to solution quality and convergence speed.

A flowchart of the five algorithmic experiments is shown in Figure 3.3. All algorithms begin by initializing the same population of scheduling rules using the same random seed. Subsequently, their fitness values were evaluated on a specific JSSP instance using a predefined objective function. In the case of SGP, no diversity metric is used, and the best rules are selected by the standard tournament method. For the EGP and MGP algorithms, the similarity values of the rules were evaluated using the edit distance and genetic marker distance metrics, respectively. The proposed distance metric was used for both PGP_P and PGP_N. For the PGP_N algorithm, the Pareto front rule with the lowest fitness value was used to measure the similarity values. The Pareto tournament method (Schmidt and Lipson, 2011) was employed to select the fittest rules for the EGP, MGP, and PGP_P algorithms. The same selection mechanism was used to evaluate the performance of the proposed metric separately. In the case of PGP_N, the best rules were selected using NSGA-II. Genetic operators were then applied to create a new population of scheduling rules. If the stopping criterion is satisfied, the algorithm terminates; otherwise, a new evolutionary iteration begins.

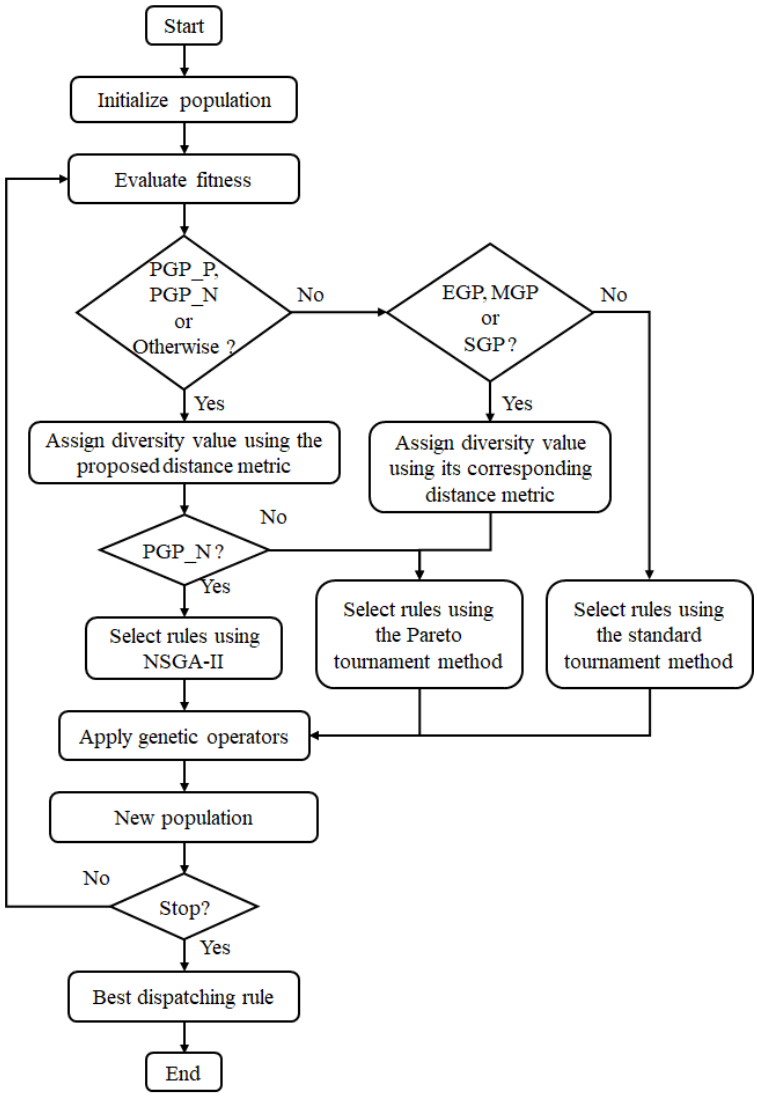


Figure 3.3: Flowchart of the five developed algorithms

3.3.2 Genetic Programming Parameters

Ten JSSP instances (ta61-ta70) based on previously proposed benchmark data (Taillard, 1993) were used for comparison purposes. These instances are chosen from the 80 available instances because they are the most complex instances with the largest number of jobs and machines, and the optimal solutions are known for the makespan objective (for validation). The job shop contains 20 machines, and 50 jobs are processed. Each job must visit all machines following a predetermined routing path. The processing times follow a uniform distribution U [1, 99]. Two objectives were investigated: makespan and mean tardiness. These objectives are chosen because the makespan represents the level of productivity that the system can achieve, whereas the mean tardiness objective assesses its ability to meet customer due dates (customer satisfaction level). Job due dates are estimated using the total work content method (Nguyen et al.,

2017a), where job due date = current time + tightness factor × total processing time, as these dates are not available in the instances reported in the literature. A tightness factor of 1.9 was used as proposed in a previous report (Sels et al., 2012a). Regarding the GP parameters used for the five algorithms, a population of 250 rules was created using the ramped-half-and-half method, with a maximum depth of 8. Table 3.1 lists the set of functions and terminals used. In addition, the underlined terminals are used when optimizing the mean tardiness objective, and are excluded in the case of the makespan objective. The function set comprises basic arithmetic operations and logical functions. The values of the crossover and mutation rates are set to 0.9 and 0.1. The algorithm terminates after 30 generations are completed. Therefore, increasing the population diversity will only affect the fitness value of the best-evolved rule at a specific generation without affecting the computational time since the number of generations is fixed. Because of the randomness inherent in the GP algorithm, 20 replications were performed. Also, the Wilcoxon rank-sum test was used with a significance level of 5 %.

Table 3.1 GP terminal and function sets

Attribute	**Explanation**
JR	Job release date
OR	Operation ready time
WR	Work remaining of the job
PT	Operation processing time
RO	Number of remaining operations in a job
WT	Operation waiting time
NPT	Processing time of the next operation
WINQ	Work in the next queue
APR	Average processing time of queued job
DD	Job due date
CT	Machine ready time (current time)
SL	Job slack
Function set	$+,\ -,\ \times,\ /,\ max,$ and abs

3.4 Results

This section discusses the evaluation of the two proposed algorithms with respect to the three approaches. In Section 3.4.1, parameter k is fine-tuned by analysing its impact on the performance of the proposed similarity metric. To gain in-depth insights into the behaviour of the five algorithms, the quality of the GP individuals was tracked across generations. Additionally, the results obtained by applying the five algorithms to the ten job shop scenarios are presented in subsections 3.4.2 and 3.4.3 for the makespan and mean tardiness objectives, respectively.

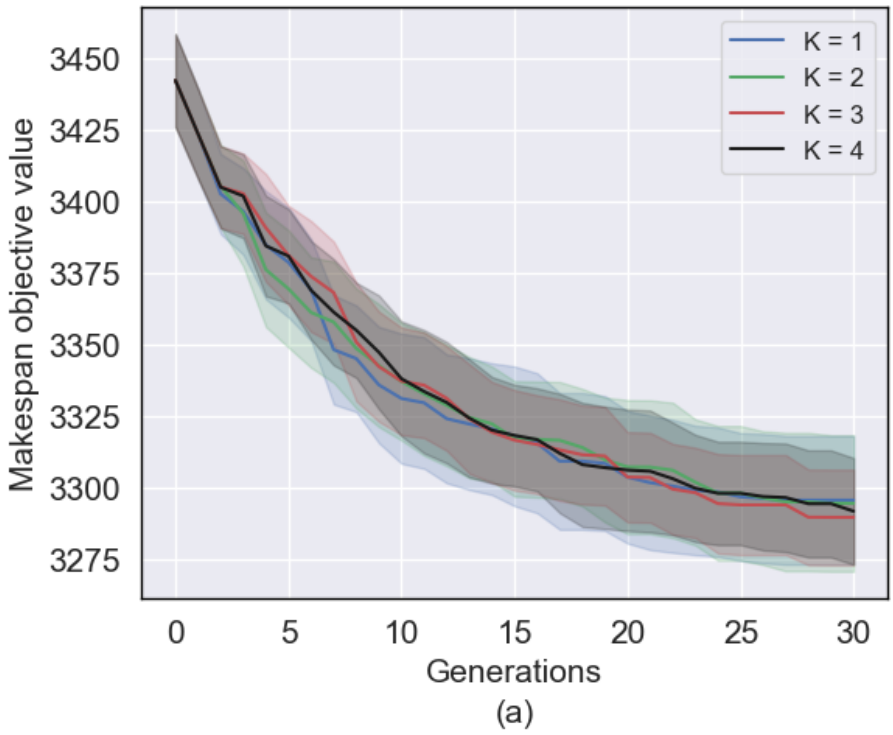
3.4.1 Parameter Analysis

The PGP_P algorithm is implemented using four values of the k parameter k={1,2,3,4}, where $k = 1$ represents the case in which a node takes a weight equal to its depth. In addition, four performance measures were considered:

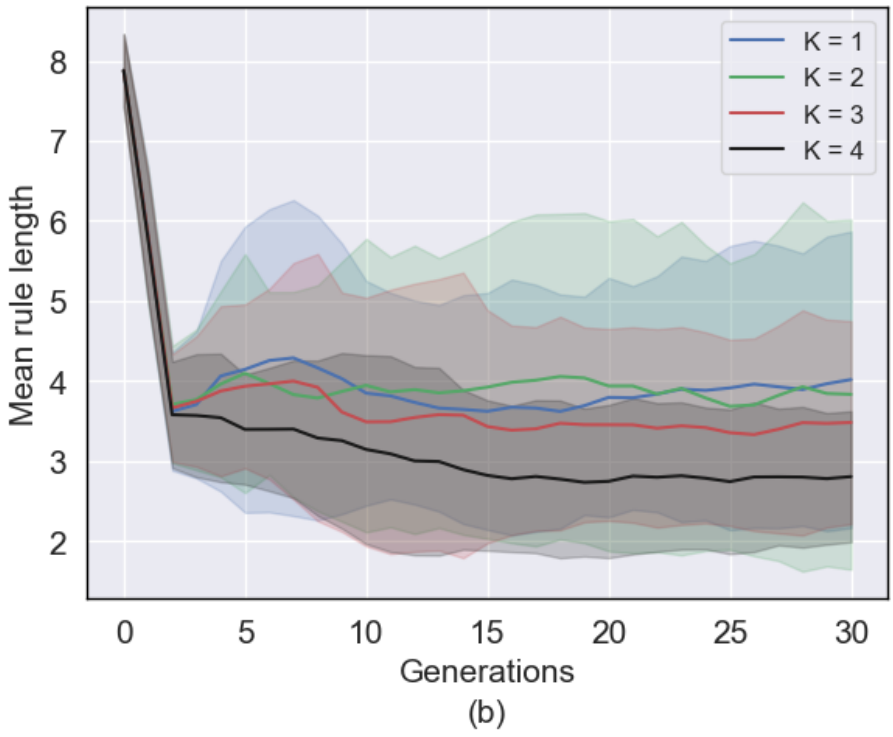
- **Solution quality:** average objective values of rules in a given generation.
- **Mean rule length:** the average number of terminals included in GP individuals.
- **Genotypic diversity:** the number of unique rules. Two rules are identical if they have the same structure and content (Burke et al., 2004).
- **Phenotypic diversity:** the number of unique fitness values in a population.

The experiments in this subsection were performed in the ta61 instance, as similar results were obtained in the other nine instances and presented numerically in the next subsections. In addition, the aim of this analysis was to investigate the performance of the algorithms across generations rather than their overall performance which is presented in the next subsections. Figure 3.4 shows the results obtained by implementing the PGP_P algorithm using the four values of the k parameter, where the mean values are shown as a solid line, and the standard deviations are depicted as a shaded area around it. As shown in Figure 3.4 (a), changing the value of the k parameter does not affect the solution quality of the evolved rules, as there is no significant difference between the four runs. In contrast, the average rule length is highly sensitive to the k value, as shown in Figure 3.4 (b). Similar observations are revealed for the other two measures, as shown in Figure 3.4 (c) and (d), where altering the k value had a great impact on the population diversity. The interesting finding here is that even though the population diversity changes according to the value of parameter k, the GP reasoning ability can mitigate this effect on the quality of the generated rules (similar

makespan results). Specifically, when the value of k is 4, if the node depth is decreased by one, its weight increases fourfold. Thus, the GP search explores the search space of small rules (higher impact on diversity), resulting in a reduction in the length of new rules, as shown in Figure 3.4 (a) and (b), where $k = 4$ has the smallest average rule length with the same fitness values. The distance metric with $k = 3$ was used in the next GP runs because it yielded acceptable results for the four performance measures.



(a)



(b)

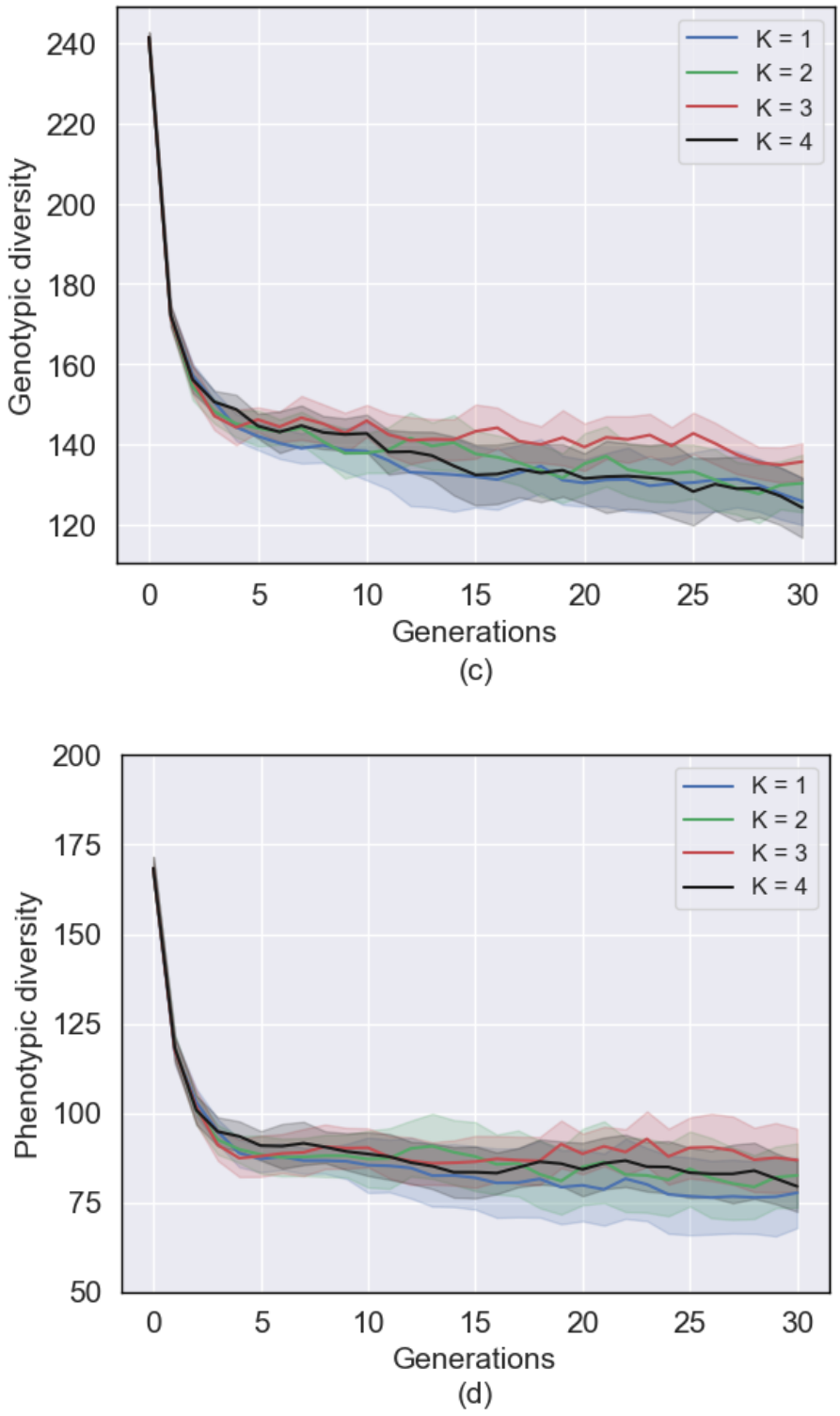


Figure 3.4: Effect of changing the value of the *k* parameter on the four performance measures.

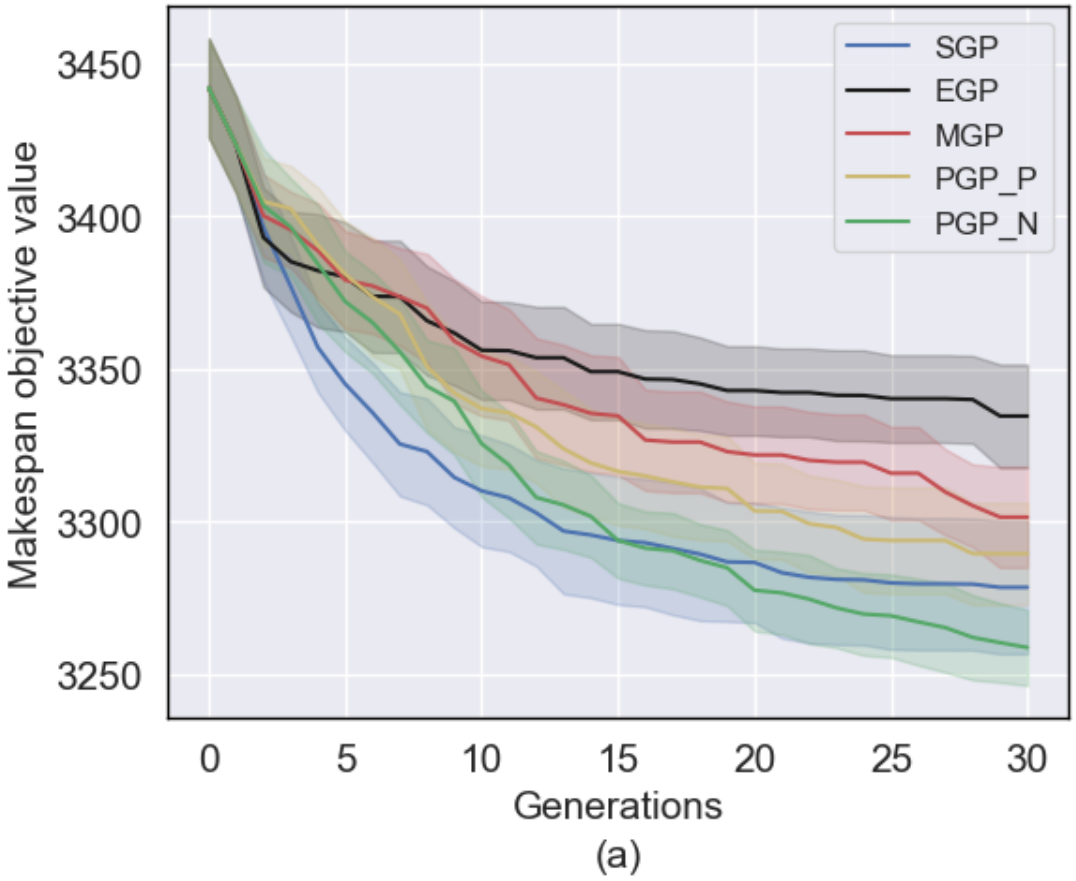
In Figure 3.5, the behaviour of the two proposed algorithms is analysed against that of the three algorithms from the literature on the ta61 instance with respect to the quality of evolved rules, rule length, computational time, and population diversity. Regarding the quality of the evolved rules, the PGP_N algorithm obtained the lowest

makespan value with a consistent improvement across generations, as shown in Figure 3.5 (a). In addition, the exploration ability of PGP_P was superior to that of the EGP and MGP algorithms. It is noteworthy that although the SGP algorithm obtained higher quality makespan results compared with the EGP, MGP, and PGP_P algorithms, these results would be reversed if the computational time was used as a stopping condition instead of the number of generations, as illustrated in Figure 3.5 (c). The use of the Pareto tournament selection method in the EGP, MGP, and PGP_P algorithms significantly decreased the average rule size by approximately 75% compared with the SGP algorithm, resulting in a reduction in computational time of approximately 56.5%, as shown in Figure 3.5 (b) and (c). Therefore, when comparing the other JSSP instances in the next subsection, the average rule length was used as an indicator of the computational time of the algorithm.

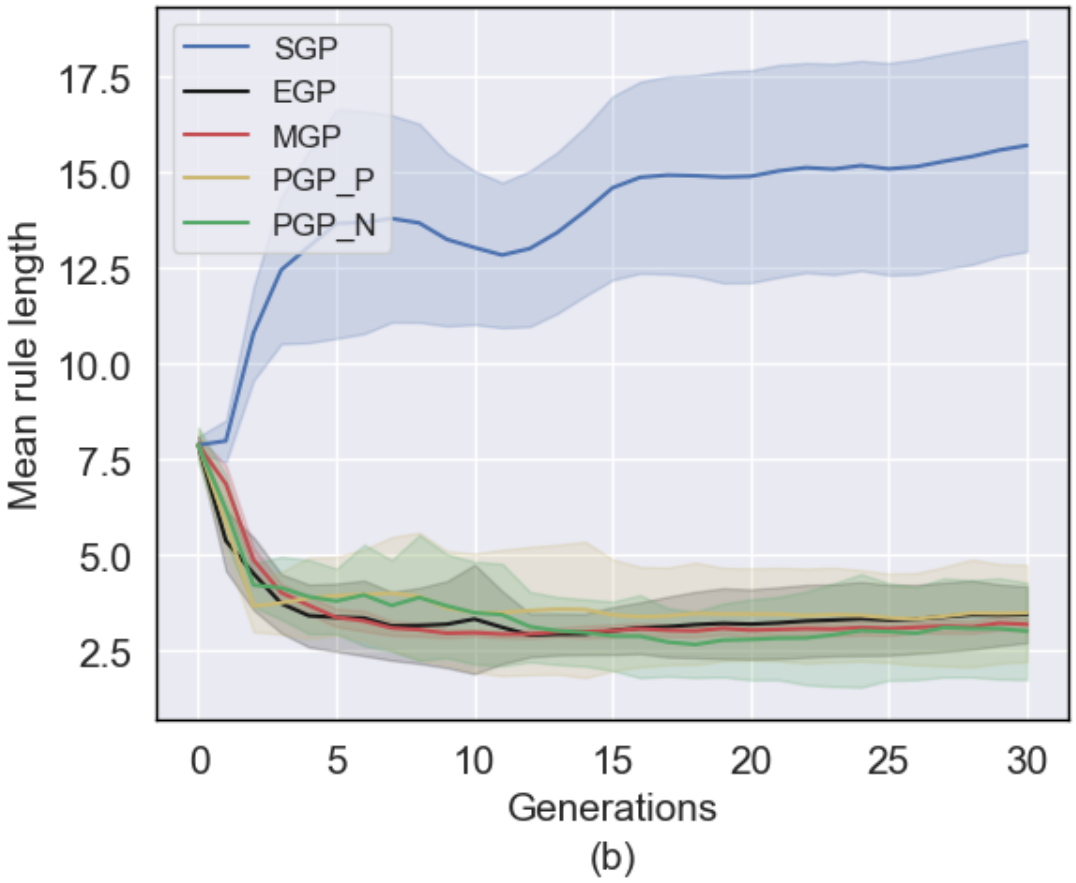
Regarding the diversity of GP individuals, the proposed PGP_N algorithm obtained higher genotypic and phenotypic diversity values across generations compared with other methods, especially in the first ten generations as depicted in Figure 3.5 (d) and (e). In addition, although the SGP algorithm generated high-quality rules, it had the lowest genetic and phenotypic diversity, which explains the exponential growth in the mean rule length as SGP strives to escape from local optima by increasing the rule sizes. Moreover, the use of the proposed distance metric increases the phenotypic diversity of GP rules compared with the literature similarity measures without sacrificing the performance or size of the rules. These results led to two main findings.

1. The PGP_N algorithm was able to generate rules with lower makespan values compared with the other methods because of the high diversity levels, especially in the early generations.
2. The use of NSGA-II as a section mechanism enhanced the exploration ability of the GP algorithm compared with the standard or Pareto tournament selection methods used in the other algorithms.

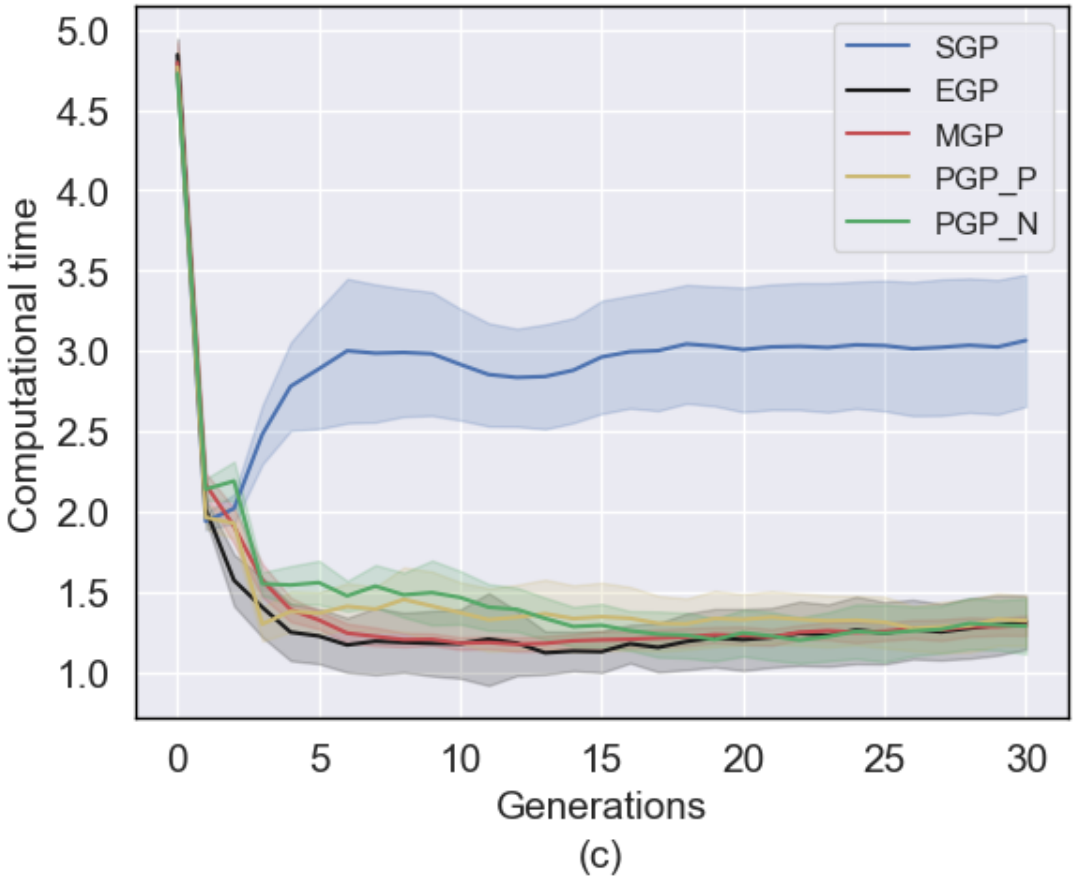
Finally, the PGP_N algorithm obtained a mean makespan value of 3315 with a 7.2% deviation from the optimal solution (2868) of this JSSP instance, which proves that GP is a promising approach for the automated design of scheduling rules.


SGP
EGP
MGP
PGP_P
PGP_N
Makespan objective value
3450
3400
3350
3300
3250
0
5
10
15
20
25
30
Generations

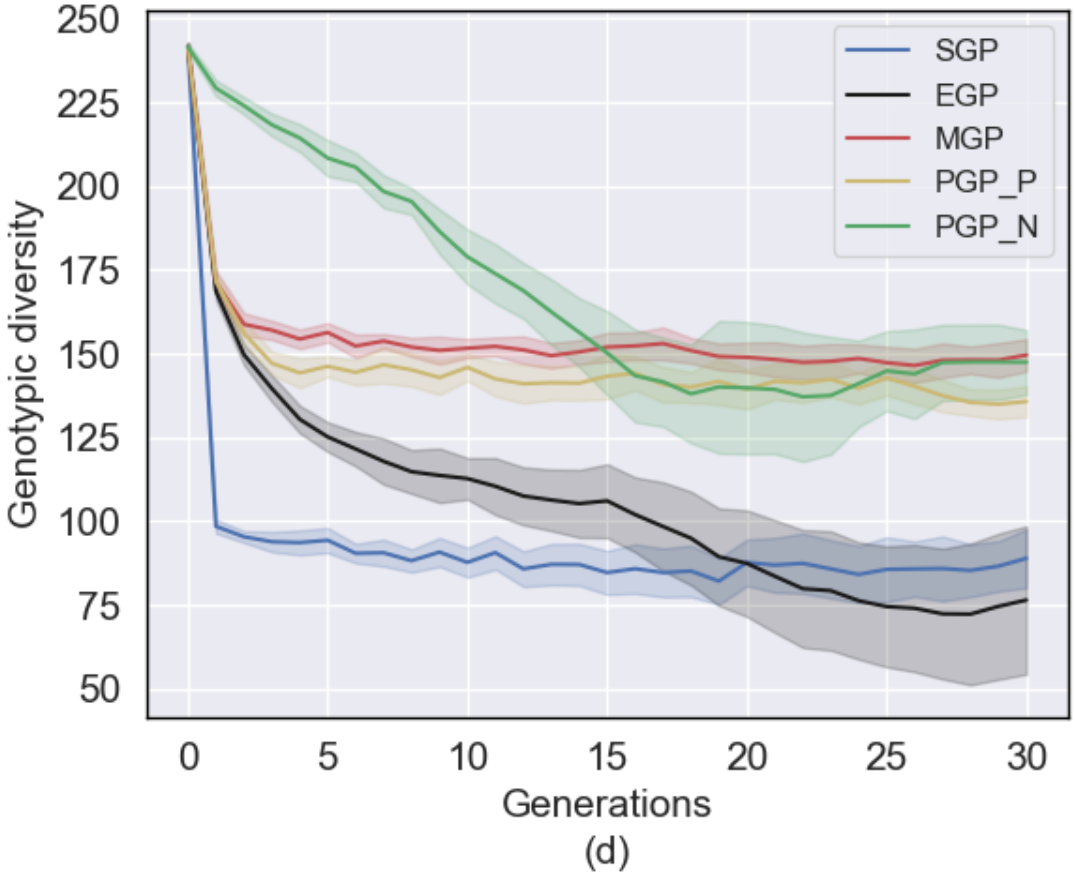
(a)


SGP
EGP
MGP
PGP_P
PGP_N
Mean rule length
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
5
10
15
20
25
30
Generations

(b)


Computational time
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
5
10
15
20
25
30
Generations
SGP
EGP
MGP
PGP_P
PGP_N

(c)


Genotypic diversity
250
225
200
175
150
125
100
75
50
0
5
10
15
20
25
30
Generations
SGP
EGP
MGP
PGP_P
PGP_N

(d)

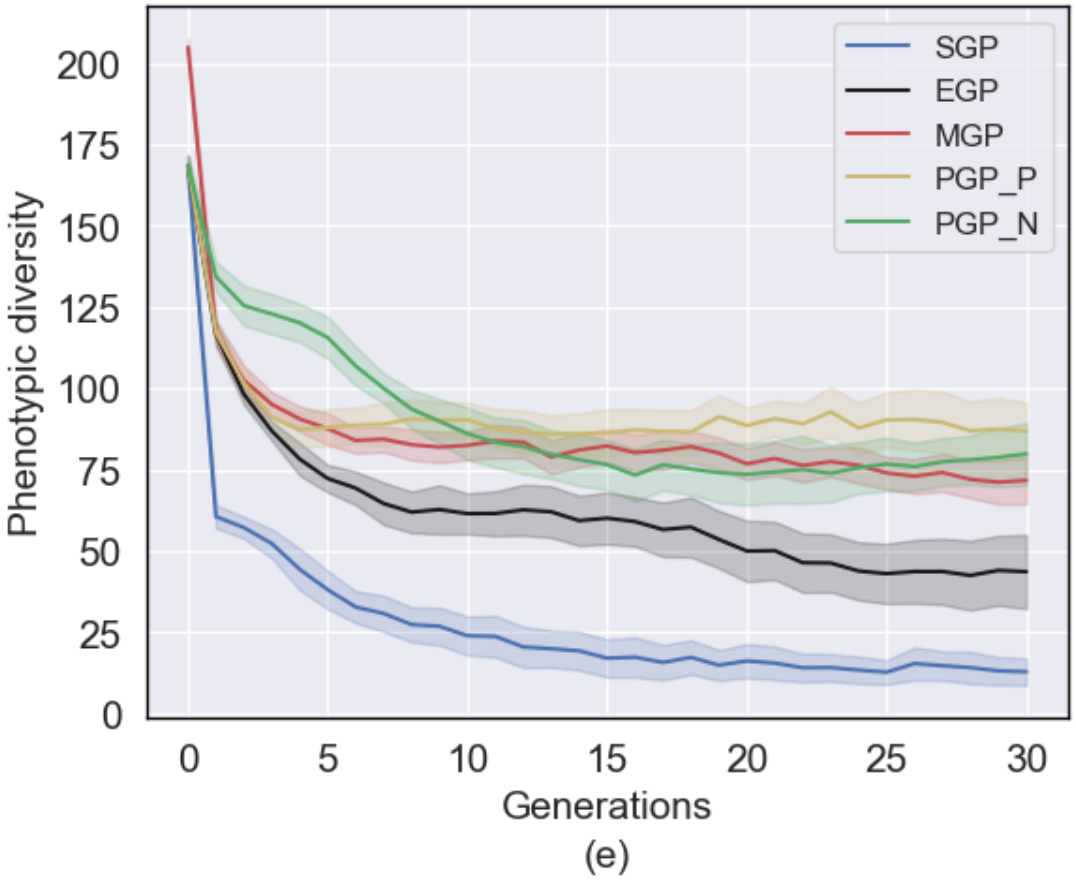


Figure 3.5: Performance of the five GP algorithms on the ta61 JSSP instance.

3.4.2 Makespan Objective

The performances of the five GP algorithms for optimizing the makespan objective were estimated. Ten instances from the literature were used and four performance measures were evaluated. These measures were Makespan Value (MV), average Rule Length (RL), Genotypic Diversity (GD), and Phenotypic Diversity (PD). The mean and standard deviation of the obtained results as are listed in Table 3.2. The statistical results achieved by comparing PGP_P with the three algorithms are represented by the tuple next to the PGP_P results (PGP_P versus SGP, PGP_P versus EGP, and PGP_P versus MGP). The symbols "+," "−," and "=" indicate that the corresponding result is significantly better, worse than, or similar to its counterpart, respectively. In addition, the performance of PGP_N is compared with that of the three literature methods, as well as the PGP_P algorithm represented by the fourth element in the tuples next to its results. The last row of the table summarizes the results counting the number of times a certain method loses (significantly worse) or wins (significantly better) against the PGP_P and PGP_N algorithms (lose/win).

Regarding the quality of the evolved rules, the PGP_N algorithm delivered similar performance to the SGP with only one loss, whereas the PGP_P produced inferior results in four instances. In contrast, the proposed algorithms significantly outperformed the SGP algorithm in terms of the average size of the rules and diversity objectives. The rules evolved using the proposed methods were able to achieve smaller makespan values compared with the EGP and MGP algorithms with wins in almost all instances. In addition, the population diversity of the PGP_P and PGP_N rules was significantly larger than that of the EGP rules in the ten instances. In contrast, although the phenotypic diversity of the PGP_P algorithm was greater than that of the MGP algorithm in the nine scenarios, the genotypic diversity was significantly worse in all instances. These results are consistent with those in the literature, as good fitness values were reported to be correlated with high phenotypic diversity. Therefore, it is clear that the phenotypic diversity among the GP evolved rules has a more pronounced effect on solution quality compared with the genotypic diversity. The integration of NSGA-II and the proposed distance metric greatly increased the genotypic diversity and reduced the average length of rules compared with the PGP_P algorithm.

The following conclusions were drawn. The edit distance method used in the EGP algorithm is clearly unsuitable for measuring the similarity between GP rules in JSSPs because it produced the worst results for the GD and PD measures. In contrast, the genetic marker diversity measure used in the MGP algorithm achieved high levels of genotypic and phenotypic diversity among the GP rules. Finally, the rules evolved using the PGP_N algorithm have the lowest makespan values, highest diversity, and smallest average rule length, which demonstrates the usefulness of integrating NSGA-II with the GP algorithm while using the proposed distance metric.

Table 3.2 Performance of the five GP algorithms in terms of optimizing the makespan objective on the ten JSSP instances

Inst.	Perf.	SGP	EGP	MGP	PGP_P	PGP_N
ta61	MV	3078.15 ± 16.4	3152.5 ± 12.5	3129.55 ± 17.2	3102.6 ± 11.6 (-, +, +)	3077.0 ± 13.0 (=, +, +, +)
	RL	14.09 ± 2.6	3.73 ± 1.1	3.38 ± 0.2	3.3 ± 0.8 (+, =, +)	3.19 ± 0.7 (+, +, =, +)

	GD	89.82 ± 6.3	103.33 ± 13.3	155.82 ± 3.4	145.68 ± 5.0 (+, +, -)	180.59 ± 8.3 (+, +, +, +)
	PD	32.08 ± 5.5	62.47 ± 7.7	93.21 ± 5.1	99.1 ± 5.9 (+, +, +)	103.9 ± 6.6 (+, +, +, =)
ta62	MV	3133.15 ± 13.8	3188.75 ± 28.0	3158.2 ± 14.8	3142.3 ± 7.8 (-, +, +)	3135.2 ± 7.5 (=, +, +, =)
	RL	11.61 ± 2.2	3.07 ± 0.6	3.15 ± 0.2	3.08 ± 1.0 (+, =, +)	2.79 ± 1.1 (+, =, =, +)
	GD	96.47 ± 6.7	78.39 ± 13.9	149.74 ± 4.2	137.44 ± 6.0 (+, +, -)	164.35 ± 11.4 (+, +, +, +)
	PD	25.35 ± 4.0	41.48 ± 7.1	74.75 ± 4.8	84.34 ± 6.6 (+, +, +)	79.28 ± 9.7 (+, +, =, -)
ta63	MV	2960.9 ± 10.3	3003.4 ± 17.7	2984.65 ± 8.8	2967.45 ± 11.6 (=, +, +)	2956.45 ± 10.1 (=, +, +, =)
	RL	13.9 ± 2.5	3.48 ± 1.3	3.42 ± 0.2	3.52 ± 1.2 (+, =, -)	3.16 ± 0.9 (+, =, +, +)
	GD	95.28 ± 6.5	101.24 ± 10.7	156.87 ± 3.3	143.83 ± 4.6 (+, +, -)	177.07 ± 9.6 (+, +, +, +)
	PD	25.68 ± 4.6	57.09 ± 6.7	85.27 ± 4.7	88.1 ± 6.7 (+, +, +)	86.99 ± 6.8 (+, +, =, =)
ta64	MV	2857.75 ± 18.2	2912.5 ± 15.0	2885.3 ± 7.2	2861.2 ± 9.5 (=, +, +)	2853.5 ± 7.0 (=, +, +, =)
	RL	13.85 ± 3.0	3.16 ± 0.7	3.11 ± 0.2	3.19 ± 1.4 (+, -, -)	2.9 ± 1.0 (+, =, =, +)
	GD	90.36 ± 6.5	98.81 ± 11.7	156.95 ± 3.1	142.15 ± 5.4 (+, +, -)	178.88 ± 9.6 (+, +, +, +)

	PD	25.95 ± 4.7	42.52 ± 6.5	65.93 ± 4.7	76.51 ± 6.9 (+, +, +)	79.68 ± 8.7 (+, +, +, =)
ta65	MV	2988.9 ± 21.0	3035.6 ± 13.3	3024.85 ± 17.9	2987.65 ± 16.8 (=, +, +)	2994.85 ± 14.1 (=, +, +, =)
	RL	12.99 ± 2.8	3.2 ± 0.9	2.86 ± 0.2	2.92 ± 1.2 (+, =, -)	2.56 ± 1.1 (+, +, =, +)
	GD	86.92 ± 7.8	75.58 ± 15.3	147.65 ± 3.9	132.8 ± 13.7 (+, +, -)	140.17 ± 17.0 (+, +, -, =)
	PD	26.51 ± 4.7	36.62 ± 7.2	56.46 ± 4.7	77.37 ± 8.6 (+, +, +)	65.22 ± 9.6 (+, +, +, -)
ta66	MV	3042.15 ± 11.4	3099.25 ± 10.4	3075.85 ± 9.2	3056.95 ± 7.7 (-, +, +)	3050.0 ± 5.8 (=, +, +, =)
	RL	14.08 ± 3.4	3.58 ± 0.9	3.44 ± 0.2	3.31 ± 1.0 (+, =, +)	3.22 ± 1.2 (+, +, +, +)
	GD	88.61 ± 6.7	102.36 ± 12.8	155.44 ± 3.7	143.47 ± 4.9 (+, +, -)	173.65 ± 9.1 (+, +, =, +)
	PD	28.48 ± 4.4	60.68 ± 8.2	94.49 ± 4.8	95.21 ± 5.3 (+, +, +)	95.21 ± 6.9 (+, +, =, =)
ta67	MV	3024.95 ± 19.8	3108.25 ± 18.5	3084.15 ± 16.2	3040.95 ± 10.1 (=, +, +)	3039.6 ± 11.6 (=, +, +, =)
	RL	13.87 ± 2.2	3.35 ± 0.7	3.51 ± 0.2	3.68 ± 1.3 (+, -, -)	3.49 ± 1.2 (+, -, +, +)
	GD	91.13 ± 5.5	109.15 ± 9.6	157.55 ± 3.1	143.85 ± 4.7 (+, +, -)	180.33 ± 7.0 (+, +, +, +)
	PD	32.65 ± 5.7	66.22 ± 6.1	96.73 ± 4.5	96.25 ± 5.7 (+, +, =)	104.62 ± 6.2 (+, +, =, =)
ta68	MV	2922.95	2985.15	2946.65	2936.35 ± 14.6	2922.1 ± 10.9

		± 11.9	± 15.8	± 8.9	(=, +, +)	(=, +, +, =)
	RL	11.57 ± 2.0	3.24 ± 0.7	3.4 ± 0.2	3.21 ± 0.8 (+, +, =)	2.76 ± 0.6 (+, +, +, +)
	GD	94.16 ± 7.3	91.48 ± 13.9	153.12 ± 3.5	143.29 ± 5.0 (+, +, -)	166.0 ± 12.3 (+, +, =, +)
	PD	25.19 ± 4.7	48.27 ± 8.1	83.4 ± 4.4	85.65 ± 7.1 (+, +, +)	79.62 ± 7.9 (+, +, =, =)
ta69	MV	3211.35 ± 6.6	3275.55 ± 14.0	3268.9 ± 11.7	3234.3 ± 12.3 (-, +, +)	3221.15 ± 9.0 (-, +, +, =)
	RL	13.92 ± 3.5	4.62 ± 2.0	3.28 ± 0.2	3.23 ± 1.1 (+, +, +)	3.11 ± 0.8 (+, +, +, +)
	GD	91.14 ± 5.6	111.56 ± 10.2	155.88 ± 3.5	140.8 ± 7.4 (+, +, -)	183.13 ± 8.1 (+, +, +, +)
	PD	28.12 ± 4.8	54.83 ± 6.7	74.94 ± 5.3	87.23 ± 6.0 (+, +, +)	94.44 ± 6.8 (+, +, +, +)
ta70	MV	3278.55 ± 21.8	3334.65 ± 16.9	3301.55 ± 16.5	3289.55 ± 16.7 (=, +, =)	3258.85 ± 12.4 (=, +, +, +)
	RL	13.82 ± 2.4	3.51 ± 0.8	3.46 ± 0.2	3.59 ± 1.3 (+, -, -)	3.29 ± 1.1 (+, =, =, +)
	GD	93.14 ± 6.0	106.65 ± 11.6	154.47 ± 3.9	146.4 ± 4.7 (+, +, -)	169.31 ± 10.2 (+, +, =, +)
	PD	28.55 ± 5.0	63.56 ± 7.5	86.05 ± 4.9	92.52 ± 6.5 (+, +, +)	90.38 ± 7.5 (+, +, =, -)
Sum.	MV	0-4, 0-1	10-0, 10-0	9-0, 10-0	2-0	None
	RL	8-1, 8-1	3-2, 6-1	4-4, 6-0	9-0	

GD	10-0, 10-0	10-0, 10-0	0-10, 6-1	9-0
PD	10-0, 10-0	10-0, 10-0	9-0, 4-0	1-3

3.4.3 Mean Tardiness Objective

The five algorithms were evaluated in terms of their ability to optimize the Mean Tardiness (MT) objective, which evaluates the ability of the job shop to meet customer due dates. The results are presented in Table 3.3. The performance of the PGP_P and PGP_N algorithms was superior to that of the SGP algorithm with respect to the RL, GD, and PD objectives. Regarding the MT objective, both SGP and PGP_N had relatively similar performances, except for one instance in which SGP outperformed PGP_N. Compared with the EGP algorithm, the proposed algorithms achieved results that are of a significantly higher quality for the MT, GD, and PD objectives in all considered instances. The conclusion regarding the average length of the evolved rules is not definitive because the PGP_P algorithm delivered superior performance in three instances and inferior performance in four instances compared with the EGP algorithm.

The results in Table 3.3 indicate that the proposed algorithms were able to create rules with higher solution quality and smaller sizes than the MGP algorithm. Specifically, PGP_P and PGP_N have significantly lower mean tardiness values than the MGP algorithm in nine and ten instances, respectively. In terms of the phenotypic diversity of the evolved rules, both PGP_P and PGP_N outperformed MGP in nine and two instances, respectively, without any loss. In contrast, the PGP_P algorithm yielded poor GD results compared with the MGP results for the ten scenarios. In addition, it is important to note that high genotypic diversity does not necessarily imply high phenotypic diversity because rules with different structures and content may obtain the same fitness values. The main reason for this phenomenon is the presence of redundant operations, for example, even though the $WINQ + PT$ rule and the $WINQ + ((PT \times NPT)/NPT)$ rule have different structures and terminals (genomes), they will return the same results (phonemes). Finally, the rules generated using the proposed multi-objective approach had higher solution quality in two instances, shorter lengths in seven instances, and higher genotypic diversity in nine instances compared with the PGP_P algorithm.

Although the results obtained using the mean tardiness objective were consistent with those obtained using the makespan objective, two additional observations were noted. First, in most scenarios, the average rule length in the case of the mean tardiness objective is greater than the average length of the rules generated for the makespan objective. Second, the PD measure in the SGP algorithm did not change significantly between the two objectives; in spite of this, the other algorithms had higher PD levels in most instances in the case of the mean tardiness objective compared with the makespan objective. Increasing the number of specified terminals related to job due dates by three in the case of the mean tardiness objective might be the reason for the increase in the average rule length and phenotypic diversity. When the number of terminals increases, the heuristic search space increases exponentially; thus, the GP algorithm increases the length of the generated rules to improve the exploration ability. In addition, as the heuristic search space and average length of rules increase, distance metrics become more efficient because the possibility of variability between the structure and fitness value of evolving rules increases.

Table 3.3 Performance of the five GP algorithms in terms of optimizing the mean tardiness objective on the ten JSSP instances

Inst.	Perf.	SGP	EGP	MGP	PGP_P	PGP_N
ta61	MT	492.14 ± 5.0	514.54 ± 5.4	507.98 ± 5.5	498.37 ± 4.8 (=, +, +)	494.0 ± 3.4 (=, +, +, =)
	RL	14.56 ± 3.6	4.62 ± 1.4	4.1 ± 0.2	3.71 ± 0.7 (+, +, +)	3.59 ± 0.8 (+, +, +, +)
	GD	94.25 ± 5.5	113.1 ± 12.5	155.83 ± 3.1	142.63 ± 5.0 (+, +, -)	168.69 ± 9.6 (+, +, =, +)
	PD	25.27 ± 3.6	74.33 ± 9.6	97.28 ± 4.6	103.32 ± 6.4 (+, +, =)	100.04 ± 5.5 (+, +, =, -)
ta62	MT	463.19 ± 7.6	485.07 ± 4.9	478.94 ± 4.3	470.15 ± 4.3 (=, +, +)	466.72 ± 4.2 (=, +, +, =)
	RL	12.21 ± 1.8	4.08 ± 1.0	4.17 ± 0.2	4.11 ± 1.0 (+, -, +)	3.68 ± 0.7 (+, +, +, +)

	GD	89.34 ± 6.4	117.38 ± 10.4	158.15 ± 3.3	145.08 ± 5.0 (+, +, -)	182.13 ± 8.1 (+, +, +, +)
	PD	27.96 ± 4.4	88.5 ± 8.0	112.89 ± 4.9	121.24 ± 6.9 (+, +, +)	128.05 ± 7.3 (+, +, =, =)
ta63	MT	461.61 ± 4.0	486.55 ± 5.5	475.17 ± 3.1	468.75 ± 4.7 (-, +, +)	466.37 ± 4.1 (-, +, +, =)
	RL	13.57 ± 2.3	4.11 ± 1.4	4.18 ± 0.2	3.9 ± 0.9 (+, =, =)	3.99 ± 1.0 (+, =, =, =)
	GD	91.78 ± 7.5	106.42 ± 11.4	157.02 ± 3.5	143.86 ± 5.0 (+, +, -)	179.58 ± 8.8 (+, +, +, +)
	PD	26.65 ± 4.0	79.03 ± 8.7	105.7 ± 4.7	116.22 ± 6.3 (+, +, +)	125.12 ± 7.6 (+, +, =, =)
ta64	MT	455.9 ± 7.3	479.64 ± 7.0	467.02 ± 6.5	457.33 ± 6.0 (=, +, +)	447.49 ± 7.0 (=, +, +, +)
	RL	14.13 ± 3.3	3.48 ± 0.6	4.11 ± 0.2	3.83 ± 1.0 (+, -, =)	3.88 ± 1.1 (+, -, =, =)
	GD	95.07 ± 6.6	99.75 ± 11.5	157.64 ± 3.1	145.97 ± 4.8 (+, +, -)	175.15 ± 8.7 (+, +, +, +)
	PD	24.27 ± 4.2	68.86 ± 7.2	94.83 ± 5.1	115.77 ± 6.6 (+, +, +)	112.81 ± 6.1 (+, +, +, -)
ta65	MT	443.85 ± 5.9	459.51 ± 5.9	454.9 ± 4.8	449.91 ± 3.7 (-, +, =)	446.21 ± 4.9 (=, +, +, =)
	RL	13.34 ± 2.1	3.8 ± 0.7	4.09 ± 0.2	3.67 ± 0.8 (+, +, +)	3.51 ± 0.5 (+, +, +, +)
	GD	92.07 ± 6.1	114.4 ± 9.5	157.6 ± 3.1	144.45 ± 4.7 (+, +, -)	179.33 ± 9.8 (+, +, +, +)

	PD	25.05 ± 3.5	81.4 ± 7.6	102.43 ± 5.3	116.4 ± 6.0 (+, +, +)	123.65 ± 7.7 (+, +, =, =)
ta66	MT	470.19 ± 7.4	499.12 ± 6.2	490.54 ± 4.9	476.14 ± 6.0 (=, +, +)	475.51 ± 6.0 (=, +, +, =)
	RL	13.97 ± 2.8	3.73 ± 1.0	3.91 ± 0.2	3.53 ± 1.0 (+, =, +)	3.16 ± 0.8 (+, +, +, +)
	GD	90.77 ± 7.1	96.24 ± 10.0	153.67 ± 3.6	141.62 ± 5.9 (+, +, -)	143.73 ± 13.2 (+, +, =, =)
	PD	24.28 ± 3.5	62.96 ± 7.0	87.22 ± 4.4	110.23 ± 6.6 (+, +, +)	90.11 ± 7.4 (+, +, =, -)
ta67	MT	488.62 ± 6.6	515.9 ± 7.1	502.96 ± 3.9	494.27 ± 3.3 (=, +, +)	491.38 ± 4.3 (=, +, +, =)
	RL	15.24 ± 4.1	3.73 ± 0.7	4.19 ± 0.2	4.15 ± 1.3 (+, -, +)	3.72 ± 1.0 (+, =, +, +)
	GD	90.96 ± 6.8	107.05 ± 9.8	155.82 ± 3.5	144.88 ± 4.9 (+, +, -)	168.02 ± 9.0 (+, +, =, +)
	PD	27.3 ± 4.0	79.58 ± 7.7	97.8 ± 5.4	119.75 ± 6.3 (+, +, +)	114.54 ± 7.1 (+, +, =, -)
ta68	MT	476.26 ± 7.6	496.38 ± 6.5	491.58 ± 4.5	481.2 ± 6.6 (=, +, +)	475.53 ± 5.8 (=, +, +, =)
	RL	14.24 ± 3.1	3.85 ± 0.8	4.08 ± 0.2	3.82 ± 1.1 (+, +, =)	4.0 ± 1.1 (+, -, =, =)
	GD	91.7 ± 6.9	103.03 ± 12.6	155.09 ± 4.0	144.81 ± 4.5 (+, +, -)	173.99 ± 8.6 (+, +, +, +)
	PD	28.31 ± 3.9	72.94 ± 8.9	99.39 ± 6.0	120.84 ± 6.9 (+, +, +)	123.42 ± 7.3 (+, +, +, =)

ta69	MT	523.12 ± 8.2	546.72 ± 5.6	538.03 ± 5.6	527.85 ± 7.4 (=, +, +)	517.62 ± 4.3 (=, +, +, +)
	RL	11.62 ± 1.7	3.4 ± 0.5	3.98 ± 0.2	3.56 ± 0.8 (+, -, +)	3.46 ± 0.9 (+, -, +, +)
	GD	92.87 ± 5.6	96.1 ± 10.3	155.46 ± 3.1	142.01 ± 5.0 (+, +, -)	163.64 ± 11.9 (+, +, =, +)
	PD	23.76 ± 3.0	65.77 ± 6.5	87.13 ± 6.4	101.75 ± 7.5 (+, +, +)	99.83 ± 5.7 (+, +, =, -)
ta70	MT	505.36 ± 6.0	526.77 ± 5.5	521.24 ± 3.4	514.11 ± 6.3 (-, +, +)	508.56 ± 5.1 (=, +, +, =)
	RL	12.92 ± 2.1	4.11 ± 0.9	4.15 ± 0.2	3.79 ± 0.9 (+, =, =)	3.39 ± 0.6 (+, +, +, +)
	GD	98.77 ± 7.2	110.45 ± 13.1	158.48 ± 3.0	141.31 ± 5.2 (+, +, -)	181.76 ± 8.7 (+, +, +, +)
	PD	23.54 ± 3.8	71.94 ± 9.5	99.15 ± 5.1	107.57 ± 7.7 (+, +, +)	113.43 ± 7.3 (+, +, =, =)
Sum.	MT	0-3, 0-1	10-0, 10-0	9-0, 10-0	2-0	None
	RL	10-0, 10-0	3-4, 5-3	6-0, 7-0	7-0	
	GD	10-0, 10-0	10-0, 10-0	0-10, 6-0	9-1	
	PD	10-0, 10-0	10-0, 10-0	9-0, 2-0	0-5	

3.5 Chapter Summary

This chapter proposed a distance metric to promote diversity among the scheduling heuristics evolved using a genetic programming algorithm. The proposed distance metric took into account four main characteristics of GP rules observed by behaviour analysis. In addition, to mitigate the bloating effect, the proposed metric was integrated with NSGA-II to optimize the solution quality, diversity value, and rule length simultaneously. Two algorithms, PGP_P and PGP_N, were developed to assess the effectiveness of the proposed distance metric and multi-objective GP approaches. In addition, two objective functions were addressed: the makespan and mean tardiness. For each objective, four performance measures were evaluated: the objective value, genotypic diversity, phenotypic diversity, and average length of the evolved rules.

The impact of the newly introduced parameter k was analysed by tracking the performance of the proposed algorithm across evolutionary generations using several k values $k = \{1, 2, 3, 4\}$ and the four performance measures. Afterward, the performance of the two proposed algorithms was compared with that of three algorithms from the literature namely, SGP, EGP, and MGP across ten benchmark job shop scheduling problem instances. Regarding the literature methods, experimental results indicated that the edit distance metric used in the EGP algorithm was not effective for measuring the similarity between the GP evolved rules. In contrast, measuring similarity using the genetic marker metric used in the MGP algorithm enhanced both genotypic and phenotypic diversity among the GP rules. In addition, the obtained results demonstrated the effectiveness of the proposed methods in generating a phenotypically diverse population of scheduling rules with smaller sizes and higher solution quality compared with other methods. Although one of the existing diversity metrics (MGP) obtained higher quality genotypic diversity results compared with the PGP_P algorithm, the evolved rules using PGP_P significantly outperformed the MGP rules in terms of the other three objectives. The rules generated using the PGP_N algorithm produced superior results compared with the literature approaches for the studied objectives. Finally, it was noted that the average rule length in the case of the mean tardiness objective was greater than that in the case of the makespan objective. This was expected since the number of features in the mean tardiness objective had three more features in the terminal set compared with the makespan objective. Reducing the number of features in the terminal set in order to reduce the size of evolved rules is the motivation behind the next chapter.

Chapter 4. GENETIC PROGRAMMING WITH FEATURE SELECTION FOR DYNAMIC JOB SHOP SCHEDULING PROBLEMS

4.1 Introduction

The ability of the GP algorithm to generate high-quality rules depends largely on the features included in the terminal set that need to cover the most crucial characteristics of the job, machine, and shop floor (F. Zhang et al., 2021c). However, there is a wide range of features to choose from which varies depending on job shop settings, problem constraints, and objective functions making the manual selection of features impractical (Zhang et al., 2019c). In addition, the inclusion of insignificant terminals leads to three major issues as follows (Mei et al., 2017b).

a) It greatly increases the search space for dispatching rules and thus negatively affects the ability of the GP algorithm to reach the most promising areas.
b) The average size of evolved rules tends to be large which hinders their understanding and implementation in real-world problems.
c) It increases the GP computational time greatly because complex rules require a high computational budget for a fitness assessment compared with simpler rules.

Therefore, feature selection is an important issue in the GP literature which can simplify the evolved rules and speed up the learning process (F. Zhang et al., 2021c). Although GP can perform feature selection automatically, its ability is limited. For example, even the best rules usually include some redundant terminals (Zhang et al., 2019c).

The main objective of this chapter is to propose a new feature selection approach for the GP algorithm to include significant features and exclude redundant ones from the structure of evolved rules. In contrast with existing feature selection approaches in the literature, the proposed approach is expected to offer the following advantages.

I. The proposed approach uses a modified attribute vector representation to estimate the weight of each terminal without being affected by the occurrence of redundant terminals or complex rule structures.
II. It is an online feature selection approach which means that it selects important features during the GP runs using the estimated weights of terminals from earlier generations to guide the search in the current generation.
III. It uses a probabilistic selection scheme rather than the inclusion or exclusion method (binary selection) to provide a broad preference scheme for each feature.

The remainder of this chapter is organized as follows. Section 4.2 provides a detailed explanation of the proposed attribute vector, and feature selection approach. The experimental details are presented in Section 4.3 including comparison design, dynamic job shop scheduling problem instances, and GP parameters. Section 4.4 provides the results in terms of parameter tuning, training performance, testing performance, feature and behaviour analysis of evolved rules, and feature selection verification. Finally, Section 4.5 presents the conclusions of this chapter.

4.2 Proposed methods

4.2.1 Modified Attribute Vector

The GP-evolved rules are usually complex tree structures that are difficult to analyse and interpret. The authors of one study (Nguyen et al., 2018b) extended the tree representation ($ptree_i$) for each rule i by an attribute vector ($avec_i$) with the goal of increasing the interpretability of the rules by selecting relevant attributes. The attribute vectors are binary arrays $(1, 0)$ with a number of elements T equal to the total number of terminals. If the terminal state x_{ij} of a terminal j in rule i is 1 (active or important), then the actual value of the attribute is used to evaluate the priority function. In contrast, if x_{ij} is 0, then the attribute is regarded as irrelevant (inactive). Its value is set to 1 to exclude its effect on estimating priority values for queued jobs. The key limitation is that it ignores situations in which a particular attribute might not be present in the

priority function. Therefore, the attribute vector is not strictly linked to its corresponding priority function, leading to the following challenges.

- Attribute vectors do not provide sufficient information about their priority functions. Some examples are that elements in an attribute vector might take a value of 1 (active) even if the rule includes only one terminal.
- Mutation operators in attribute vectors may not always provide a true influence on how evolved rules estimate priority values (redundant operations). In other words, changing the activation state of a terminal has no effect if this terminal is not present in the rule. However, it might exert a future effect if any of these terminals emerge after tree crossover or mutation, although this is not certain.
- The attribute vector mutation operator is applied to only one random terminal with a fixed mutation rate that ignores the relative importance of the terminals.

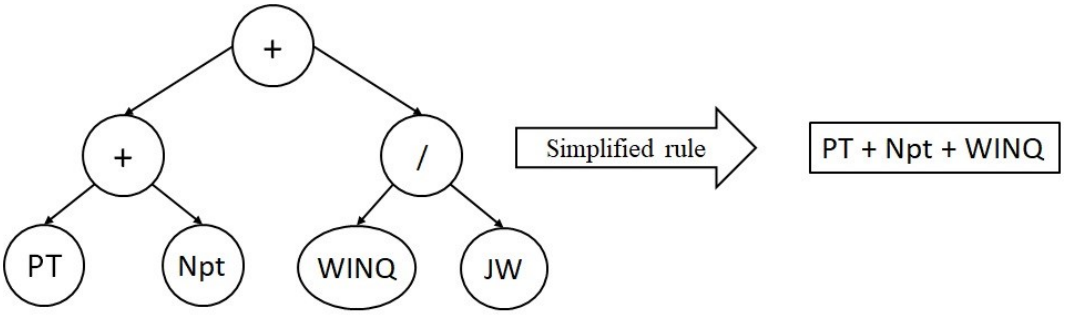
Therefore, a new attribute vector representation is proposed by modifying the representation proposed in that earlier paper (Nguyen et al., 2018b). The proposed attribute vector extends the binary representation, where the terminal state x_{ij} of rule i and terminal j can be $active = 1$ or $inactive = 0$ to a ternary array where three states exist for each terminal. If terminal j appears in the priority function of rule i, then its state x_{ij} may be $active = 1$ or $inactive = -1$; otherwise, it is $absent$ with a state equal to 0. For tree-based GP, priority functions are generated using a predefined set of functions and terminals. The set of terminals and functions used is shown in Table 4.1.

Table 4.1 GP terminal and function sets

Terminal	**Description**	**Terminal**	**Description**
WINQ	Work in the next queue	DD	Due date of the job
OR	Ready time of the operation	CT	Current time
RO	Number of remaining operations	SL	Slack of the job
PT	Operation processing time	JR	Release date of a job
WT	Waiting time of the operation	WR	Work reaming of the job
Npt	Processing time of next operation	JW	Weight of the job
Apr	Average processing time of queued jobs	#	Random number from 0 to 4
Functions	$+, -, \times, /, min, max, abs$		

An example of a dispatching rule encoded using the proposed representation and the literature representation is presented in Figure 4.1. Although both representations have the same priority function ($ptree_i$) for the function PT + Npt + WINQ / JW and the similar simplified version (PT + Npt + WINQ), the main difference appears in the attribute vectors ($avec_i$). In the case of literature representation, there are ten active attributes, although only three of them exist in the rule. In addition, the WT, SL, and JW terminals are encoded in the same way (inactive), even though JW is presented in the priority function. In contrast, by using the proposed representation, one can readily distinguish between active terminals (PT, Npt, and WINQ), inactive terminals (JW), and absent terminals (remaining terminals). This concise abstraction engenders two main advantages over the literature representation.

- It supports attribute vectors to be linked precisely to the structure of their corresponding priority functions. Therefore, attribute vectors are useful to gain useful insights into the structure of complex priority functions.
- It ensures that any change in the feature's state will have a direct effect on the rule. This additional capability facilitates the feature selection mechanism presented in the next subsection.



Representation	JR	OR	RO	WR	PT	DD	CT	SL	WT	Npt	WINQ	Apr	JW
Literature	1	1	1	1	1	1	1	0	0	1	1	1	0
Proposed	0	0	0	0	1	0	0	0	0	1	1	0	-1

Figure 4.1: Example of a rule in the literature and proposed representations.

4.2.2 Genetic operators and Feature Selection Approach

Genetic operators are applied in a two-step procedure. In the first step, the standard subtree crossover and mutation operators are applied to priority functions (J. R. Koza, 1994b). The second step represents the proposed feature selection mechanism as follows.

a) A subset $\mathbb{S}$ of the best-selected rules from the current generation is used to estimate the weights of the terminals in the next generation. The weight of a terminal j is reflected by its activation probability AP_j in the attribute vector.
b) At the end of each evolutionary step, the weight of each terminal AP_j is estimated as shown in Equation 4.1.
c) Attribute vectors are copied from the parents. Another mutation is applied using Equation 4.2. For each rule i, in the absence of a certain terminal j the value of x_{ij} in $avec_i$ will be 0.
d) Conversely, if terminal j is presented, then a uniform random number ($rand$) between 0 and 1 is generated. Two situations can occur. If $rand$ is less than or equal to its activation probability AP_j, then x_{ij} is 1 (active), otherwise, the terminal x_{ij} is inactive and takes a value of -1.

The idea underlying this approach is that, if a particular terminal is active in most of the selected rules, then it reflects its great weight. Therefore, the GP algorithm will be directed to use that terminal heavily in the next generation. In other words, if the activation probability of a terminal equals 1 (very important), then it will be active in all the dispatching rules containing this terminal in the next generation. In contrast, if the activation probability of a terminal is 0 (greatly irrelevant), then it will be inactive in the next generation even if it is present in the priority function of some rules. Therefore, the activation probability acts as an on-the-fly feature selection mechanism that uses past evolutionary information to estimate the importance of each terminal based on its effect on the best-generated rules. Finally, to address a special case in which a certain terminal did not occur or in which it was inactive in all selected rules that rarely happen, a fixed activation probability (revive) of 0.05 is used. The motive is to activate this terminal in only 5% of newly generated rules because it might have a tangible effect on other dispatching rules with different structures. Table 4.2 presents an illustrative example of how terminals' weights are evaluated using attribute vectors of five arbitrary dispatching rules.

$$AP_j = \frac{\sum_{i=1}^{|\mathbb{S}|} 1\,[x_{ij} = 1]}{\sum_{i=1}^{|\mathbb{S}|} 1\,[x_{ij} = 1] + \sum_{i=1}^{|\mathbb{S}|} 1\,[x_{ij} = -1]} \tag{4.1}$$

$$x_{ij} = \begin{cases} 0, & terminal\ j\ does\ not\ occur\ in\ rule\ i \\ 1, & rand \leq AP_j \\ -1, & rand > AP_j \end{cases} \tag{4.2}$$

Table 4.2 An example for estimating terminals' weights using five attribute vectors

Rules	JR	OR	RO	WR	PT	DD	CT	SL	WT	Npt	WINQ	Apr	JW
Rule 1	1	1	1	-1	1	1	1	0	0	1	1	0	0
Rule 2	0	0	0	0	1	0	0	0	0	1	1	0	-1
Rule 3	-1	0	1	-1	-1	1	1	1	1	0	1	0	0
Rule 4	1	1	-1	-1	0	-1	0	1	-1	-1	1	0	-1
Rule 5	1	0	-1	1	-1	-1	1	1	1	1	1	0	-1
Weight	0.75	1	0.5	0.25	0.5	0.5	1	1	0.67	0.75	1	0.05	0.05

4.2.3 Overall Algorithm Framework

The proposed approach including the novel representation and feature selection is presented as Algorithm 1 in Figure 4.2. The algorithm starts by initialising a random population of dispatching rules. Each rule (R_i) is represented by two parts: the priority function in the tree structure ($ptree_i$) and an attribute vector ($avec_i$). The activation probabilities array AP is initialised with the same initial probability $Prob_{int}$ for all terminals. As suggested in an earlier paper (Zhou and Yang, 2019), the same random seed is used for all individuals in the same generation, whereas the seed is changed between generations to avoid overfitting to a specific problem instance. A DES model is developed to evaluate the steady-state performance of scheduling policies. All the generated rules are evaluated across a set of predefined training instances representing different job shop settings. If the fitness of a certain rule is smaller than (in the case of minimisation problems) the fitness of the best-recorded individual, then the best rule and its fitness value are updated as shown in steps 7–13.

The mating pool is formed by high-quality rules that are chosen from the current population using the tournament selection method. Moreover, a subset $\mathbb{S}$ of the selected rules is used to update features' weights in the activation probability array. The standard tree crossover and mutation operators are used. In addition, the proposed adaptive mutation operator is applied to the attribute vectors of the newly generated rules as described in steps 18–28. If the stopping criterion is met, then the algorithm terminates and the best rule is returned; otherwise, another evolutionary iteration begins by following the same steps.

Inputs: training simulation scenarios $O \leftarrow \{O_1, O_2, \dots, O_N\}$

Output: the best evolved rule R_{best}

1: Initialize population $P_1 \leftarrow \{R_1, R_2, \dots, R_n\}$,

$$R_i \leftarrow \{ptree_i, avec_i\}, avec_i \leftarrow \{x_{i1}, x_{i2}, \dots, x_{iT}\}$$

2. Initialize activation probabilities array $AP \leftarrow \{T\ values\ equal\ 0 < Prob_{int} \leq 1\}$

3: Set $R_{best} \leftarrow null$ and the best fitness value $f(R_{best}) \leftarrow +\infty$

4: $gen \leftarrow 1$

5: **while** $gen \leq \max generation$ **do**

6: reset the random seed

7: **for all** $R_i \in P_{gen}$ **do**

8: evaluate $f(R_i)$ by applying R_i to each scenario $O_k \in O$

9: **if** $f(R_i) < f(R_{best})$ **then**

10: $R_{best} \leftarrow R_i$

11: $f(R_{best}) \leftarrow f(R_i)$

12: **end if**

13**:** **end for**

14: select the best |P| individuals of P_{gen} to join mating pool

15: estimate AP using a subset | 𝕊 | of the best rules

16: **for all** $R_i \in P_{gen}$ **do**

17: apply genetic operators on $ptree_i$

18: **for all** $x_{ij} \in avec_i$ **do**

19: **if** terminal j not in $ptree_i$ **then**

20: $x_{ij} \leftarrow 0$

21: **else**

22: **if** $rand \leq AP_j$ **then**

23: $x_{ij} \leftarrow 1$

24: **else**

25: $x_{ij} \leftarrow -1$

26: **end if**

27: **end if**

28: **end for**

29: **end for**

30: $gen \leftarrow gen + 1$

31: **end while**

32: **return** R_{best}

Figure 4.2: The proposed genetic programming algorithm

4.3 Experiment Design

To investigate the effectiveness of the proposed approach compared with other methods that have been reported in the literature, a set of numerical experiments was performed.

4.3.1 Comparison Design

Four GP approaches are considered for evolving scheduling policies in the DJSSP. The overall algorithm framework is shown in Figure 4.3. The exclusive operations of the proposed approach are highlighted with a dashed frame. Three GP algorithms are adopted from the literature including Standard Genetic Programming algorithm (SGP) (Geiger et al., 2006b), Non-dominated Sorting Genetic Programming (NSGP) (Hunt et al., 2016b), Hybrid Genetic Programming (HGP) (Nguyen et al., 2018b) to provide a detailed comparison between the Proposed Genetic Programming (PGP) approach and the current methods. The SGP algorithm is regarded as evaluating the usefulness of extending the standard version of the GP algorithm with the proposed new representation and feature selection capability. In addition, the HGP algorithm is developed to check whether the modification proposed in the current representation enhances the GP algorithm in evolving shorter dispatching rules without sacrificing solution quality. From one earlier study (Hunt et al., 2016c), the authors suggested integrating GP and a multi-objective algorithm to improve solution quality and rule

length simultaneously. In addition, the integration between GP and NSGA-II, denoted as NSGP, obtained dispatching rules with better performance than that of SPEA2 for multi-objective DJSSP (Nguyen et al., 2015b). Therefore, in order to ascertain whether it is adequate to consider rule length as an explicit objective along with solution quality rather than an implicit consideration that is achieved using the proposed framework. Therefore, the NSGP algorithm is adopted to optimise both solution quality and rule length in an explicit fashion.

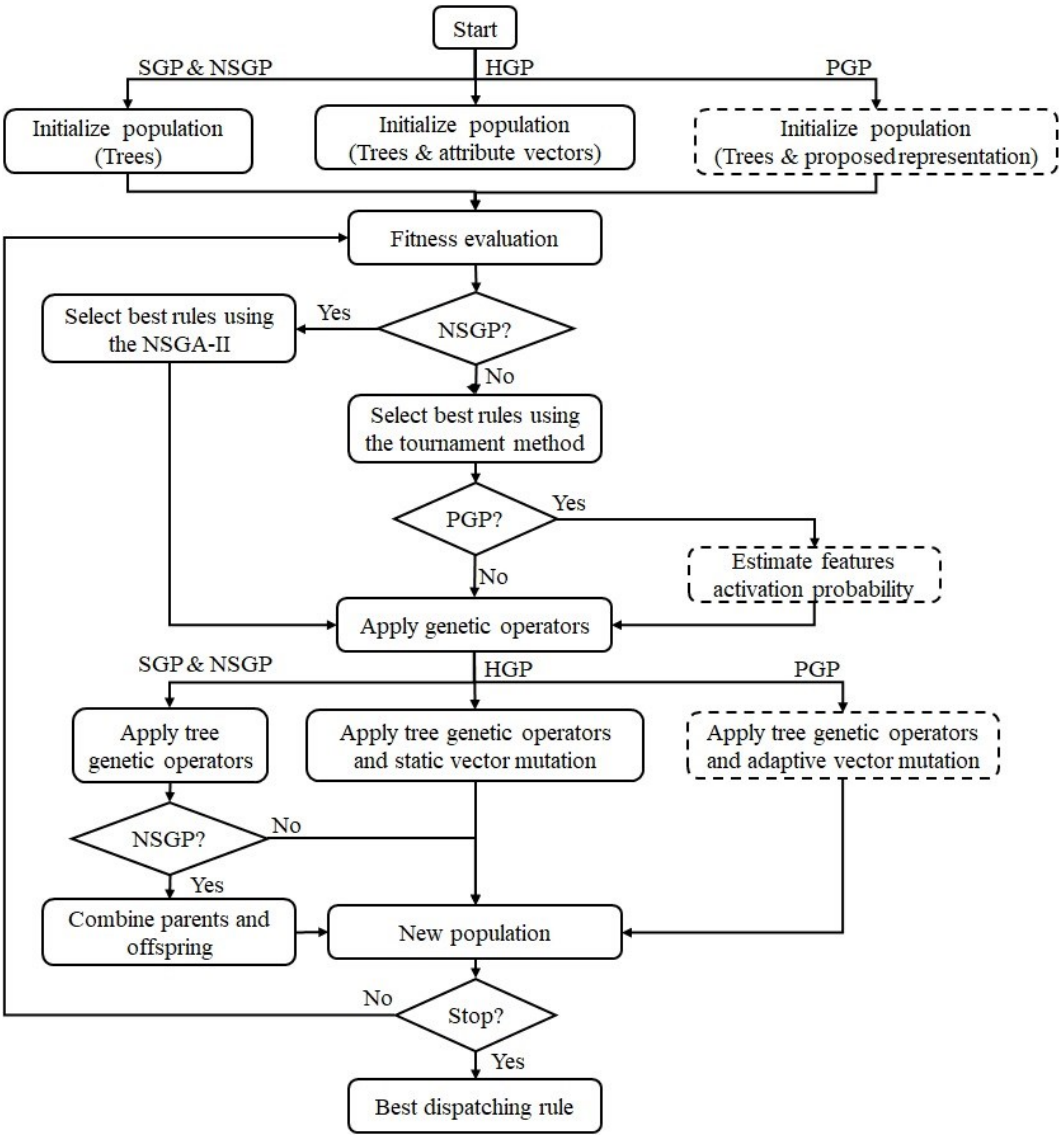


Figure 4.3: Framework of the four algorithmic experiments

The four algorithms start by initialising a population of rules in the tree structure. For the case of HGP, attribute vectors are initialised using the representation shown by Nguyen et al. (Nguyen et al., 2018b), whereas PGP uses the modified representation presented herein. In the case of the SGP, HGP, and PGP algorithms, the solution quality of evolved rules is evaluated. The best rules are chosen using a tournament method

according to their fitness values for reproduction. For the NSGP algorithm, NSGA-II (Deb et al., 2002b) is used as a selection mechanism to assign ranks and crowding distance to each individual. Individuals with higher ranks and smaller crowding distances are selected as parents. In addition, the fitness value for each rule is expressed using both the rule length and solution quality, unlike the other three algorithms in which the fitness value represents only the solution quality. Afterward, in the case of the PGP method, the features' weights are updated in the activation probability array.

For all algorithms, generic operators are conducted on priority functions to create offspring. The additional mutation operator is used in the HGP algorithm, which selects a single attribute in the attribute vector randomly and inverts its state. In the case of PGP, attribute vector mutations are applied using activation probabilities estimated from the prior generation. The new population is formed in the case of the NSGP by combining both parents and offspring to ensure elitism, as recommended by Deb et al. (Deb et al., 2002b). The evolutionary cycle is repeated for a predefined number of generations. Finally, the algorithm terminates. The best-evolved rule is obtained. For the sake of comparison, 30 literature dispatching rules are adopted to verify the superiority of the GP reasoning mechanism in outperforming the standard rules commonly used in industry. The chosen rules are shown in Table 4.3. These rules have obtained high-solution quality in accordance with the considered objectives in previous studies (Nguyen et al., 2013c; Zhou et al., 2020b).

Table 4.3 Benchmark dispatching rules

Rule	Description	Rule	Description
SPT	Shortest processing time	SL	Slack
LPT	Longest processing time	PW	Process waiting time
EDD	Earliest due date	WATC	Weighted apparent tardiness cost
FDD	Earliest flow due date	COVERT	Cost over time
FIFO	First in first out	OPFSLK/PT	Operation flow slack / processing time
LIFO	Last in first out	LWKR + SPT	Least work remaining + processing time

LWKR	Least work remaining	CR + SPT	Critical ratio + processing time
MWKR	Most work remaining	SPT + PW	Processing time + waiting time
NPT	Next processing time	SPT+PW+FDD	Processing time + PW + FDD
WINQ	Work in next queue	SL / MOR	Slack per most operation remaining
CR	Critical ratio	SL / LWKR	Slack per least work remaining
AVPRO	Average processing time / operation	PT+WINQ	Processing time + WINQ
MOD	Modified due date	2*PT+WINQ+Npt	2Processing time + WINQ + next processing time
MOR	Most operation remaining	PT+WINQ+SL	Processing time + WINQ + SL
NSL	Negative slack	2PT+WINQ+ NPT + WSL	2Processing time + WINQ + next processing time + waiting slack

4.3.2 Dynamic Job Shop Simulation Model

A simulation model of a symmetrical job shop was developed that was considered in relevant earlier studies (Hildebrandt et al., 2010b; Shady et al., 2020c). The simulation configurations are the following.

- Jobs arrive stochastically according to a Poisson distribution and at a rate that engenders a predetermined utilisation level in the job shop.
- The job shop consists of 10 machines.
- Each job has 2–10 operations.
- Processing times follow a uniform distribution U [1, 49].
- Weights of jobs are assigned based on a 4:2:1 rule (Nguyen et al., 2013c).
- A tightness factor is used to estimate due dates using the total work content method. $Due\ date = current\ time + tightness\ factor \times total\ processing\ time.$

At each simulation replication, the job shop starts empty. All the collected data up to the 500th job are discarded. Statistics from the 501st job to the next finished 2500 jobs are used to calculate the performance measures. Three objective functions are investigated including Total Weighted Tardiness (TWT), Mean Tardiness (MT), and

Mean Flow Time (MFT). The TWT objective is chosen to assess not only the ability to meet due dates, but also how to prioritise jobs with higher weights. In addition, the MT objective is used to estimate the average delay, which indicates the level of customer satisfaction. Finally, the MFT objective is used to verify the adaptability of the proposed method in cases where increased throughput is the desired goal.

Generally, all hyper-heuristics including the GP approach generate new heuristics by gathering reusable knowledge from a set of training instances either in a supervised or unsupervised manner (Nguyen et al., 2017b). Therefore, defining a set of scenarios that reflect the problem domain that the heuristics are likely to encounter in their future use is a critically important step. Two factors are examined when selecting training cases including the training set size and computational time, as recommended in an earlier paper (Branke et al., 2016b). If a small training set is chosen, then the generated heuristics are likely to suffer from overfitting leading to poor performance in unseen scenarios. However, a large training set increases the runtime of the heuristics' evaluation phase without ensuring better results. Training and testing scenarios are shown in Table 4.4. Simulation scenarios are denoted by a tuple (u, t) to represent a combination of job shop utilisation $u\%$ and tightness factor t. It is noteworthy that, in the case of the MFT objective, the tightness factor of 3 is set in all scenarios because changing the value of the tightness factor does not affect the job flow time. In addition, in the MT scenarios, tighter due dates are used to estimate the quality of generated heuristics under more challenging scenarios compared with the job shop settings used for the TWT objective. In the training stage, a single simulation replication is executed for each configuration. In the testing stage, 20 simulation replications are performed for each scenario. The objective value $(obj_{i,n})$ of rule i in an instance n is estimated using the developed DES model. Moreover, for each objective, a reference rule (ref) is chosen to normalise the obtained results. The overall performance of a rule $(fitness_i)$ is assessed through a set N_o of training instances for a specific objective o, as shown in Equation 4.3. The WATC, Covert, and PT + WINQ rules are used respectively as reference rules to minimise the TWT, MT, and MFT objectives. These rules are chosen because they have yielded superior results for the objectives under study (Mei et al., 2017c; Sels et al., 2012b; Zhou et al., 2019). Finally, the percentage change PC_o in the performance of a given method to a reference rule for an objective function o is estimated using Equation 4.4 where $obj_{best,n}$ denotes the best evolved rule.

$$fitness_i = \frac{1}{|N_o|}\sum_{n=1}^{|N_o|}\frac{obj_{i,n}}{obj_{ref,n}} \quad 4.3$$

$$PC_o = \frac{1}{|N_o|}\sum_{n=1}^{|N_o|}\frac{obj_{ref,n} - obj_{best,n}}{obj_{ref,n}} \times 100 \quad 4.4$$

Table 4.4 Parameter settings of the training and testing scenarios

Factor	Training	Testing
TWT	(80, 3), (80, 6), (80, 8), (90, 2), (90, 6), (90, 8)	(80, 3), (80, 4), (80, 5), (80, 6), (80, 7), (80, 8), (85, 3), (85, 4), (85, 5), (85, 6), (85, 7), (85, 8), (90, 3), (90, 4), (90, 5), (90, 6), (90, 7), (90, 8), (95, 3), (95, 4), (95, 5), (95, 6), (95, 7), (95, 8)
MT	(80, 1.5), (80, 3), (85, 4), (90, 1.5), (90, 3), (90, 4)	(80, 1.5), (80, 2), (80, 2.5), (80, 3), (80, 3.5), (80, 4), (85, 1.5), (85, 2), (85, 2.5), (85, 3), (85, 3.5), (85, 4), (90, 1.5), (90, 2), (90, 2.5), (90, 3), (90, 3.5), (90, 4), (95, 1.5), (95, 2), (95, 2.5), (95, 3), (95, 3.5), (95, 4)
MFT	(70, 3), (85, 3), (97.5, 3)	(70, 3), (72.5, 3), (75, 3), (77.5, 3), (80, 3), (82.5, 3), (85, 3), (87.5, 3), (90, 3), (92.5, 3), (95, 3), (97.5, 3)

4.3.3 GP Parameter Settings

A population size of 750 rules is generated using the ramped-half-and-half method with maximum depth of 8 in the four developed algorithms. The crossover, mutation, and elitism rates are set respectively as 85%, 10%, and 10%. In addition, tournament selection is used with size equal to 5. The algorithm terminates after completing 50 generations. These parameters have been addressed in earlier studies (Hildebrandt et al., 2010b; Shady et al., 2020d). Regarding HGP, the attribute mutation probability of 0.5 is used because it obtained the best results. In addition, it is recommended in another earlier paper (Nguyen et al., 2018b).

4.4 Results

First, multiple experiments are conducted to tune the new parameters in the PGP algorithm. Second, the proposed algorithm is compared with the literature methods related to convergence speed, rule length, and computational time. Third, further discussions are presented to analyse the structure of best-evolved rules and to elucidate their internal mechanisms under the three considered objectives. Lastly, the set of significant features obtained using the PGP algorithm is compared with the results

obtained using the Feature Selection Genetic Programming (FSGP) algorithm (Mei et al., 2017c) to verify the validity of the proposed framework in the three objectives. It is noteworthy that this chapter is not particularly addressing improvement of the solution quality of evolved rules but on achieving more interpretable rules with shorter lengths without sacrificing efficiency.

4.4.1 Fine-tuning the Parameters of the Proposed Algorithm

Because similar findings were obtained for the three objectives, the process of selecting the suitable parameters for the PGP algorithm is presented for the TWT objective. Two new parameters must be adjusted, including the number of selected individuals $\mathbb{S}$ and the initial activation probability $Prob_{int}$. Three values for the number of selected rules are examined while fixing $Prob_{int}$ to 0.5. The $\mathbb{S}$ values are 150, 300, and 450 chosen rules denoted respectively as $Exp.150$, $Exp.300$, and $Exp.450$. The proposed algorithm is executed for 20 independent runs for each parameter setting.

The obtained results are shown in Figure 4.4: the mean values are shown as a solid line; the standard deviations are depicted as shaded area. In addition, a Wilcoxon's rank-sum test with a significant level of 0.05 was conducted. As shown in Figure 4.4(a), although no significant difference was found among the results of the three experiments related to the number of active terminals, $Exp.150$ had a larger number of active terminals compared with the cases of 300 and 450 rules. One reason might be that when the number of selected rules is small ($\mathbb{S} = 150$), the feature selection mechanism does not gain sufficient information to distinguish between terminals' weights. It therefore has low selective pressure. In contrast, high selective pressure is achieved by selecting a larger number of rules ($\mathbb{S} = 300 \text{ and } 500$) resulting in fewer active terminals. Regarding the average number of inactive rules, as shown in Figure 4.4 (b), the three experiments obtained somewhat similar performance. Similar findings are presented in Figure 4.4 (c), where $Exp.150$ obtained larger values of mean rule length and wider standard deviation than other experiments.

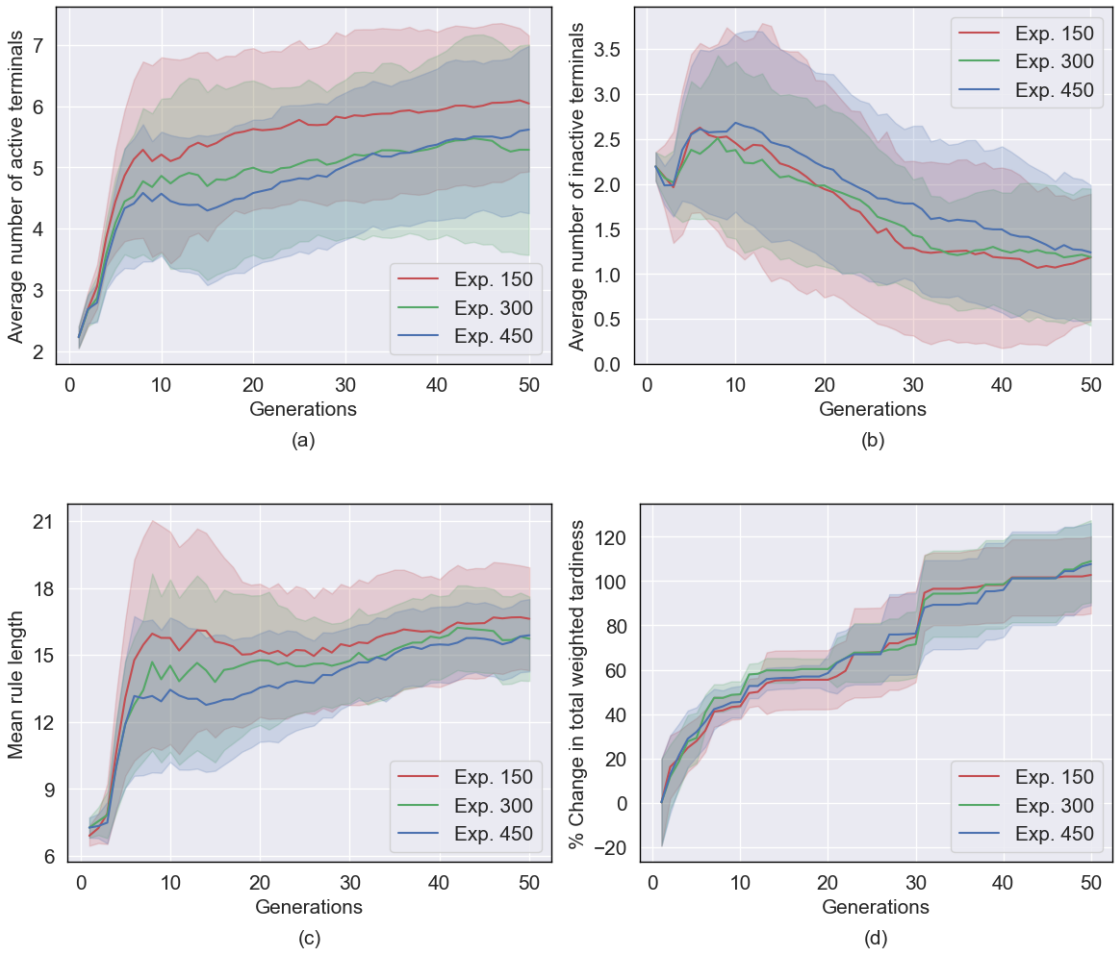


Figure 4.4: Impact of number of selected rules 𝕊 on the PGP algorithm

Regarding the quality of created rules, $Exp.150$ shows a kind of stagnation after the 33rd generation, whereas $Exp.300$ and $Exp.450$ are gradually increasing until the algorithm terminates as depicted in Figure 4.4 (d). These results demonstrate that, although the algorithm selective pressure is influenced by the number of selected rules 𝕊 to some degree, this effect is not statistically significant. Therefore, the number of selected individuals 𝕊 equal to 300 is chosen for the following PGP runs because it achieved slightly better results.

The effects of different values of initial activation probability $Prob_{int}$ were assessed. The examined values are 0.1, 0.5, and 0.9 while fixing the number of selecting rules to 300. As shown in Figure 4.5 (a), it appears that changing the initial activation probability significantly affects the algorithm's computational time. Figure 4.5 (b) illustrates these results because the use of $Prob_{int}$ equal to 0.9 had the highest average rule length among all experiments.

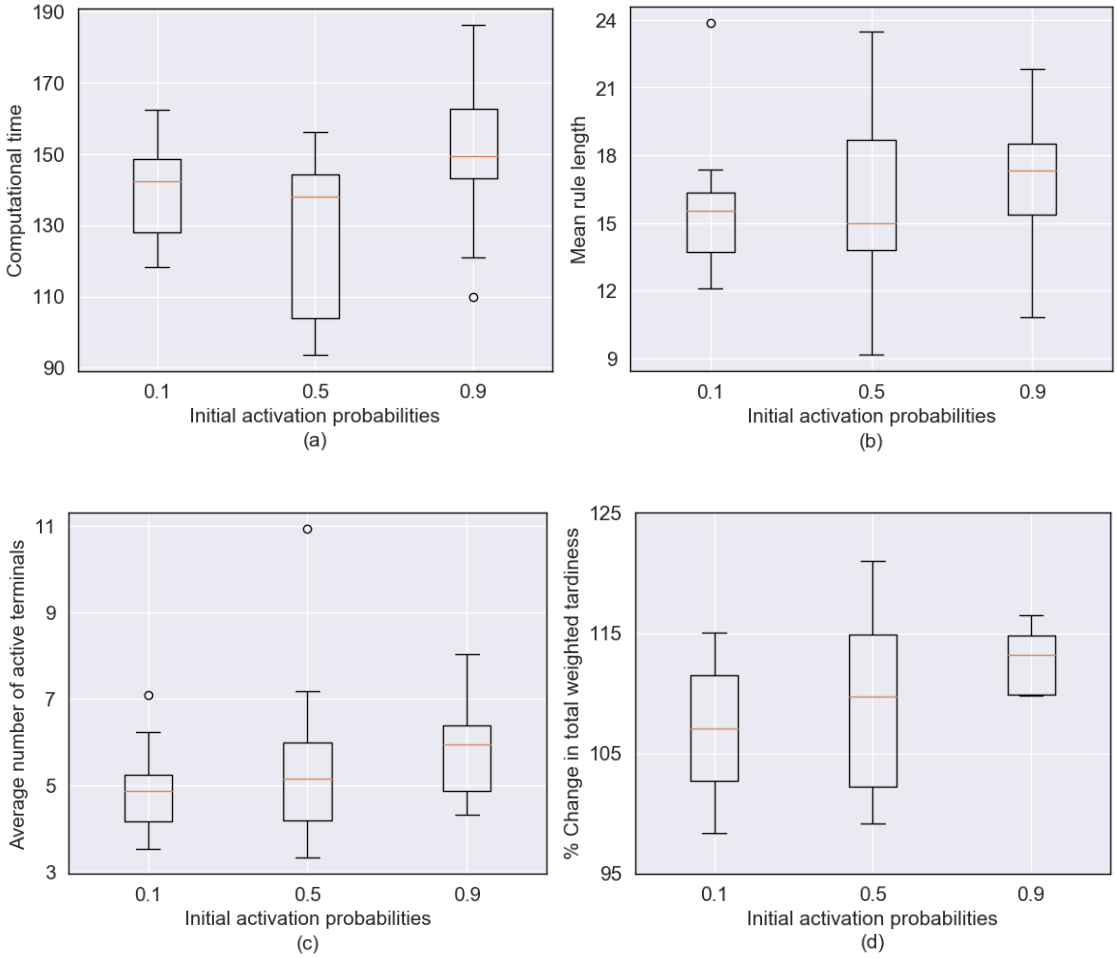


Figure 4.5: Impact of initial activation probability on the PGP algorithm

Statistical differences were found between 0.1 and 0.9, and between 0.5 and 0.9 with p-values equal to 0.011 and 0.017. Regarding the average number of active terminals, no strong effect of changing the initial activation probability was found, as depicted in Figure 4.5 (c). From Figure 4.5 (d), it is clear that a small change in the $Prob_{int}$ value has a negligible effect on the percentage deviation in all total weighted tardiness. The only statistical difference was found between $Prob_{int}$ equal to 0.1 and 0.9 (p=0.02). Consequently, $\mathbb{S}$=300 and $Prob_{int}$=0.5 were chosen for the PGP algorithm because they yielded acceptable results under all performance measures.

4.4.2 Training Performance

Results of statistical analyses by comparison of PGP with the three algorithms from the literature in the three objective functions are represented by the tuple next to the PGP results (PGP versus SGP, PGP versus NSGP, and PGP versus HGP), as depicted in Table 4.5. Symbols "+", "-" and "=" respectively denote that the corresponding result is significantly better, worse than, or similar to its counterpart. Figure 4.6 (a1), (a2), and

(a3) respectively show the percentage changes in the TWT, MT, and MFT objectives. In terms of the solution quality of evolved rules, the PGP algorithm yielded similar performance to those of the SGP and HGP algorithms for the TWT and MT objectives. In addition, PGP showed significantly better performance than all algorithms from the literature in the MFT objective. Obviously, the NSGP algorithm is adversely affected more by premature convergence than the other methods, which confirms the claim that it is beneficial to consider the rule length implicitly to not negatively affect the solution quality.

For the computational budget, the NSGP algorithm experienced the highest computational time because both parents and offspring must be evaluated, which is twice the number of evaluations required in the other GP algorithms, as shown in Figure 4.6 (b1), (b2), and (b3). In contrast, the PGP algorithm significantly outperformed the literature algorithms in terms of the TWT, MT, and MFT objectives because, as shown in Figure 4.6 (c1), (c2), and (c3), the PGP algorithm has the second-lowest average rule length, which significantly reduces the time necessary for fitness assessment of the generated rules. Unlike the NSGP algorithm, only offspring individuals are evaluated in the PGP algorithm. From these results, it is obvious that assigning the rule length equal weight to that of the solution quality (the NSGP algorithm) helps shorter individuals with lower solution quality to survive across generations, which negatively affects the exploration capability of the GP algorithm.

Table 4.5 Mean and standard deviation of the performance measures in the training phase

Perf. Meas.	Objective	SGP	NSGP	HGP	PGP
Percentage change	TWT	110.71± 21.67	39.77 ± 6.73	102.75 ± 21.18	108.92 ± 19.47 (=, +, =)
	MT	125.65 ± 2.46	55.01 ± 10.91	124.35 ± 2.96	125.03 ± 3.53 (=, +, =)
	MFT	56.72 ± 0.5	47.75 ± 12.49	55.5 ± 0.3	56.28 ± 0.45 (+, +, +)
Comp. time	TWT	181.73 ± 10.93	199.22 ± 2.94	162.17 ± 10.33	135.26 ± 6.82 (+, +, +)
	MT	154.55 ± 12.79	179.15 ± 3.91	159.7 ± 14.15	129.66 ± 10.98 (+, +, +)

	MFT	119.26 ± 8.2	138.43 ± 2.67	110.57 ± 6.3	92.16 ± 5.38 (+, +, +)
Mean rule length	TWT	16.83 ± 3.16	6.12 ± 0.22	16.9 ± 3.36	14.32 ± 2.41 (+, -, +)
	MT	14.62 ± 2.67	6.21 ± 0.23	15.44 ± 2.71	12.75 ± 2.41 (+, -, +)
	MFT	11.87 ± 1.51	6.12 ± 0.23	12.11 ± 1.14	10.43 ± 1.15 (+, -, +)
Average number of absent terminals	TWT	5.58 ± 1.84	8.48 ± 0.21	5.66 ± 1.7	6.4 ± 1.57 (+, -, +)
	MT	6.05 ± 1.67	8.42 ± 0.23	5.79 ± 1.83	6.43 ± 1.57 (+, -, +)
	MFT	7.5 ± 1.23	8.48 ± 0.22	7.2 ± 1.01	7.48 ± 1.02 (=, -, +)
Average number of active terminals	TWT	7.42 ± 1.84	4.52 ± 0.21	5.37 ± 1.36	4.88 ± 1.32 (+, -, +)
	MT	6.65 ± 1.67	4.58 ± 0.23	5.29 ± 1.4	4.54 ± 1.01 (+, =, +)
	MFT	5.5 ± 1.23	4.52 ± 0.22	4.53 ± 0.63	3.45 ± 0.62 (+, +, +)

The main reason for using attribute vectors is to guide GP towards important features and to deactivate or exclude irrelevant features resulting in shorter rules. To verify whether the proposed approach meets this goal, the average number of active and absent terminals across generations was traced. As shown in Figure 4.6 (d1), (d2), and (d3), although the four algorithms had the same average number of absent terminals at the first generation, PGP achieved significantly better results than those obtained using the SGP and HGP algorithms for the TWT and MT objectives. For the MFT objective, no significant difference was found between SGP and PGP algorithms, although PGP significantly outperforms the HGP algorithm. Regarding the average number of active terminals presented in Figure 4.6 (e1), (e2), and (e3), the PGP approach achieved the fewest number of active terminals among the SGP and HGP methods for the three objectives. In addition, the rules generated using PGP are significantly smaller than those of NSGP in the MFT objective. Although the NSGP algorithm achieved the

greatest number of absent terminals and the fewest active terminals, these results were achieved at the expense of the quality of evolved rules and high computation costs.

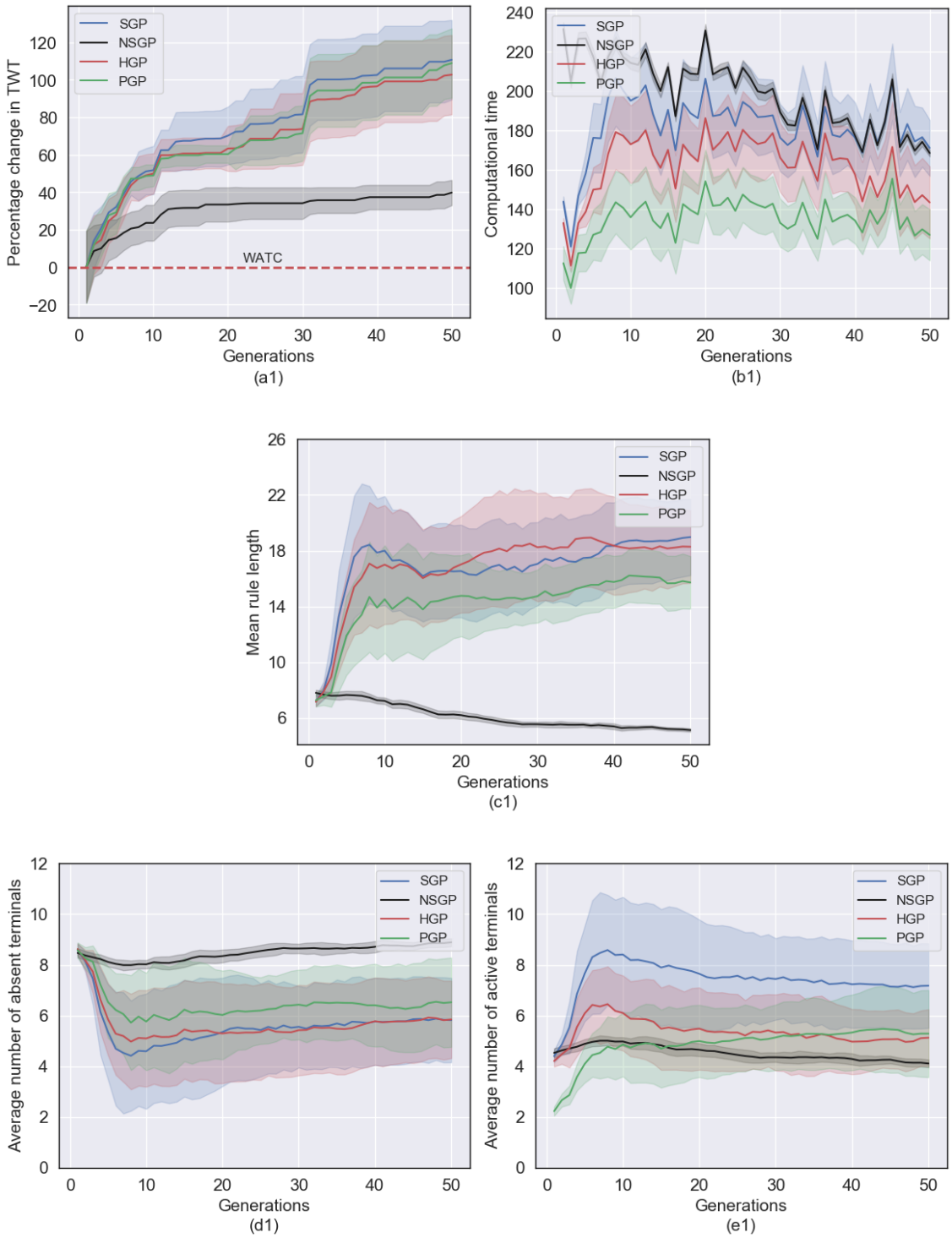
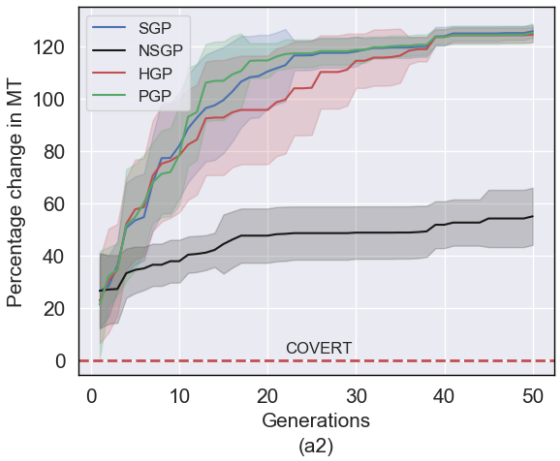
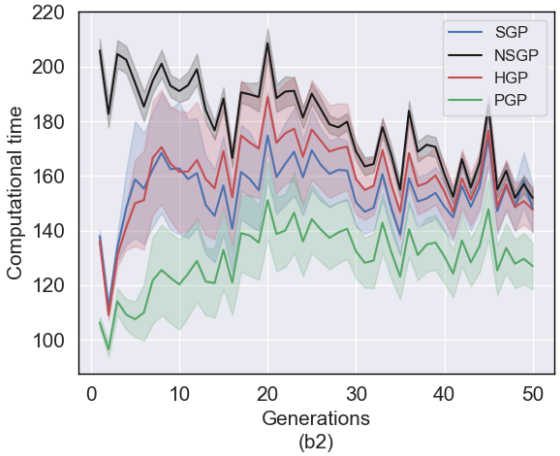
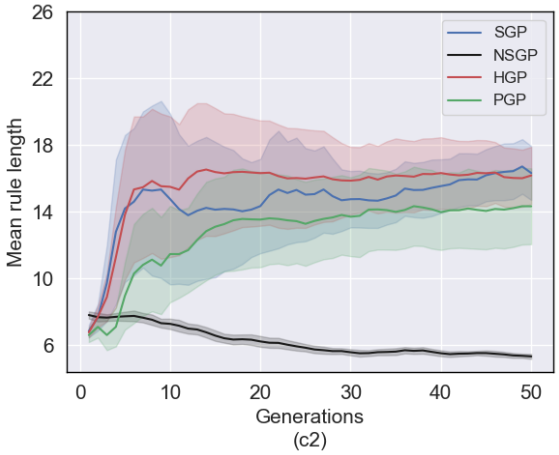
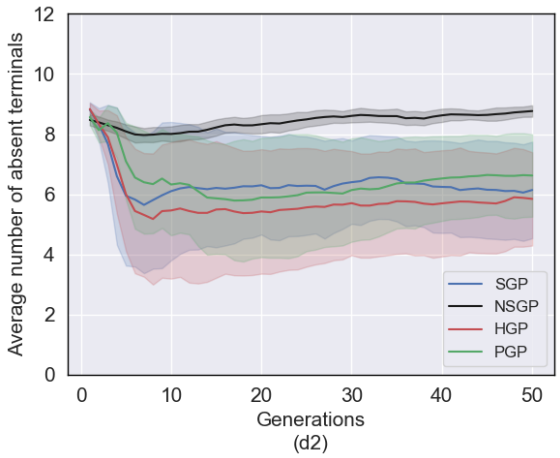


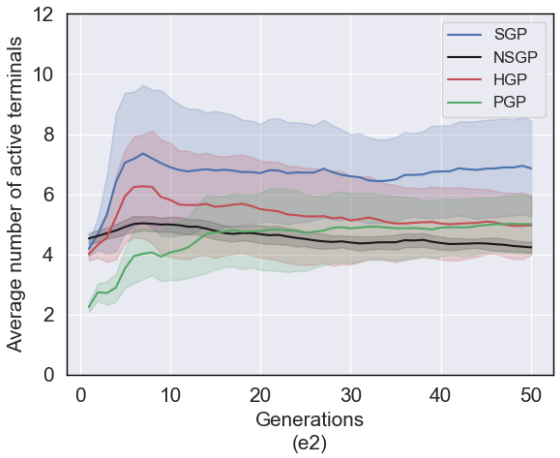
Figure 4.6: The performance of the GP algorithms during the training phase for the three objectives. Figures (a1), (b1), (c1), (d1), (e1) are for the TWT objectives. Figures (a2), (b2), (c2), (d2), (e2) are for the MT objectives. Figures (a3), (b3), (c3), (d3), (e3) are for the MFT objectives.

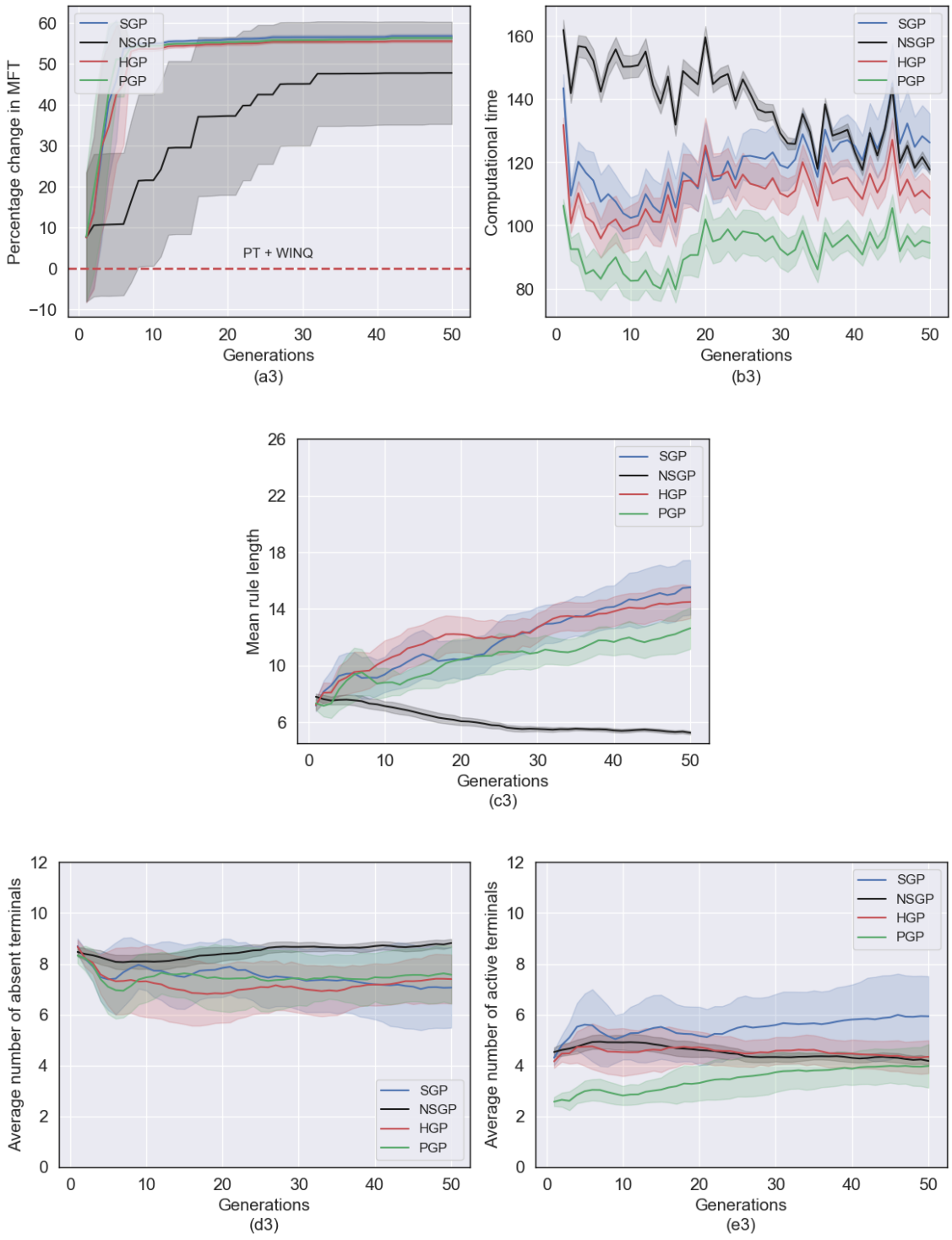

SGP
NSGP
HGP
PGP
Percentage change in MT
COVERT
Generations
(a2)


SGP
NSGP
HGP
PGP
Computational time
Generations
(b2)


SGP
NSGP
HGP
PGP
Mean rule length
Generations
(c2)


SGP
NSGP
HGP
PGP
Average number of absent terminals
Generations
(d2)


SGP
NSGP
HGP
PGP
Average number of active terminals
Generations
(e2)



4.4.3 Testing Performance

Testing experiments were performed to ascertain whether the reduction in the average rule length suppresses the GP's exploration ability to form superior rules under unseen scenarios. In addition, 30 manually made rules were used to provide more evidence that the GP algorithm is a promising machine-learning technique capable of creating scheduling heuristics under different configurations and objectives without direct

human intervention. Table 4.6 (a), (b), and (c) respectively show the means and standard deviations of the studied methods in the TWT, MT, and MFT objectives. Furthermore, the last row of each table provides a summary of the results obtained using a tuple (k, l, m), where k, l, and m respectively represent the number of times a certain method wins (significantly better), draws (no significant difference), and loses (significantly worse) against the PGP method. The best literature rule (BLR) for each scenario is included in parentheses along with its objective value. Because human-made rules are designed to handle specific system settings, no single rule works well in all 24 scenarios. For the TWT objective, the PGP algorithm outperforms the BLR in 11 scenarios. It obtained the same performance in 13 scenarios. Compared with all other literature methods, the PGP algorithm exhibited significantly poor performance in only 4 scenarios against the SGP algorithm while outperforming in 8 scenarios. The NSGP algorithm had the worst TWT results compared with other GP algorithms: the PGP algorithm significantly outperformed it in 18 scenarios and similar performance in 6 scenarios. Finally, HGP obtained significantly worse results compared with PGP in 7 scenarios, with no significant difference found in 17 scenarios.

For the MT objective, the PGP approach significantly outperformed the BLR in 11 scenarios while obtaining similar results in 13 scenarios, as shown in Table 4.6 (b). It is worth noting that when due dates are very tight, the best human-made rule is the PT + WINQ rule, which seems counterintuitive for minimising the MT objective. Similar results have been described in earlier studies because, when the due date factor is very small, numerous jobs become tardy. As a result, the objective becomes reduction of the completion time of these jobs (Mei et al., 2017c). In contrast, the COVERT rule, which is one dispatching rule that takes into account due-date-related information, became the BLR when the tightness factors were large. Comparison of PGP with the other GP literature algorithms shows that it achieved similar performance to that of SGP in all scenarios, in addition to providing significantly better results than the HGP in two scenarios. As expected, the generated rules using the NSGP algorithm had significantly worse solution quality than that obtained using the PGP algorithm in the 24 system configurations.

Regarding the MFT objective, as depicted in Table 4.6 (c), the PGP algorithm significantly outperforms the results obtained from both the BLR and NSGP algorithms in the 12 job shop settings. In addition, the gap widens in challenging scenarios with high utilisation levels (high job arrival rates) compared with low utilisation scenarios.

No significant difference was found between the performance of the PGP algorithm and both SGP and HGP algorithms because the PGP achieved better results in only one scenario in the case of SGP and two scenarios in the case of HGP. Finally, in the three considered objectives, the BLRs showed higher standard deviations than those of the GP algorithms, indicating that the human-made rules have low robustness. Specifically, manually designed rules lack the ability to obtain consistent performance across different job shop settings. Therefore, the GP extensions proposed in this chapter do not limit the solution quality of generated rules in favour of reducing the number of selected features. In other words, the proposed approach is able to achieve a significant reduction of the rule length and of the computational time without sacrificing the performance of evolved rules.

Table 4.6 Mean and standard deviation of the considered methods in the testing phase. (a): the TWT objective, (b): the MT objective, and (c): the MFT objective.

(a): scenarios	**BLR**	**SGP**	**NSGP**	**HGP**	**PGP**
(3, 0.8)	88776.85 ± 30140.07 (COVERT)	56726.01 ± 11241.72	86159.32 ± 15250.35	57047.29 ± 8984.61	56635.67 ± 11655.13 (+, =, +, =)
(3, 0.85)	194286.4 ± 51664.4 (ATC)	149354.06 ± 28496.28	229690.39 ± 32785.38	149306.86 ± 21124.09	145764.9 ± 28602.55 (+, +, +, =)
(3, 0.9)	366232.85 ± 104891.17 (ATC)	363986.61 ± 69343.26	548164.46 ± 73156.71	361161.44 ± 57110.55	347100.0 ± 63449.15 (+, +, +, +)
(3, 0.95)	726419.3 ± 244325.95 (ATC)	865385.5 ±177930.55	1229854.68 ± 162961.44	837916.48 ± 148150.41	788290.79 ± 148226.04 (=, +, +, +)
(4, 0.8)	9744.2 ± 10060.48 (SL/RO)	4945.72 ± 1895.55	7893.51 ± 3099.55	5443.52 ± 2260.1	5310.03 ± 2168.64 (=, +, +, =)
(4, 0.85)	55799.9 ± 34566.46 (COVERT)	28072.82 ± 8561.04	49798.54 ± 13309.3	29601.98 ± 9455.21	29088.92 ± 7867.44 (+, =, +, =)
(4, 0.9)	196604.95 ± 77786.43 (ATC)	134081.75 ± 34149.56	229520.81 ± 41327.15	134403.23 ± 27780.53	134109.43 ± 35477.9 (+, =, +, =)

(4, 0.95)	506982.85 ± 224807.64 (ATC)	531832.68 ± 120474.09	819903.53 ± 123725.86	517073.26 ± 94084.74	497311.42 ± 108193.34 (+, +, +, +)
(5, 0.8)	434.85 ± 99.66 (SL/RO)	277.4 ± 180.26	390.0 ± 346.2	393.06 ± 300.44	391.74 ± 286.37 (=, -, =, =)
(5, 0.85)	3733.4 ± 4094.16 (SL/RO)	2752.09 ± 1521.91	5241.06 ± 2755.81	3232.1 ± 1763.51	3137.53 ± 1629.79 (+, -, +, =)
(5, 0.9)	81346.85 ± 57911.32 (COVERT)	32349.83 ± 14163.97	67476.16 ± 22478.61	34694.23 ± 13783.61	33309.38 ± 12602.53 (+, =, +, =)
(5, 0.95)	338495.05 ± 193371.31 (ATC)	283197.38 ± 77804.52	481649.86 ± 90541.06	276233.92 ± 55507.91	270509.09 ± 71361.67 (+, +, +, =)
(6, 0.8)	51.25 ± 45.69 (SL/RO)	57.92 ± 41.88	68.69 ± 101.95	70.32 ± 68.77	59.68 ± 50.3 (=, =, =, =)
(6, 0.85)	93.65 ± 62.84 (SL/RO)	210.87 ± 143.39	199.58 ± 183.0	239.64 ± 226.55	231.75 ± 244.23 (=, =, =, =)
(6, 0.9)	14170.9 ± 22668.5 (SL/RO)	4876.26 ± 3278.33	12500.55 ± 8100.89	5467.22 ± 3181.07	5459.32 ± 3807.42 (=, -, +, =)
(6, 0.95)	221297.1 ± 164720.05 (ATC)	125112.34 ± 46205.63	238464.92 ± 62936.98	122721.71 ± 39602.29	121501.17 ± 31598.03 (+, =, +, =)
(7, 0.8)	24.95 ± 29.22 (SL)	33.18 ± 31.43	30.77 ± 43.68	33.6 ± 33.06	27.76 ± 25.37 (=, +, =, +)
(7, 0.85)	54.25 ± 76.95 (SL/RO)	32.9 ± 33.08	38.52 ± 59.84	48.68 ± 60.25	36.21 ± 38.45 (=, =, +, +)
(7, 0.9)	328.0 ± 570.32 (SL/RO)	417.13 ± 478.79	1001.46 ± 931.7	618.85 ± 610.8	622.45 ± 742.44 (=, -, +, =)
(7, 0.95)	122338.7 ± 94257.85 (2*PT+WINQ+Npt+WSL)	43338.93 ± 23040.37	96262.26 ± 38381.32	43405.94 ± 18589.56	42166.64 ± 18235.3 (+, =, +, =)
(8, 0.8)	23.2 ± 38.74 (SL/RO)	20.97 ± 23.14	23.86 ± 32.97	20.9 ± 22.62	20.0 ± 20.84 (=, =, =, =)
(8, 0.85)	16.45 ± 29.92 (ATC)	19.86 ± 22.81	21.78 ± 19.5	28.7 ± 40.24	20.25 ± 26.01

					(=, =, =, +)
(8, 0.9)	30.9 ± 44.26 (SL/RO)	54.88 ± 80.95	61.47 ± 128.29	93.69 ± 174.95	36.76 ± 46.94 (=, +, +, +)
(8, 0.95)	32531.45 ± 56731.89 (SL/RO)	10388.34 ± 7859.3	26974.03 ± 17255.77	10957.37 ± 6329.87	10863.88 ± 7267.15 (=, =, +, =)
Summary	(0, 13, 11)	(4, 12, 8)	(0, 6, 18)	(0, 17, 7)	

(b): scenarios	**BLR**	**SGP**	**NSGP**	**HGP**	**PGP**
(1.5, 0.8)	128.19 ± 16.93 (PT+WINQ)	128.83 ± 6.24	158.54 ± 16.3	129.92 ± 5.89	127.2 ± 4.4 (=, =, +, +)
(1.5, 0.85)	175.23 ± 25.55 (PT+WINQ)	177.33 ± 8.26	218.2 ± 23.29	178.31 ± 7.76	175.39 ± 6.55 (=, =, +, =)
(1.5, 0.9)	253.34 ± 43.32 (PT+WINQ)	254.34 ± 12.31	316.05 ± 36.81	256.12 ± 11.7	252.32 ± 10.84 (=, =, +, =)
(1.5, 0.95)	395.69 ± 99.64 (2*PT+WINQ+Npt)	405.73 ± 21.63	504.41 ± 68.12	404.83 ± 20.45	400.7 ± 19.42 (=, =, +, =)
(2, 0.8)	77.09 ± 13.67 (PT+WINQ)	73.33 ± 4.76	100.52 ± 12.58	75.15 ± 5.08	72.96 ± 4.68 (=, =, +, +)
(2, 0.85)	117.9 ± 22.45 (PT+WINQ)	116.6 ± 7.15	158.19 ± 19.97	119.48 ± 7.56	116.32 ± 6.96 (=, =, +, =)
(2, 0.9)	189.99 ± 40.48 (PT+WINQ)	190.14 ± 10.8	256.06 ± 33.53	194.54 ± 11.91	189.65 ± 10.59 (=, =, +, =)
(2, 0.95)	328.34 ± 97.21 (2*PT+WINQ+Npt)	338.86 ± 20.42	444.7 ± 62.6	341.58 ± 20.97	336.16 ± 18.42 (=, =, +, =)
(2.5, 0.8)	42.52 ± 11.04 (COVERT)	34.61 ± 2.95	53.15 ± 8.32	36.1 ± 4.09	35.26 ± 4.14 (+, =, +, =)
(2.5, 0.85)	79.44 ± 18.22 (PT+WINQ)	68.93 ± 5.3	104.1 ± 14.93	72.39 ± 6.91	70.13 ± 6.77 (+, =, +, =)
(2.5, 0.9)	142.2 ± 35.88 (PT+WINQ)	135.36 ± 9.7	198.73 ± 28.76	141.2 ± 11.71	137.65 ± 11.25 (=, =, +, =)
(2.5, 0.95)	274.11 ± 92.69 (2*PT+WINQ+Npt)	280.03 ± 18.48	386.6 ± 57.42	285.14 ± 20.99	279.66 ± 19.41 (=, =, +, =)
(3, 0.8)	20.0 ± 6.8 (COVERT)	12.94 ± 1.88	21.68 ± 5.14	13.46 ± 2.82	13.37 ± 2.62 (+, =, +, =)
(3, 0.85)	46.69 ± 15.98 (COVERT)	35.52 ± 3.67	59.69 ± 10.51	38.0 ± 5.74	37.12 ± 5.35 (+, =, +, =)

(3, 0.9)	108.18 ± 30.73 (PT+WINQ)	91.04 ± 8.41	144.38 ± 22.64	96.21 ± 10.77	92.98 ± 10.49 (+, =, +, =)
(3, 0.95)	230.41 ± 84.13 (PT+WINQ)	226.65 ± 17.31	328.82 ± 50.91	233.6 ± 19.47	228.29 ± 19.67 (=, =, +, =)
(3.5, 0.8)	7.85 ± 3.75 (COVERT)	3.77 ± 1.01	7.01 ± 2.43	4.04 ± 1.5	4.02 ± 1.44 (+, =, +, =)
(3.5, 0.85)	25.56 ± 10.97 (COVERT)	15.65 ± 2.3	28.97 ± 6.91	16.77 ± 3.97	16.56 ± 3.57 (+, =, +, =)
(3.5, 0.9)	75.33 ± 32.59 (COVERT)	55.38 ± 6.06	97.64 ± 17.98	60.59 ± 9.53	58.06 ± 9.42 (+, =, +, =)
(3.5, 0.95)	195.05 ± 77.56 (PT+WINQ)	177.78 ± 15.2	272.48 ± 46.38	187.26 ± 18.83	181.47 ± 17.99 (=, =, +, =)
(4, 0.8)	2.19 ± 2.26 (SL/RO)	0.88 ± 0.56	1.66 ± 0.94	0.97 ± 0.72	0.97 ± 0.71 (=, =, +, =)
(4, 0.85)	12.33 ± 7.5 (COVERT)	5.72 ± 1.4	11.69 ± 3.75	6.26 ± 2.41	6.24 ± 2.23 (+, =, +, =)
(4, 0.9)	49.54 ± 27.48 (COVERT)	29.79 ± 4.73	59.92 ± 13.65	33.91 ± 8.29	32.45 ± 7.59 (+, =, +, =)
(4, 0.95)	166.99 ± 71.12 (PT+WINQ)	135.44 ± 13.76	218.29 ± 39.4	143.33 ± 17.46	138.76 ± 16.98 (+, =, +, =)
Summary	(0, 13, 11)	(0, 24, 0)	(0, 0, 24)	(0, 22, 2)	

(c): scenarios	**BLR**	**SGP**	**NSGP**	**HGP**	**PGP**
(3, 0.7)	286.17 ± 10.95 (PT+WINQ)	283.93 ± 1.62	284.82 ± 1.64	283.78 ± 1.63	283.97 ± 1.65 (+, =, +, =)
(3, 0.725)	298.55 ± 12.79 (PT+WINQ)	295.42 ± 1.92	296.27 ± 2.01	295.6 ± 1.79	295.36 ± 1.74 (+, =, +, +)
(3, 0.75)	312.78 ± 13.49 (PT+WINQ)	308.99 ± 2.05	310.0 ± 2.36	308.78 ± 2.13	308.88 ± 2.12 (+, =, +, =)
(3, 0.775)	328.34 ± 16.39 (PT+WINQ)	324.31 ± 2.23	325.56 ± 2.55	324.14 ± 2.38	323.88 ± 2.18 (+, +, +, =)
(3, 0.8)	348.07 ± 19.69 (PT+WINQ)	341.64 ± 2.45	343.94 ± 2.77	341.59 ± 2.53	341.55 ± 2.46 (+, =, +, =)
(3, 0.825)	370.3 ± 24.3 (PT+WINQ)	362.51 ± 2.81	365.23 ± 3.36	362.23 ± 2.96	362.3 ± 2.67 (+, =, +, =)
(3, 0.85)	396.63 ± 27.89 (PT+WINQ)	387.55 ± 3.34	390.99 ± 3.87	387.8 ± 3.45	387.2 ± 3.56 (+, =, +, +)
(3, 0.875)	431.19 ± 37.2 (2*PT+WINQ+Npt)	417.62 ± 3.69	422.24 ± 4.51	417.39 ± 4.04	417.69 ± 3.79 (+, =, +, =)
(3, 0.9)	475.83 ± 45.48 (PT+WINQ)	458.27 ± 5.0	464.75 ± 6.02	458.02 ± 5.0	458.26 ± 4.85 (+, =, +, =)
(3, 0.925)	532.32 ± 67.36 (PT+WINQ)	513.2 ± 6.19	523.33 ± 7.35	513.27 ± 6.74	514.11 ± 6.57 (+, =, +, =)
(3, 0.95)	618.97 ± 101.44 (2*PT+WINQ+Npt)	592.13 ± 8.97	607.27 ± 10.26	591.57 ± 9.06	592.9 ± 8.79 (+, =, +, =)
(3, 0.975)	728.07 ± 148.15	701.35 ± 13.6	723.83 ± 14.98	701.46 ± 11.58	702.07 ± 11.51 (+, =, +, =)

	(2*PT+WINQ+Npt)				
Summary	(0, 0, 12)	(0, 11, 1)	(0, 0, 12)	(0, 10, 2)	

4.4.4 Feature Analysis of the GP Best Evolved Rules

Because the SGP, HGP, and PGP algorithms generated dispatching rules with similar performance compared with those obtained using the NSGP, the best-generated rules using the SGP, HGP, and PGP methods are considered for additional analysis. Figure 4.7 presents the distribution of the terminals of the 20 best-rules generated using the three algorithms, where Figure 4.7 (a), (b), and (c) respectively represent the results obtained for the TWT, MT, and MFT objectives. It is readily apparent that the PGP algorithm evolved smaller rules, and that the gap separating relevant and irrelevant terminals is wider than that of the literature methods. In addition, the number of terminals in the rules created using the PGP algorithm is much smaller than that evolved using other literature methods resulting in more compact and interpretable rules. For the TWT objectives, the PGP rules achieved a reduction in the number of terminals by 32.53% and 15.38% compared with the SGP and HGP algorithms. In addition, the most important terminals are PT, SL, JW, RO, and WINQ, whereas Npt, WT, JR, and OR terminals are not significant.

Regarding the MT objective, the PGP rules have 31.19% and 10.36% fewer terminals than the rules generated using the SGP and HGP algorithms. In contrast to the TWT target, as expected, the GP algorithms considered the JW terminal as an irrelevant feature, whereas the weights of the RO, WR, SL, and WINQ terminals increased notably. In the case of the MFT objective, the PGP algorithm reduced the number of terminals in the best rules by 26.52% and 18.65%, respectively, compared with the SGP and HGP algorithms. The best rules generated using the three algorithms extensively included the PT, Npt, and WINQ terminals, indicating their importance in reducing the MFT objective. These findings are consistent with the claim that the weight of terminals varies based on the objective. The selection ability of the GP methods in the literature is limited. In addition, the proposed feature selection approach is able to identify important features and to exclude irrelevant features in different objective functions, resulting in smaller rules that positively affect GP computation costs.

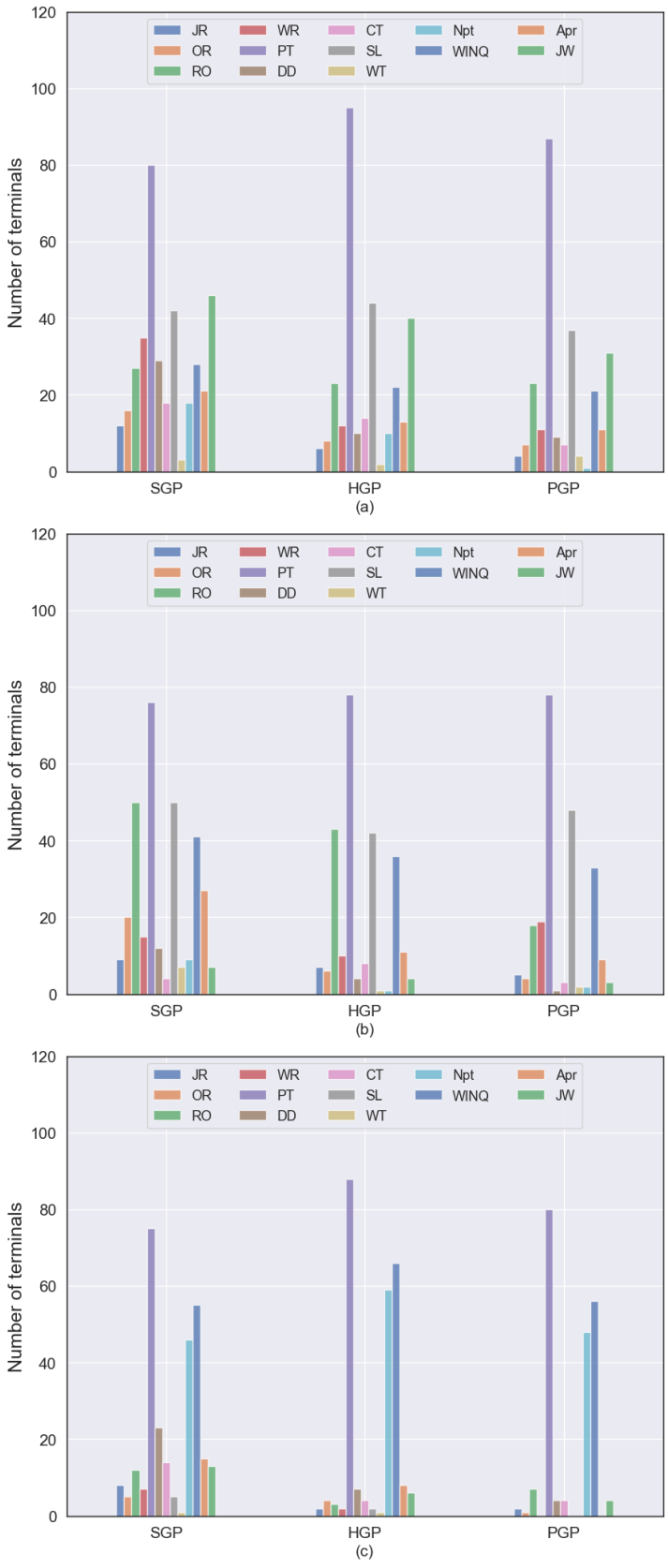


Figure 4.7: Terminals distribution in the best-evolved rules for the SGP, HGP and PGP algorithms. (a): for the TWT objective, (b): for the MT objective, (c): for the MFT objective

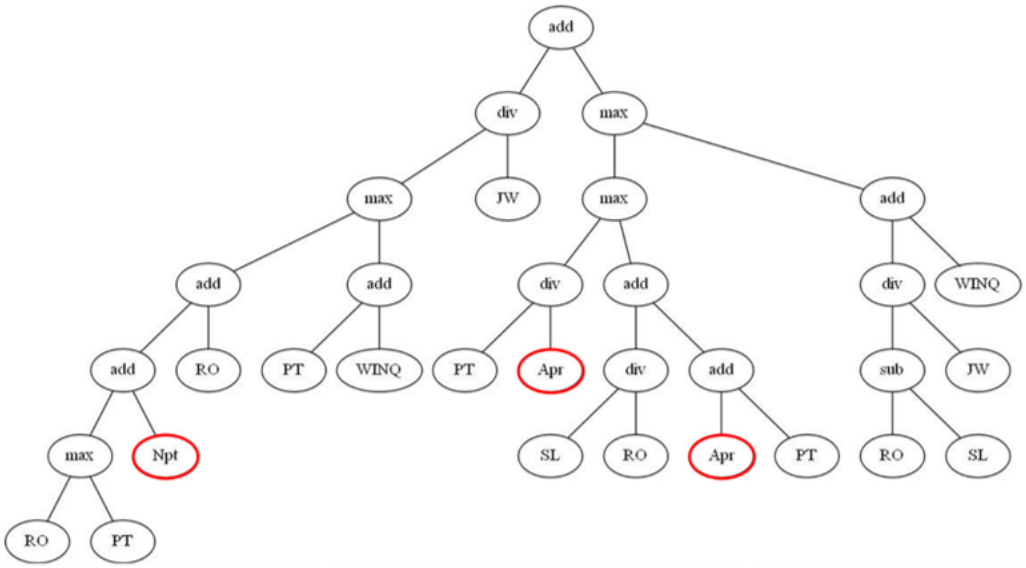
4.4.5 Behaviour Analysis of the PGP Best Evolved Rules

To gain more knowledge about the PGP rule structure, two versions of the best-evolved rules rule are developed for the TWT, MT, and MFT objectives. The first version presents only priority functions, as shown in Figure 4.8 (a1, b1, and c1), whereas the second version, shown in other panels of Figure 4.8 (a2, b2, and c2), includes information presented respectively in their attribute vectors for the TWT, MT, and MFT objectives. Regarding the TWT objective, all existing terminals are active, except that the Npt and Apr terminals (highlighted by red frames) are deactivated. Therefore, the attribute vector was able to eliminate three terminals from the current 17, as shown in Figure 4.8 (a1) and (a2). In addition, the simplified version of the rule is represented in a mathematical form in Equation 4.5. In the case of the MT objective, the proposed representation was able to eliminate 13 terminals out of the 20 terminals that occurred in the evolved rule. The disabled terminals were the JR, OR, WT, and JW terminals, revealing their negligible effect on minimising the MT objective. Moreover, the simplified version of the best-generated rule to minimise the MT objective is shown in a mathematical form in Equation 4.6. Regarding the MFT objective, the attribute vector deactivated 3 terminals from the current 12 terminals presented in the priority function. Excluded terminals are the SL, Apr, and JW terminals. The included terminals are the PT, WINQ, and Npt terminals. The best PGP rule generated for minimising the MFT objective is presented mathematically in Equation 4.7.

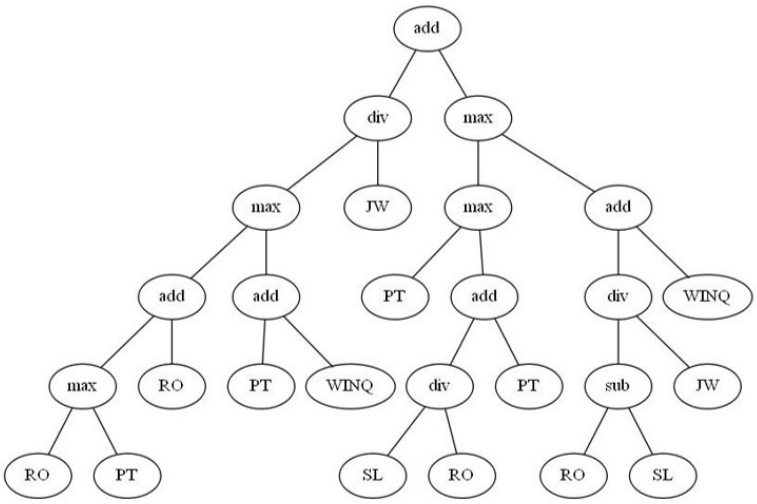
$$TWT\ rule = \frac{Max(Max(RO, PT) + PR, PT + WINQ)}{JW} + Max\left(Max\left(\frac{SL}{RO} + PT, PT\right), \frac{RO - SL}{JW} + WINQ\right) \quad 4.5$$

$$MT\ rule = \min\left(\frac{WINQ + \frac{SL}{RO}}{PT}, max(CT, PT\ x\ NPT)\right) \quad 4.6$$

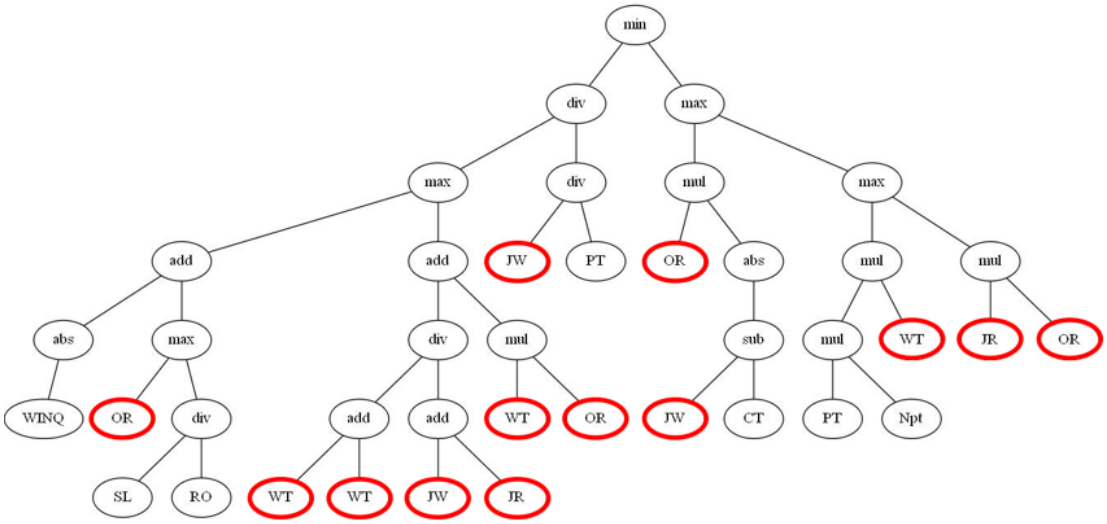
$$MFT\ rule = \max(\max(\min(Npt, WINQ + PT) + WINQ\ , PT)\ , WINQ - \max\ (WINQ + PT))\ x\ PT \quad 4.7$$


add
div
max
max
JW
max
add
add
add
div
add
div
WINQ
add
RO
PT
WINQ
PT
Apr
div
add
sub
JW
max
Npt
SL
RO
Apr
PT
RO
SL
RO
PT

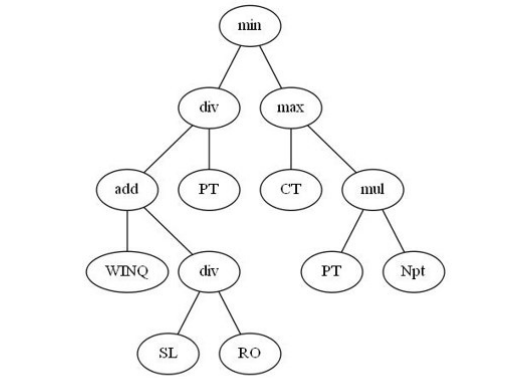
(a1)


add
div
max
max
JW
max
add
add
add
PT
add
div
WINQ
max
RO
PT
WINQ
div
PT
sub
JW
RO
PT
SL
RO
RO
SL

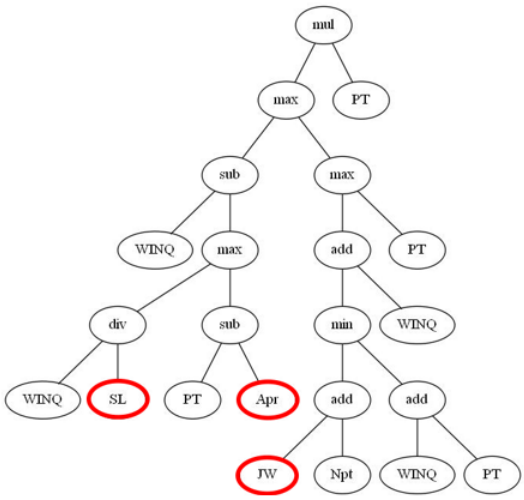
(a2)


min
div
max
max
div
mul
max
add
add
JW
PT
OR
abs
mul
mul
abs
max
div
mul
sub
mul
WT
JR
OR
WINQ
OR
div
add
add
WT
OR
JW
CT
PT
Npt
SL
RO
WT
WT
JW
JR

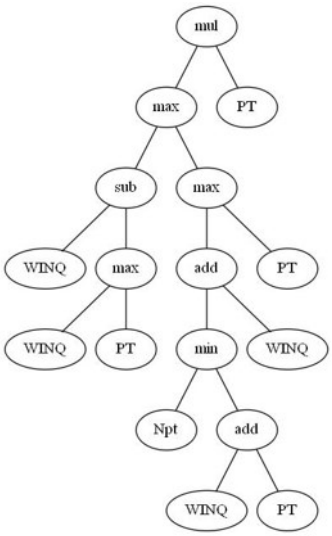
(b1)



(b2)



(c1)



(c2)

Figure 4.8: The priority function of the best PGP rules. (a1) for the TWT without considering the attribute vector, (a2) for the TWT after considering the attribute vector, (b1) for the MT without considering the attribute vector, (b2) for the MT after considering the attribute vector, (c1) for the MFT without considering the attribute vector, (c2) for the MFT after considering the attribute vector

To elucidate the phenotypic characterisation of the PGP rules, 20 decision situations were sampled from an actual DES run, as recommended in an earlier paper (Hildebrandt and Branke, 2015b). The phenotypic behaviour helps to understand the internal mechanism of evolved rules. It provides some insights into how these rules prioritise operations and why they achieved superior performance. Technically, these experiments are conducted to examine the influence of changing terminals' values in estimating priority values of jobs. The rule assigns high priorities to jobs with lower priority values (higher rank) from the set of waiting jobs. Regarding the TWT objective, the best PGP rule favours jobs with low processing time, high weight, low WINQ, low slack, and numerous remaining operations. In addition, the order of the included features based on their weight in descending order is: SL, PT, WINQ, JW, and RO. The best PGP rule generated to minimise the MT objective assigns higher priority to jobs with less processing time, less work in the next queue, less slack value, and more unprocessed operations. Additionally, it appears that the current time and the next processing time features have no direct effect on estimating the priority of a job. The order of the included features according to their influence is the following: PT, WINQ, SL, RO, CT, and Npt terminals. For the MFT objective, the evolved rule assigns higher priority values to jobs with the following characteristics: less processing time, less work in the next queue, and less processing time of the next operation. Moreover, the PT feature is the most significant terminal in terms of the performance of the evolved rule followed by the WINQ terminal, and finally the Npt terminal. It is noteworthy that experts may take advantage of these findings to design superior rules manually in less complex structures compared with PGP rules. However, the exact numerical estimation of terminals' relations and interaction effects among them remains challenging and requires further future investigation.

4.4.6 Feature Selection Verification

To verify the credibility of the proposed feature selection framework, an efficient feature selection algorithm from the literature (Mei et al., 2017c) was developed for comparison and designated as FSGP. In contrast to the proposed approach, which selects important features during the GP run (online) using relative weights of terminals (probabilistic), the FSGP algorithm considers feature selection as a pre-processing step (offline) to select features used in future GP runs in a binary manner (include/exclude). Therefore, it is unsuitable to compare it to the PGP algorithm using the same performance measures used earlier. Instead, it is employed to ensure that the proposed

approach is able to identify the same critical terminals obtained using the FSGP algorithm in an online manner. The activation probability of each terminal is tracked during the PGP run to estimate the terminals' weights at a given generation. As shown in Figure 4.9 (a2, b2, and c2), the colour of each cell in the heat maps represents the average activation probability for a specific terminal during the PGP run. In contrast, the results obtained using the FSGP algorithm are depicted in Figure 4.9 (a1, b1, and c1), respectively, for the TWT, MT, and MFT objectives. Important terminals identified at the end of the FSGP run are shown in bright cells, whereas irrelevant terminals are shown in dark cells. Although the y-axis label differs between the two methods, they express the same meaning because the proposed approach applies feature selection during the run (generations), whereas the FSGP approach obtains the set of important features when the GP run finishes (GP replications).

For the TWT objective results presented in Figure 4.9 (a2), similar findings were obtained using the two algorithms as the most important terminals (bright columns) are RO, WR, PT, SL, WINQ, Apr, and JW. Although the DD terminal is identified as an important terminal in the PGP algorithm, it is regarded as an irrelevant terminal using the FSGP algorithm. The inclusion of the SL terminal might be the reason because it can substitute some DD and CT terminals and can therefore affect their weights indirectly. Terminals of least importance (dark columns) include JR, OR, CT, WT, and Npt terminals. Regarding the MT objective, the significant terminals are RO, PT, DD, CT, SL, WINQ, and Apr terminals, whereas the non-significant terminals include JR, OR, WR, WT, Npt, and JW. Finally, for the MFT objective, the set of important terminals contains PT, Npt, and WINQ terminals. In addition, CT and Apr terminals are medium-weight features, whereas the set of irrelevant terminals includes all the remaining terminals. As the results showed, the proposed approach is efficient for selecting significant terminals and for eliminating irrelevant ones using a probabilistic selection mechanism during the GP run.

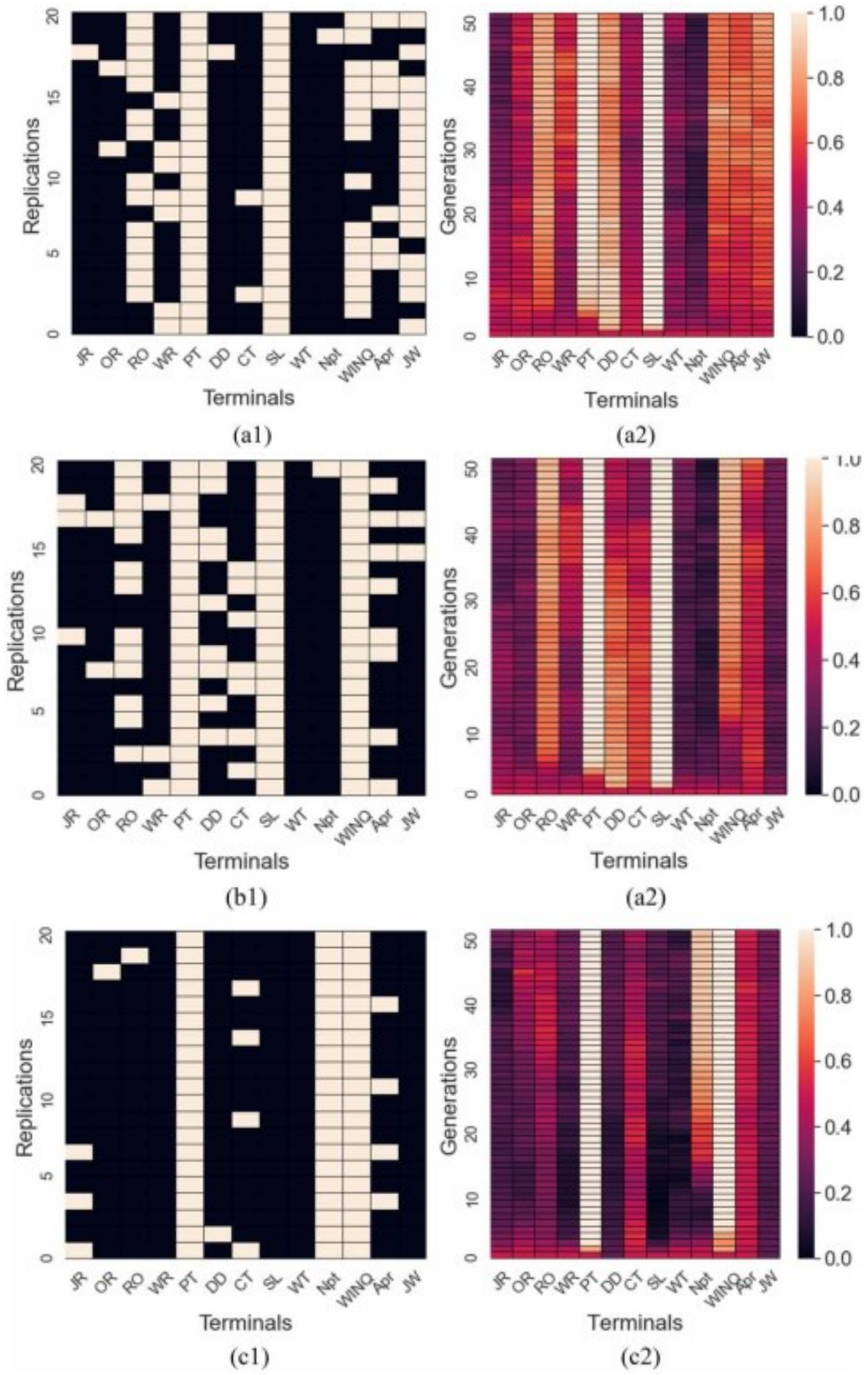


Figure 4.9: Matrix plot of feature selection results. (a1) the FSGP for the TWT, (a2) the PGP for the TWT, (b1) the FSGP for the MT, (b2) the PGP for the MT, (c1) the FSGP for the MFT, (c2) the PGP for the MFT

4.5 Chapter Summary

This chapter proposed a new GP representation and an online feature selection approach for evolving more interpretable rules for the dynamic job shop scheduling problems using the GP algorithm. The new attribute vector representation controlled complex GP structures. It also extracted useful information related to the terminals' contributions. Then, evolutionary information gained from the current generation was used by the feature selection mechanism to guide the GP to consider more important terminals and neglect irrelevant ones. The newly introduced parameters were fine-tuned through some pilot experiments. Then their effects were evaluated using various performance measures. In addition, the proposed algorithm (PGP) was compared with three algorithms (SGP, NSGP, and HGP) from the literature using rule length, computational time, and solution quality as performance measures. Regarding the solution quality of evolved rules, three objective functions were investigated, including total weighted tardiness, mean tardiness, and mean flow time.

Experimentally obtained results demonstrated the effectiveness of the proposed approach for generating compact rules while reducing computation time considerably without compromising solution quality. The reduction in computation costs resulted from a significant decrease in the average rule length and the average number of active terminals in the PGP method compared with counterpart methods reported in the relevant literature. In addition, the generality of evolved rules was evaluated through a set of testing scenarios. Similar findings were achieved. Regarding the distribution of the terminals in the best rules, the proposed approach obtained smaller rules with more meaningful terminals. The best-evolved PGP rules were represented in both tree and mathematical forms to gain useful insights into their structure, how the rules estimate priority values, and the reasons underlying their superior performance. Finally, the ability of the PGP algorithm to identify important features was evaluated by comparing it with an offline feature selection algorithm from the literature in the three objective functions. As the results showed, the PGP algorithm identified the same set of important features in an online manner without additional GP runs.

Chapter 5. GEP WITH FEATURE SELECTION FOR DYNAMIC JOB SHOP SCHEDULING PROBLEMS

5.1 Introduction

As stated in previous chapters the variable representation of the tree-based GP approach is the reason behind the exponential growth in the size of generated rules due to the bloating effect. Therefore, in order to reduce the size of the GP search space, two research directions have been proposed (Nguyen et al., 2017c).

I. Integrate the tree-based GP algorithm with a feature selection algorithm.
II. Develop other GP representations with specific restrictions rather than the standard tree representation.

The key difference between the two methods is that the first approach guides the GP algorithm to promising areas in the search space by focusing on the most significant terminals, whereas the second approach reduces the GP search space by changing the encoding scheme of heuristics and imposing several restrictions on the growth of generated rules. In other words, the representation approach deals directly with the GP bloating problem, while the feature selection approach handles the problem in an indirect manner (Zhang, Mei, and Zhang 2019).

Regarding the integration between the tree-based GP algorithm and feature selection approach, this research direction is the most common technique in the literature. Therefore, the objective of the previous chapter was to develop a new feature selection approach for the GP algorithm. Regarding the use of other GP representations to restrict

the search space, the GEP algorithm is used in a limited number of studies compared with the GP algorithm to generate JSS rules as shown in (Nie et al., 2010, 2011, 2013a; Ozturk et al., 2019). Although the GEP algorithm obtained higher quality dispatching rules in a shorter computational time compared with the GP algorithm in previous studies (Nguyen et al., 2017c), its use is still limited for two main reasons. First, the GEP-generated rule usually contains a noncoding portion that is not used to estimate priority values at the current generation, however, these terminals and functions might have an impact on future generations when genetic operators alter their sequence. Therefore, the GEP individual needs a decoding process in order to be converted to a human-readable format (mathematical format) which is an additional process compared with the tree-based GP representation. Second, there is a larger set of genetic operators and several parameters that must be adjusted based on the problem understudy than the standard GP algorithm which requires many computationally expensive experiments (Ferreira, 2001). Specifically, there are ten genetic operators and parameters for the GEP algorithm including head length, number of genes, linking function, mutation, IS (Insertion Sequence) transposition, RIS (Root IS) transposition, gene transposition, one-point, two-point, and gene recombination rates, whereas there are only three parameters in the GP algorithm which are maximum tree depth, crossover, and mutation rates.

Because of the above reasons, a feature selection approach for the GEP algorithm has not been proposed in the literature, whereas there are five approaches for the GP algorithm even though both algorithms generate rules that might contain insignificant terminals. Therefore, this chapter has three main objectives as follows.

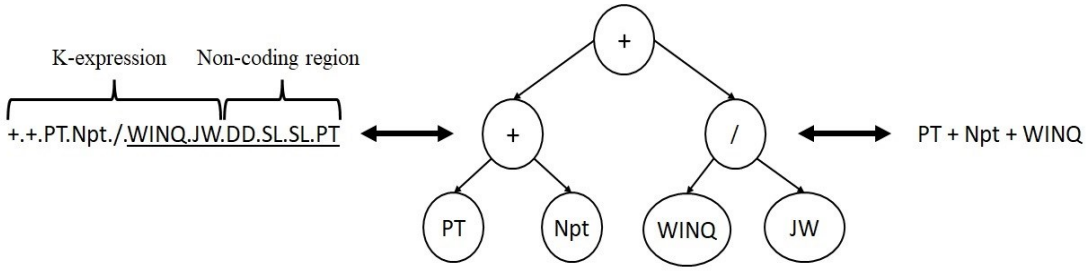
a) Modify the feature selection approach proposed in the previous chapter to be applicable to the GEP algorithm.
b) Study the impact of integrating the modified approach with the GEP algorithm.
c) Modify the GP feature selection approach introduced in (Nguyen et al., 2018a) to be applicable to the GEP algorithm, and thus compare its performance with the proposed GEP algorithm.

The rest of this chapter is organized as follows. Section 5.2 provides a detailed explanation of the proposed GEP algorithm with the feature selection approach. Section 5.3 presents the numerical experiments including the fitness assessment module, design of the experiments, and parameter settings. Training and testing results are given in Section 5.4. Finally, Section 5.5 presents the conclusions of this chapter.

5.2 GEP algorithm with the proposed feature selection approach

Because there is a noncoding region in the GEP individuals, the attribute vector is linked to the K-expression (valid portion) of a given individual rather than the whole linear chromosome of the rule. An illustrative example is presented in Figure 5.1, where a GEP rule is encoded using the proposed and literature attribute vector representations. The length of the head is set to 5, and the maximum number of arguments in the function set, shown in Table 4.1, equals 2. Then, the tail length $t = 5 \times (2 - 1) + 1 = 6$ (underlined elements). The 11-elements genotype is decoded into an expression tree (K-expression) containing only 7 elements. Therefore, the proposed and literature attribute vectors are modified to be linked to the K-expression part only instead of the original genotype as in the case of the GP algorithm. For instance, although SL terminal occurred in the linear chromosome (non-coding region), it is presented as an absent terminal in both representations. However, the main difference between the proposed and literature representations remains the same. The literature representation indicates that there are ten active (important) attributes even though there are only three of them in the K-expression. In contrast, with respect to the proposed representation, it is easy to distinguish between active terminals (PT, Npt, WINQ), inactive terminals (JW), and absent terminals (the rest) without reference to the expression tree. Therefore, the proposed representation offers the following advantages.

I. It supports the attribute vector being precisely bound to the valid region of its corresponding rule. Thus, attribute vectors can be used to abstract the complex structure of DRs without being affected by noncoding regions.
II. It ensures that any change (mutation) of the feature's state in the attribute vector will have a direct effect on the performance of the priority function.
III. It enables the phenotype behaviour of evolved rules to change in response to changes in attribute vectors without affecting their genotypes. This maintains the evolutionary information presented in the structure of evolved rules across generations.



(a) Genotype (b) Expression Tree (c) Simplified Phenotype

Representation	JR	OR	RO	WR	PT	DD	CT	SL	WT	Npt	WINQ	Apr	JW
Literature	1	1	1	1	1	1	1	0	0	1	1	1	0
Proposed	0	0	0	0	1	0	0	0	0	1	1	0	-1

Figure 5.1: Example of a rule using the literature and proposed representations.

The pseudocode of the proposed algorithm that integrates the GEP algorithm with the attribute vector is shown in Figure 5.2. The algorithm starts by initializing a random population P of dispatching rules by using a predefined set of functions and terminals. Each rule R_i consists of two main parts, expression tree ET_i which is the K-expression of the GEP chromosome, and an attribute vector AV_i using the proposed representation. The activation probability array AP is initialized with the same initial probability $Prob_{int.}$ for all terminals. After several pilot experiments, the $Prob_{int.}$ is set to 0.5 since it obtained a robust performance regarding the considered objectives (Shady et al., 2021c). In order to avoid overfitting to a specific problem instance, the same random seed is employed to evaluate all individuals in the same generation while the seed value changes between generations. A set of training instances N is used to evaluate the performance of the generated rules under different job shop settings.

For minimisation objectives, if the fitness value of a specific rule $f(R_i)$ is smaller than the best individual found so far $f(R_{best})$, the best rule and its fitness value are updated as depicted in lines 7 through 13. Afterwards, the best-evolved rules are chosen to represent the parents of the next generation. The weight of each terminal in the activation probability array is estimated using a subset $|\mathbb{S}|$ of the best individuals. Then, the GEP genetic operators are applied to the genotypes of individuals to generate offspring. Similar to the previous chapter, the proposed mutation operator (feature selection) is applied to the attribute vector of evolved rules to activate critical terminals and deactivate irrelevant ones, as shown in lines 18-28. The newly created rules constitute the next generation, and this evolutionary process is repeated for several generations until the termination condition is satisfied.

Inputs: training simulation scenarios $N \leftarrow \{N_1, N_2, \dots, N_t\}$

Output: the best evolved rule R_{best}

1: Initialize population $P_i \leftarrow \{R_1, R_2, \dots, R_n\}$,

$$R_i \leftarrow \{ET_i, AV_i\}, AV_i \leftarrow \{x_{i1}, x_{i2}, \dots, x_{iT}\}$$

2. Initialize activation probabilities array $AP \leftarrow \{T\ values\ equal\ 0 < Prob_{int} \leq 1\}$

3: Set $R_{best} \leftarrow null$ and the best fitness value $f(R_{best}) \leftarrow +\infty$

4: $gen \leftarrow 1$

5: **while** $gen\ \leq$ max $generation$ **do**

6: reset the random seed

7: **for all** $R_i\ \in\ P_{gen}$ **do**

8: evaluate $f(R_i)$ by applying R_i to each scenario $N_k \in N$

9: **if** $f(R_i) <\ f(R_{best})$ **then**

10: $R_{best} \leftarrow R_i$

11: $f(R_{best}) \leftarrow\ f(R_i)$

12: **end if**

13**:** **end for**

14: select the best |P| individuals of P_{gen} to join mating pool

15: estimate AP using a subset | $\mathbb{S}$ | of the best rules

16: **for all** $R_i\ \in\ P_{gen}$ **do**

17: apply genetic operators on ET_i

18: **for all** $x_{ij}\ \in\ AV_i$ **do**

19: **if** terminal j not in ET_i **then**

20: $x_{ij} \leftarrow 0$

21: **else**

22: **if** $rand \leq AP_j$ **then**

23: $x_{ij} \leftarrow 1$

24: **else**

25: $x_{ij} \leftarrow -1$

26: **end if**

27: **end if**

28: **end for**

29: **end for**

30: $gen \leftarrow gen + 1$

31: **end while**

32: **return** R_{best}

Figure 5.2: Proposed gene expression programming algorithm with the feature selection approach.

5.3 Numerical Experiments

5.3.1 Fitness Assessment Module

A simulation model for the symmetrical job shop used in previous studies (Nguyen et al., 2014b; Shady et al., 2021d) was developed using the following settings:

- Jobs arrive stochastically according to a Poisson distribution.
- The job shop consists of ten machines (no breakdowns).
- A warm-up period of 500 jobs is used, and the statistics are collected from the next 2000 jobs.
- Each job has 2 to 10 operations (re-entry is not allowed).
- Processing times follow a uniform discrete distribution with a range of [1, 49].
- The weights of 20%, 60%, and 20% of jobs are set as 1, 2, and 4 respectively.
- Job due dates are assigned using the total work content method with different tightness factor values.

The meta-algorithm shown in Figure 5.3 is employed to generate non-delay schedules for the evolved dispatching rules. Given specific job shop settings and a dispatching rule, there are two events that increment the current time in the simulation

model. The first event is when a job is released, and the second event is when a machine is idle, as shown in lines 3 and 10, respectively. A released job can be processed immediately if the next machine on its route is idle; otherwise, it joins the machine's queue, as illustrated in lines 3-8. When a machine becomes free and there are jobs in its queue, the dispatching rule is used to prioritise the queued jobs. Then, the machine starts processing the job with the highest priority, as depicted in lines 10-14. After processing all the required jobs, the algorithm terminates, and the objective value is estimated using the generated schedule, as shown in lines 17 and 18.

Inputs: job shop configuration (n), dispatching rule (i)

Output: the objective value $obj_{i,n}$

1: Current time = 0

2: **while** $there\ are\ unscheduled\ jobs$ **do**

3: **if** there is a job released **do**

4: **if** the next machine in its route is idle **then**

5: Start processing the first operation of the job

6: **else**

7: The job enters the queue of the first machine in its route

8: **end if**

9: **end if**

10: **if** there is an idle machine **do**

11: Calculate priority values for all queued operations using the rule i

12: Start processing the operation with the highest priority value

13: Update the ready time of the machine and the job's next operation

14: **end if**

15: Current time += 1

16**: end while**

17: Calculate the objective value $obj_{i,n}$

18: **return** $obj_{i,n}$

Figure 5.3: Meta-algorithm of scheduling heuristics.

Three job shop scheduling objectives are considered including Total Weighted Tardiness (TWT), Mean Tardiness (MT), and Mean Flow Time (MFT). The training and testing scenarios are illustrated in Table 5.1. The tuple (u, t) represents the scenario where the utilization level is $u\%$ and the tightness factor is t. The tightness factor in the case of the TWT objective is slightly looser than in the case of the MT objective since the TWT considers not only the job due date as in the MT objective but also the job weight. Also, the utilization level varies greatly in the MFT scenarios compared with the tightness factor, which remains constant.

Table 5.1 Parameter settings of the training and testing scenarios

Obj.	Training scenarios	Testing scenarios
TWT	(80, 3), (80, 6), (80, 8), (90, 2), (90, 6), (90, 8)	(80, 3), (80, 4), (80, 5), (80, 6), (80, 7), (80, 8), (85, 3), (85, 4), (85, 5), (85, 6), (85, 7), (85, 8), (90, 3), (90, 4), (90, 5), (90, 6), (90, 7), (90, 8), (95, 3), (95, 4), (95, 5), (95, 6), (95, 7), (95, 8)
MT	(80, 1.5), (80, 3), (85, 4), (90, 1.5), (90, 3), (90, 4)	(80, 1.5), (80, 2), (80, 2.5), (80, 3), (80, 3.5), (80, 4), (85, 1.5), (85, 2), (85, 2.5), (85, 3), (85, 3.5), (85, 4), (90, 1.5), (90, 2), (90, 2.5), (90, 3), (90, 3.5), (90, 4), (95, 1.5), (95, 2), (95, 2.5), (95, 3), (95, 3.5), (95, 4)
MFT	(70, 3), (85, 3), (97.5, 3)	(70, 3), (72.5, 3), (75, 3), (77.5, 3), (80, 3), (82.5, 3), (85, 3), (87.5, 3), (90, 3), (92.5, 3), (95, 3), (97.5, 3)

For a specific objective o, the average normalized objective value of a rule i through training scenarios N_o is taken as the rule fitness value $fitness_i$ as shown in Equation 5.1, where ref refers to a reference rule. The WATC, Covert, and PT + WINQ rules are used as reference rules for the purpose of normalization as they are efficient human-made rules in minimising the TWT, MT, and MFT objectives, respectively (Sels et al., 2012b). For each testing configuration, 20 replications are used to assess the generality of the evolved rules under unseen scenarios. Finally, the percentage change PC_o in the performance of a given method with respect to a reference rule $obj_{ref,n}$ is estimated by Equation 5.2, where $obj_{best,n}$ is the objective value of the best-evolved rule.

$$fitness_i = \frac{1}{|N_o|}\sum_{n=1}^{|N_o|}\frac{obj_{i,n}}{obj_{ref,n}} \quad 5.1$$

$$PC_o = \frac{1}{|N_o|}\sum_{n=1}^{|N_o|}\frac{obj_{ref,n} - obj_{best,n}}{obj_{ref,n}} \times 100 \qquad 5.2$$

5.3.2 Design Of the Experiments

In order to assess the effectiveness of the proposed approach, four algorithms from the literature are developed for the sake of comparison. The developed algorithms are the standard genetic programming algorithm (SGP) (J. R. Koza, 1994b), the standard gene expression programming algorithm (GEP) (Nie et al., 2013b), the GEP algorithm with the literature feature selection approach (HGEP), and the GEP with the proposed feature selection approach (PGEP). The framework of the four algorithms is depicted in Figure 5.4, and the exclusive operations of the proposed approach are highlighted by a dashed line. The reasons for choosing these four algorithms are as follows:

I. Comparing the GP with the GEP algorithms: to analyse the actual difference between the two standard algorithms for the dynamic job shop scheduling settings understudy.
II. Comparing the GEP with the PGP: to verify that the proposed feature selection approach reduces the size of GEP rules and computational time.
III. Comparing the HGP with the PGP: to verify whether the proposed modification in the current attribute vector and feature selection ability reduce the size of evolved rules without sacrificing solution quality.

All algorithms start by generating a population of dispatching rules using the set of terminals and functions given in Table 4.1. The SGP algorithm uses the tree-based representation to initialise expression trees, while the HGEP, GEP, and PGEP adopt the K-expression to obtain expression trees for fixed-length linear chromosomes. Moreover, the proposed representation is employed to initialise attribute vectors for the individuals created using the PGEP algorithm, whereas attribute vectors are initialised using the representation given in (Nguyen et al., 2018c) for the HGEP algorithm. Then, the performance of evolved rules is estimated through a set of training instances using the developed simulation model. The rules with high solution quality are selected to form the parents of the next generation using the tournament method. The offspring are created by applying the GP genetic operators to the SGP individuals, and the GEP genetic operators to the HGEP, GEP, and PGEP individuals. An additional mutation operator is applied in the HGP algorithm to invert the state of a randomly selected attribute from the attribute vector. In the case of the PGEP algorithm, attribute vector

mutations are applied using activation probabilities estimated from the prior generation, where the activation probabilities (weights) of the terminals are updated using a subset of the best-selected rules. If the maximum number of generations is met, the algorithm terminates and returns the rule with the best objective value.

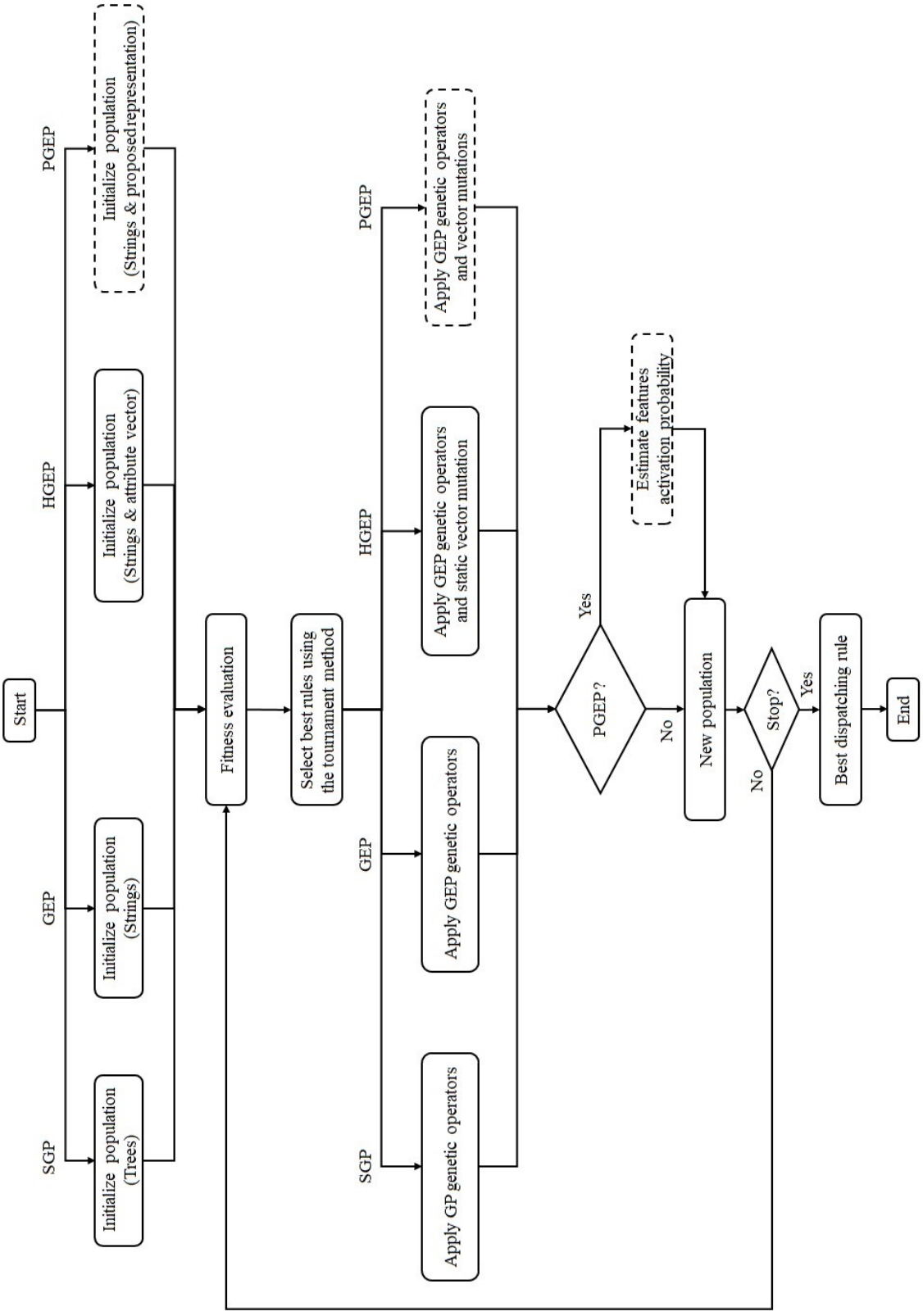


Figure 5.4: Framework of the four algorithmic experiments.

5.3.3 Parameter Settings

The algorithms were implemented in Python 3.7 on a system with an Intel(R) Xeon(R) 3.6GHz and 64 GB of RAM. In order to provide a fair comparison between the four algorithms, similar parameters are shared between them. A population of 750 dispatching rules is initialised, and the tournament size is set to 5 for the four algorithms. Table 5.2 shows the parameter settings for the four algorithms. Most of these parameters are commonly used in previous studies (Nie et al., 2013b; Shady et al., 2021c). Since the GP and GEP algorithms use different representations to express dispatching rules, it is crucial to define appropriate initialisation parameters that can create rules of the same size in the first generation. It has been observed from preliminary experiments that using a maximum depth of 8 for the GP algorithms results in the same average rule length as GEP rules generated with a chromosome comprising two genes and a gene head length equal to 26. The activation probability at the initial generation $Prob_{int}$ and the number of selected individuals $|\mathbb{S}|$ are the new parameters introduced for the PGEP algorithm. The $|\mathbb{S}|$ and $Prob_{int}$ parameters are set to 300 and 0.5, respectively. Regarding HGP and HGEP algorithms, the attribute mutation probability of 0.5 is used because it had the best results and is also recommended in (Nguyen et al., 2018c). Finally, the termination condition for all algorithms is set to 50 generations, and 20 independent runs of each algorithm are performed for each objective.

Table 5.2 Parameter settings for the four algorithms

SGP	**Values**	**GEP, HGEP, and PGEP**	**Values**
Maximum tree depth	8	Head length	26
Crossover rate	0.85	Number of genes	Two genes
		linking function	Addition
Mutation rate	0.1	Mutation, IS, RIS, and Gene transposition rates	0.03%, 0.1%, 0.1%, and 0.1%
Elitism rate	0.05	One-point, two-point, and Gene recombination rates	0.2%, 0.5%, and 0.1%

5.4 Results

Due to the stochastic nature of evolutionary algorithms, the Wilcoxon rank-sum test with a significance level of 0.05 was used to examine the statistical difference in the performance of the algorithms. In the following results, the statistical results obtained from comparing PGEP with the other three algorithms are represented by a tuple next to the PGEP results (SGP versus PGEP, GEP versus PGEP, and HGEP versus PGEP). Also, the symbols "+", "-" and "=" indicate that the corresponding result is significantly better, worse than, or similar to its counterpart. It is noteworthy that this chapter is not primarily focused on improving the performance of evolved rules, but on generating rules with small sizes and high interpretability without sacrificing efficiency. Smaller rules are mathematically simpler than complex ones, and thus contribute positively to reducing the computational expenses of evolutionary algorithms and are feasible to implement in the industry

5.4.1 Training Performance

Five performance measures were considered to compare the performance of the four algorithms across generations during the training phase. The performance measures are the percentage change in objective value, computational time, mean rule length, the average number of absent terminals, and the average number of active terminals. For all performance measures used, a smaller value indicates better performance, except for the percentage change objective, where larger values indicate better performance. Table 5.3 shows the mean and standard deviation of the four algorithms under the TWT, MT, and MFT objectives. Figure 5.5 (a1), (a2), and (a3) show the percentage change in the TWT, MT, and MFT objectives across generations, respectively. Regarding the performance of evolved rules, there is no significant difference in the percentage change in the objective values of the PGEP algorithm compared with the other methods under MFT objectives, as shown in Table 5.3. In terms of the MT objective, the PGEP algorithm had lower solution quality compared with the SGP algorithm, whereas it had the same performance as GEP and HGEP algorithms. In addition, the PGEP algorithm obtained significantly worse results compared with the SGP algorithm and better results compared with the HGEP algorithm in the TWT objective. Finally, the rules generated using the SGP algorithm had better solution quality compared with the GEP algorithm for the MT and TWT objective.

In terms of computational costs, although the HGEP algorithm had a smaller computational time compared with the GEP algorithm in the TWT and MT objectives, this reduction was at the expense of the quality of evolved rules as the HGEP had significantly worse results compared with the other algorithms related to the TWT and MT objectives. In contrast, the computational time for the PGEP algorithm is significantly smaller than the three other algorithms under all objective functions while achieving high solution quality. This finding indicates that feature selection methods that obtained promising results in the GP literature do not guarantee the same high performance when integrating with the GEP algorithm. The computational time of the PGEP algorithm is greatly smaller than the SGP and GEP, respectively, as shown in Figure 5.5 (b1), (b2), (b3), and Table 5.3. These results demonstrate the ability of the proposed feature selection approach to reduce the computational requirements of both the GP and GEP algorithm. Finally, the compactional time of the GEP algorithm is significantly smaller than the SGP algorithm in the three objectives.

Table 5.3 Performance measures in the training phase

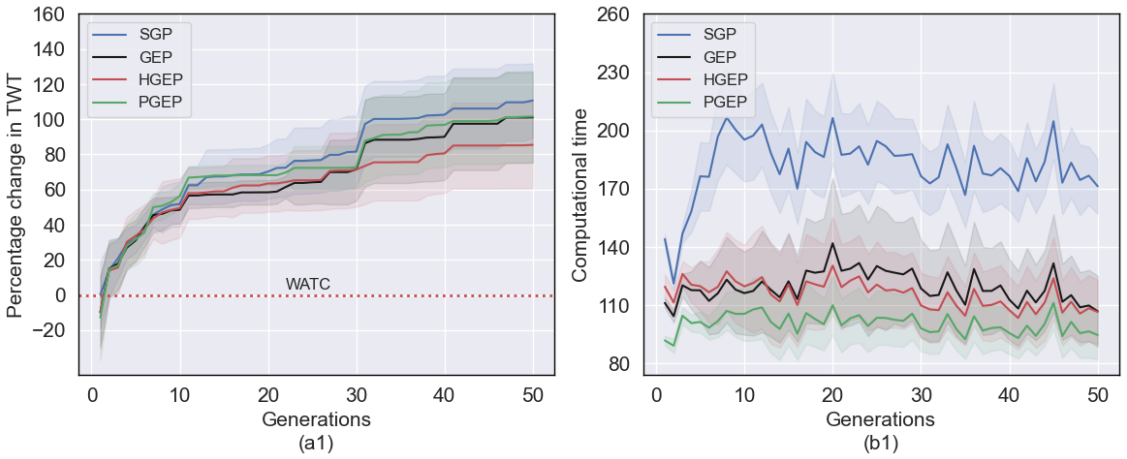
Perf. Meas.	Obj.	SGP	GEP	HGEP	PGEP
Percentage change	TWT	110.71 ± 21.67	101.1 ± 25.86	85.47 ± 24.59	101.61 ± 28.68
					(-, =, +)
	MT	125.65 ± 2.46	122.11 ± 3.85	121.68 ± 5.7	122.54 ± 3.63
					(-, =, =)
	MFT	56.72 ± 0.5	56.2 ± 0.26	56.33 ± 0.3	55.5 ± 0.63
					(=, =, =)
Computational time	TWT	181.73 ± 10.93	119.53 ± 22.26	115.64 ± 12.66	100.52 ± 11.86
					(+, +, +)
	MT	154.55 ± 12.79	116.23 ± 22.56	107.84 ± 15.07	101.53 ± 11.65
					(+, +, +)
	MFT	119.26 ± 8.2	69.0 ± 8.55	71.78 ± 8.02	63.33 ± 11.72
					(+, +, +)
Mean rule length	TWT	16.83 ± 3.16	11.13 ± 2.24	9.07 ± 1.96	9.11 ± 1.61
					(+, +, =)
	MT	14.62 ± 2.67	10.76 ± 2.61	9.67 ± 1.71	8.77 ± 1.36

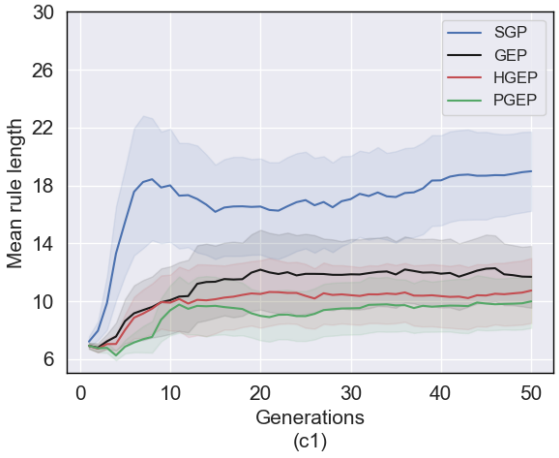
					(+, +, +)
	MFT	11.87 ± 1.51	5.78 ± 0.88	6.12 ± 0.89	5.71 ± 1.2
					(+, +, +)
Average number of absent terminals	TWT	5.58 ± 1.84	7.09 ± 1.71	7.67 ± 1.85	8.05 ± 1.25
					(+, +, +)
	MT	6.05 ± 1.67	7.0 ± 2.07	7.27 ± 1.4	7.77 ± 1.17
					(+, +, +)
	MFT	7.5 ± 1.23	9.08 ± 0.9	8.87 ± 0.93	8.96 ± 1.29
					(+, -, +)
Average number of active terminals	TWT	7.42 ± 1.84	5.91 ± 1.71	3.85 ± 1.23	3.31 ± 0.63
					(+, +, +)
	MT	6.65 ± 1.67	6.0 ± 2.07	4.11 ± 0.98	3.46 ± 0.6
					(+, +, +)
	MFT	5.5 ± 1.23	3.92 ± 0.9	3.39 ± 0.62	2.73 ± 0.78
					(+, +, +)

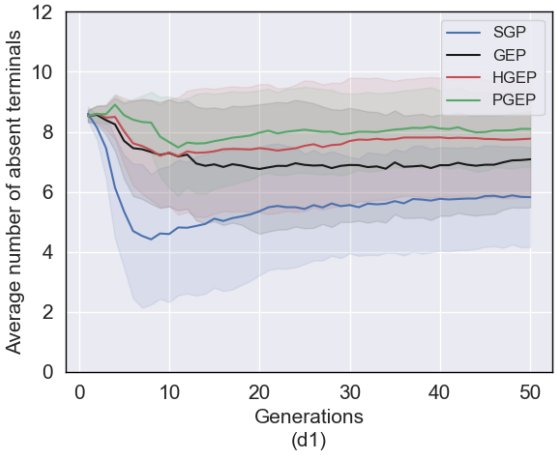
As expected, the reduction in computational costs resulted from a decrease in the average length of evolved rules, as illustrated in Table 5.3 and Figure 5.5 (c1), (c2), and (c3). The rules evolved using the PGEP algorithm had the smallest length compared with the three algorithms in the three objectives. This supports our hypothesis that integrating the proposed approach with the restricted search space of the GEP algorithm is beneficial in reducing the size of created rules. In addition, the GEP algorithm generated rules with significantly smaller sizes compared with the SGP algorithm which is similar to the findings in the literature. The evolved rules using the HGEP algorithm were smaller than the GEP algorithm for the MT and TWT objectives. In contrast, GEP evolved rules had smaller sizes compared with the HGEP algorithm with respect to the MFT objective.

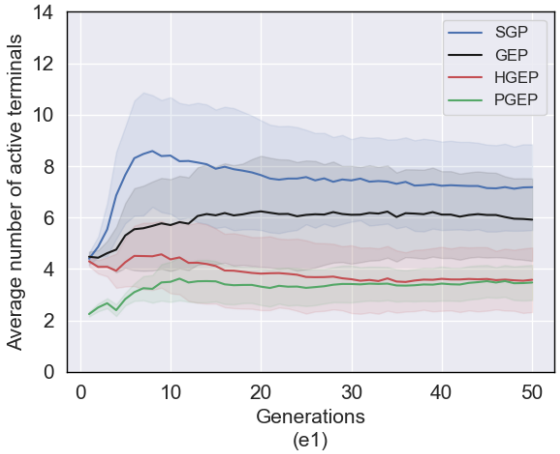
Since there are three states (active, inactive, and absent) for each terminal in the proposed representation while there are only two states (absent and active) in the representation of the SGP and GEP algorithms, the number of inactive terminals was excluded from the comparison. Afterwards, the average number of absent and active terminals of the evolved rules using the four algorithms is compared. As depicted in Figure 5.5 (d1-e1), (d2-e2), and (d3-e3), the use of the proposed feature selection

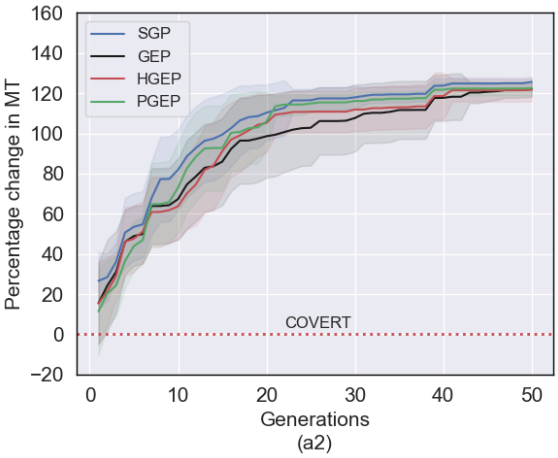
approach supports the PGEP algorithm in increasing the number of absent terminals and reducing the number of active terminals compared with the GEP algorithm. Specifically, the proposed algorithm (PGEP) obtained significantly more absent terminals and fewer active terminals compared with the other algorithms under the TWT, MT, and MFT objectives. These results demonstrate the ability of the proposed feature selection approach to reduce the size of evolved rules, which greatly reduces the training time of the GP and GEP algorithms. In addition, the PGEP algorithm was able to evolve rules with acceptable performance in more understandable structures and affordable computational costs. Similarly, the HGEP algorithm had significantly larger number of absent terminals and smaller number of active terminals compared with the GEP algorithm in the TWT and MT objective. However, in the case of the MFT objective, the GEP algorithm generated rules with more absent terminals compared with the HGEP algorithm. Finally, the GEP algorithm evolved rules with larger number of absent terminals and fewer active terminals in all objectives compared with the SGP algorithm.

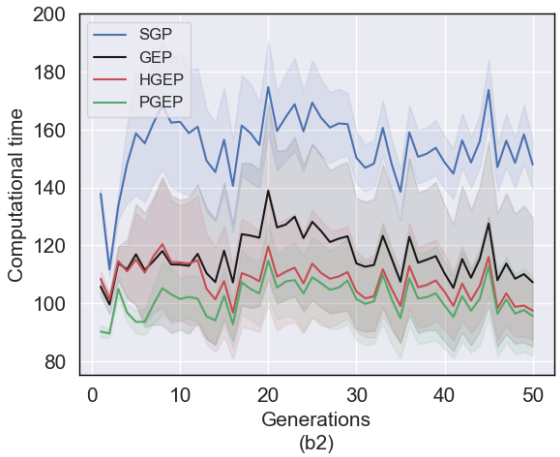


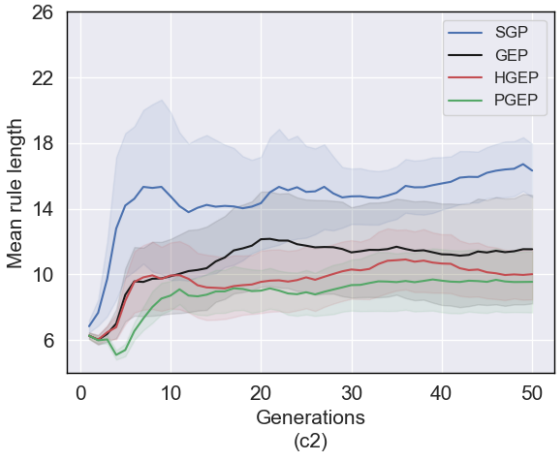


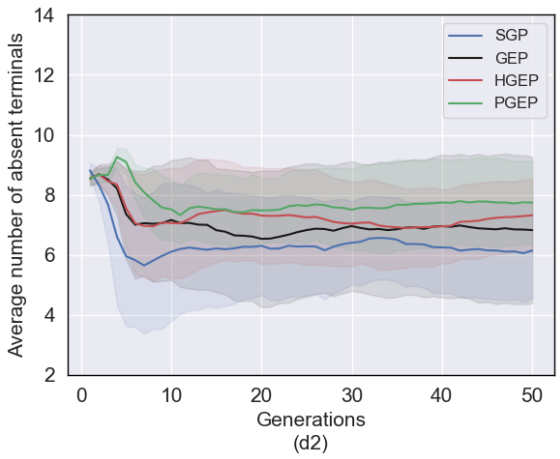

Average number of absent terminals
Generations
(d1)
SGP
GEP
HGEP
PGEP


Average number of active terminals
Generations
(e1)
SGP
GEP
HGEP
PGEP

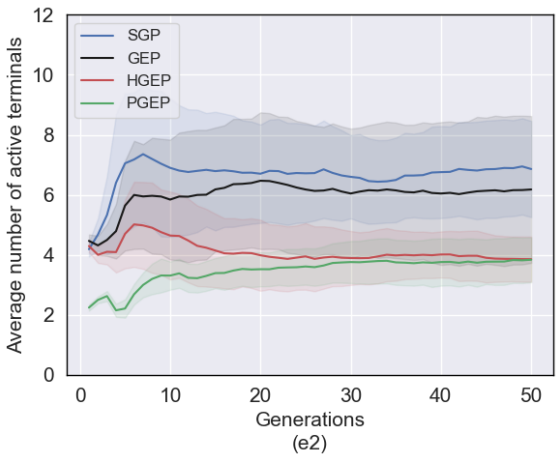

Percentage change in MT
COVERT
Generations
(a2)
SGP
GEP
HGEP
PGEP


Computational time
Generations
(b2)
SGP
GEP
HGEP
PGEP

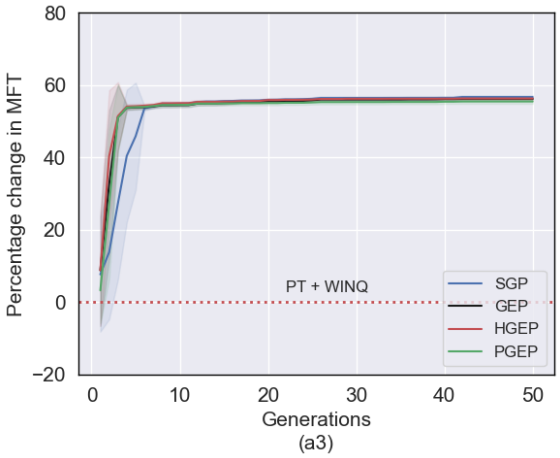

Mean rule length
Generations
(c2)
SGP
GEP
HGEP
PGEP


Average number of absent terminals
Generations
SGP
GEP
HGEP
PGEP

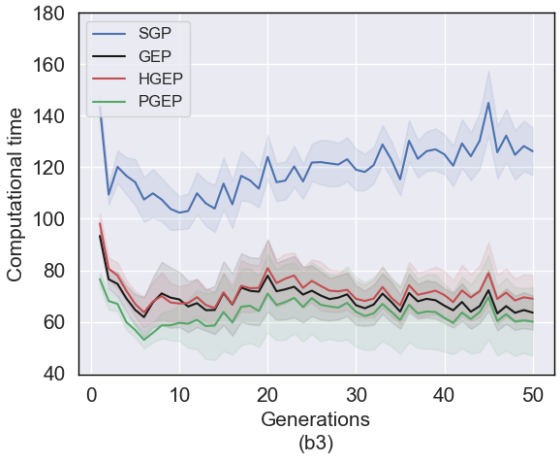
(d2)


Average number of active terminals
Generations
SGP
GEP
HGEP
PGEP

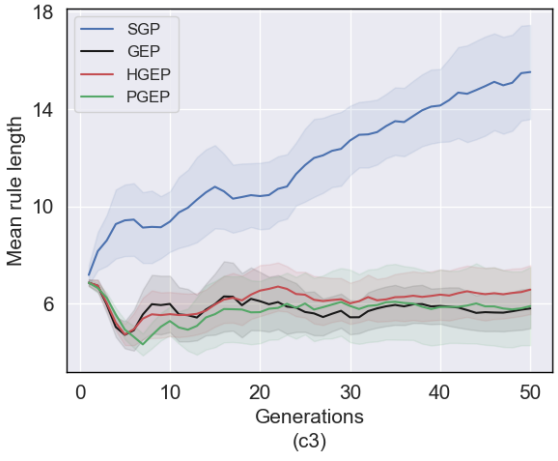
(e2)


Percentage change in MFT
PT + WINQ
Generations
SGP
GEP
HGEP
PGEP

(a3)


Computational time
Generations
SGP
GEP
HGEP
PGEP

(b3)


Mean rule length
Generations
SGP
GEP
HGEP
PGEP

(c3)

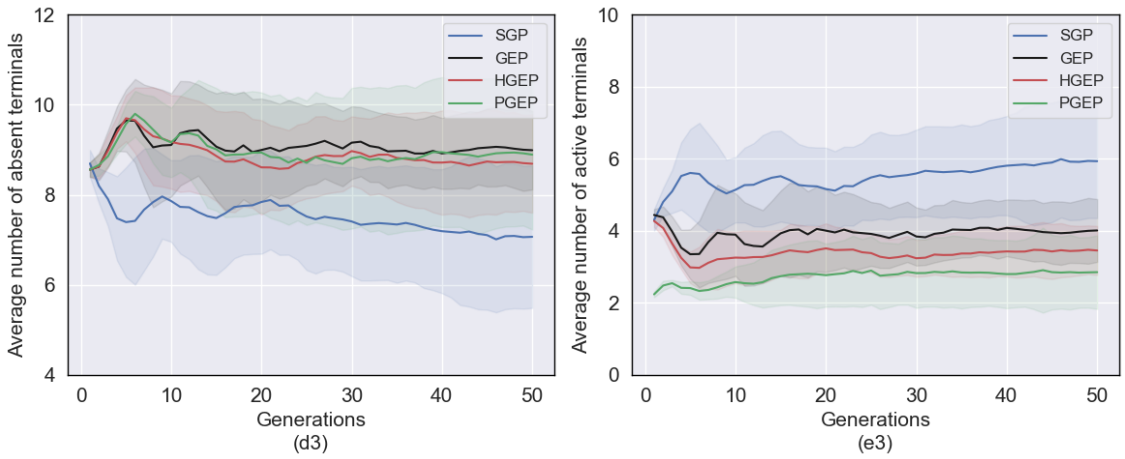


Figure 5.5: The performance of the four algorithms during the training phase for the three objectives. Figures (a1), (b1), (c1), (d1), (e1) are for the TWT objective. Figures (a2), (b2), (c2), (d2), (e2) are for the MT objective. Figures (a3), (b3), (c3), (d3), (e3) are for the MFT objective.

5.4.2 Testing Performance

In order to ensure that the algorithms did not overfit the training data and evolved rules are applicable to unseen scenarios, the performance of the best 20 dispatching rules for each algorithm is evaluated through several testing scenarios. Table 4.6 (a), (b), and (c) show the mean and standard deviation of the four algorithms in the TWT, MT, and MFT objectives, respectively. The last row of each table provides a summary of the statistical results using a tuple (a, b, c) where a, b, and c represent the number of times a certain method wins (significantly better), draws (no significant difference), and loses (significantly worse) respectively, against the PGEP algorithm.

Regarding the TWT objective, the PGEP algorithm showed better performance compared with the results of SGP, GEP and HGEP in 3, 4 and 24 scenarios, respectively. The rules evolved using the PGEP algorithm had similar performance to the rules of the SGP, GEP and HGEP algorithms in 9, 17 and zero scenarios, respectively. As expected from the training performance, the HGEP algorithm suffered from premature convergence as the PGEP algorithm significantly outperformed it in all testing scenarios related to the TWT objective. For the MT objective, The PGEP and GEP algorithms had relatively similar mean tardiness results, where there was no significant difference in 20 scenarios. In addition, the PGEP had significantly higher solution quality in 4 scenarios compared with the GEP algorithm. In contrast, the SGP

rules significantly outperformed the PGEP rules in 13 scenarios and had the same results in 11 scenarios. Moreover, the PGEP algorithm outperformed the HGEP algorithm in 12 scenarios and had similar solution quality in the other 12 scenarios without any loss.

Although the SGP algorithm got significantly smaller TWT values compared with the PGEP algorithm in 12 scenarios and smaller MT values in 13 scenarios, these results would be inverted if the algorithm running time was used as a stop criterion because the PGEP had significantly smaller computational budget. It is noteworthy that the four algorithms evolved rules with the same MFT values revealing that the MFT objective was an easier objective to handle than the TWT and MT objectives. In other words, although the MFT objective demonstrated the difference between the four algorithms with respect to computational time, rule size, and the number of active and absent terminals, it could not distinguish the difference in the quality of evolved rules in both training and testing scenarios. Lastly, the obtained results demonstrate that the feature selection approach proposed for the GEP algorithm does not compromise the solution quality of generated rules in favour of reducing their sizes.

Table 5.4 Mean and standard deviation of the considered methods in the testing phase. (a): the TWT objective, (b): the MT objective, and (c): the MFT objective.

(a): scenarios	SGP	GEP	HGEP	PGEP
(3, 0.8)	56726.01 ± 11241.72	58022.72 ± 15863.38	101253.44 ± 48089.76	57663.35 ± 12422.92 (=, =, +)
(3, 0.85)	149354.06 ± 28496.28	153799.87 ± 37593.35	208837.32 ± 71059.64	149419.37 ± 27099.19 (=, =, +)
(3, 0.9)	363986.61 ± 69343.26	377615.16 ± 86768.73	436724.15 ± 110828.44	362972.51 ± 64786.78 (=, +, +)
(3, 0.95)	865385.5 ±177930.55	879876.83 ± 231526.76	926905.66 ± 194164.66	871979.63 ± 168267.7 (-, =, +)
(4, 0.8)	4945.72 ± 1895.55	4927.19 ± 2385.55	37948.44 ± 35052.87	5578.71 ± 2543.36 (-, -, +)
(4, 0.85)	28072.82 ± 8561.04	29034.58 ± 11572.88	91360.33 ± 68176.96	30037.94 ± 10175.36 (-, =, +)
(4, 0.9)	134081.75 ± 34149.56	141619.04 ± 41753.05	245387.8 ± 123239.66	138129.97 ± 36348.15 (-, =, +)
(4, 0.95)	531832.68± 120474.09	546325.44 ± 138968.32	668262.88 ± 192167.13	541882.87 ± 110754.28

				(-, =, +)
(5, 0.8)	277.4 ± 180.26	259.05 ± 197.47	19879.19 ± 21389.5	291.75 ± 253.13 (=, =, +)
(5, 0.85)	2752.09 ± 1521.91	2766.84 ± 1787.79	49617.1 ± 51215.34	3252.21 ± 1916.95 (-, -, +)
(5, 0.9)	32349.83 ± 14163.97	34221.21 ± 16185.79	143029.62 ± 119675.18	35265.81 ± 15445.64 (-, =, +)
(5, 0.95)	283197.38 ± 77804.52	292025.79 ± 77245.89	475756.62 ± 214144.53	294605.52 ± 74624.15 (-, =, +)
(6, 0.8)	57.92 ± 41.88	51.74 ± 39.73	11827.85 ± 13413.04	44.81 ± 43.71 (+, +, +)
(6, 0.85)	210.87 ± 143.39	153.31 ± 165.15	31947.81 ± 35641.5	155.32 ± 188.62 (=, =, +)
(6, 0.9)	4876.26 ± 3278.33	5124.12 ± 3746.84	94100.68 ± 98160.64	5880.26 ± 4082.97 (-, -, +)
(6, 0.95)	125112.34 ± 46205.63	127780.5 ± 40939.7	339214.12 ± 228857.15	133341.03 ± 45107.48 (-, =, +)
(7, 0.8)	33.18 ± 31.43	32.1 ± 29.57	7489.3 ± 8887.08	26.7 ± 27.43 (+, +, +)
(7, 0.85)	32.9 ± 33.08	29.12 ± 30.47	22435.81 ± 25958.95	32.09 ± 33.33 (=, =, +)
(7, 0.9)	417.13 ± 478.79	439.21 ± 538.48	71061.12 ± 80981.05	488.9 ± 528.63 (=, =, +)
(7, 0.95)	43338.93 ± 23040.37	43695.22 ± 20922.73	251006.94 ± 222340.49	47917.42 ± 23663.5 (-, =, +)
(8, 0.8)	20.97 ± 23.14	20.4 ± 20.88	4753.67 ± 6085.89	15.82 ± 17.65 (+, +, +)
(8, 0.85)	19.86 ± 22.81	19.63 ± 23.92	15989.11 ± 19053.37	21.81 ± 28.28 (=, =, +)
(8, 0.9)	54.88 ± 80.95	49.88 ± 82.99	55365.89 ± 64569.62	39.17 ± 61.37 (=, =, +)
(8, 0.95)	10388.34 ± 7859.3	10465.74 ± 6952.21	200628.42 ± 208898.31	12264.97 ± 8102.39 (-, =, +)
Summary	(12, 9, 3)	(3, 17, 4)	(0, 0, 24)	

(b): scenarios	SGP	GEP	HGEP	PGEP
(1.5, 0.8)	128.83 ± 6.24	133.3 ± 9.7	129.69 ± 6.35	129.47 ± 5.52 (=, +, =)
(1.5, 0.85)	177.33 ± 8.26	183.87 ± 13.59	178.53 ± 10.54	177.95 ± 7.39 (=, +, =)
(1.5, 0.9)	254.34 ± 12.31	265.97 ± 22.1	256.23 ± 16.01	255.02 ± 10.93 (=, +, =)
(1.5, 0.95)	405.73 ± 21.63	424.06 ± 40.31	404.81 ± 23.84	404.72 ± 18.21 (=, +, =)
(2, 0.8)	73.33 ± 4.76	76.53 ± 6.09	76.89 ± 7.27	74.78 ± 4.47 (=, =, =)
(2, 0.85)	116.6 ± 7.15	122.92 ± 10.01	120.55 ± 10.99	119.56 ± 6.88 (=, =, =)
(2, 0.9)	190.14 ± 10.8	201.47 ± 16.71	194.5 ± 15.99	194.7 ± 10.63 (=, =, =)
(2, 0.95)	338.86 ± 20.42	357.53 ± 35.73	339.87 ± 24.46	342.15 ± 17.79 (=, =, =)
(2.5, 0.8)	34.61 ± 2.95	36.28 ± 3.67	42.89 ± 9.12	36.46 ± 3.69 (-, =, +)
(2.5, 0.85)	68.93 ± 5.3	73.48 ± 6.41	78.17 ± 12.34	72.86 ± 6.4 (-, =, +)
(2.5, 0.9)	135.36 ± 9.7	145.85 ± 12.66	145.22 ± 17.41	142.51 ± 10.74 (-, =, =)
(2.5, 0.95)	280.03 ± 18.48	297.62 ± 28.92	285.35 ± 25.49	287.8 ± 17.74 (=, =, =)
(3, 0.8)	12.94 ± 1.88	13.9 ± 2.62	23.21 ± 10.74	14.17 ± 2.8 (-, =, +)
(3, 0.85)	35.52 ± 3.67	38.43 ± 4.99	49.36 ± 14.13	39.1 ± 5.49 (-, =, +)
(3, 0.9)	91.04 ± 8.41	98.49 ± 10.23	106.15 ± 19.35	97.89 ± 10.23 (-, =, +)
(3, 0.95)	226.65 ± 17.31	242.61 ± 23.49	237.91 ± 27.24	236.14 ± 17.66 (=, =, =)
(3.5, 0.8)	3.77 ± 1.01	4.21 ± 1.4	13.19 ± 9.67	4.43 ± 1.61 (-, =, +)
(3.5, 0.85)	15.65 ± 2.3	17.11 ± 3.64	30.91 ± 15.48	17.89 ± 4.02 (-, =, +)
(3.5, 0.9)	55.38 ± 6.06	60.97 ± 8.79	76.91 ± 21.79	62.09 ± 9.52 (-, =, +)
(3.5, 0.95)	177.78 ± 15.2	192.04 ± 19.5	197.69 ± 28.93	190.06 ± 18.25 (=, =, =)
(4, 0.8)	0.88 ± 0.56	1.03 ± 0.7	8.34 ± 7.92	1.14 ± 0.8 (-, =, +)
(4, 0.85)	5.72 ± 1.4	6.48 ± 2.2	20.1 ± 14.62	7.03 ± 2.48 (-, =, +)
(4, 0.9)	29.79 ± 4.73	34.38 ± 7.24	54.8 ± 24.09	35.81 ± 8.07 (-, =, +)
(4, 0.95)	135.44 ± 13.76	147.59 ± 16.95	162.72 ± 31.55	146.99 ± 17.04 (-, =, +)
Summary	(13, 11, 0)	(0, 20, 4)	(0, 12, 12)	

(c): scenarios	SGP	GEP	HGEP	PGEP
(3, 0.7)	283.93 ± 1.62	283.74 ± 1.56	284.15 ± 1.89	284.18 ± 1.56 (=, =, =)
(3, 0.725)	295.42 ± 1.92	295.31 ± 1.77	295.76 ± 2.32	295.56 ± 1.89 (=, =, =)
(3, 0.75)	308.99 ± 2.05	308.7 ± 1.84	309.19 ± 2.61	308.73 ± 2.06 (=, =, =)
(3, 0.775)	324.31 ± 2.23	324.12 ± 2.29	324.46 ± 2.84	324.43 ± 2.27 (=, =, =)
(3, 0.8)	341.64 ± 2.45	341.7 ± 2.25	342.52 ± 3.41	342.22 ± 2.6 (=, =, =)
(3, 0.825)	362.51 ± 2.81	362.55 ± 2.61	363.13 ± 4.39	363.13 ± 2.95 (=, =, =)
(3, 0.85)	387.55 ± 3.34	387.53 ± 2.96	388.72 ± 5.58	388.85 ± 3.26 (=, =, =)
(3, 0.875)	417.62 ± 3.69	417.96 ± 3.82	419.16 ± 7.42	418.8 ± 3.65 (=, =, =)
(3, 0.9)	458.27 ± 5.0	459.49 ± 4.78	460.43 ± 8.75	459.61 ± 5.33 (=, =, =)
(3, 0.925)	513.2 ± 6.19	515.09 ± 5.51	517.09 ± 11.8	516.63 ± 5.93 (=, =, =)
(3, 0.95)	592.13 ± 8.97	593.92 ± 6.45	597.46 ± 15.84	595.51 ± 8.24 (=, =, =)
(3, 0.975)	701.35 ± 13.6	704.52 ± 8.6	709.27 ± 22.26	705.08 ± 11.54 (=, =, =)
Summary	(0, 12, 0)	(0, 12, 0)	(0, 12, 0)	

5.4.3 Insights Into the Best-Evolved Rules

The distribution of the terminals of the 20 best rules evolved using the four algorithms is plotted in Figure 5.6 (a), (b), and (c) representing the results obtained for the TWT, MT, and MFT objectives, respectively. Typically, extensive use of a particular terminal in high-performance rules indicates that it is extremely important to the objective used in the training phase. The total number of terminals used in the best TWT rules for the PGEP algorithm is smaller than those in the SGP, GEP, and HGEP by 58.13%, 29.91%, and 15.14%, respectively. Also, the gap between relevant and irrelevant terminals is wider in the PGEP results than in the other three methods. Similarly, the difference in the number of terminals between significant and insignificant terminals in the best-generated rules was more pronounced in the PGEP algorithm than in the GEP and HGEP algorithms. The most important terminals in the case of minimizing the TWT objective are PT, SL, JW, RO, and WINQ while Npt, WT, WR, JR, and OR terminals are insignificant as depicted in Figure 5.6 (a).

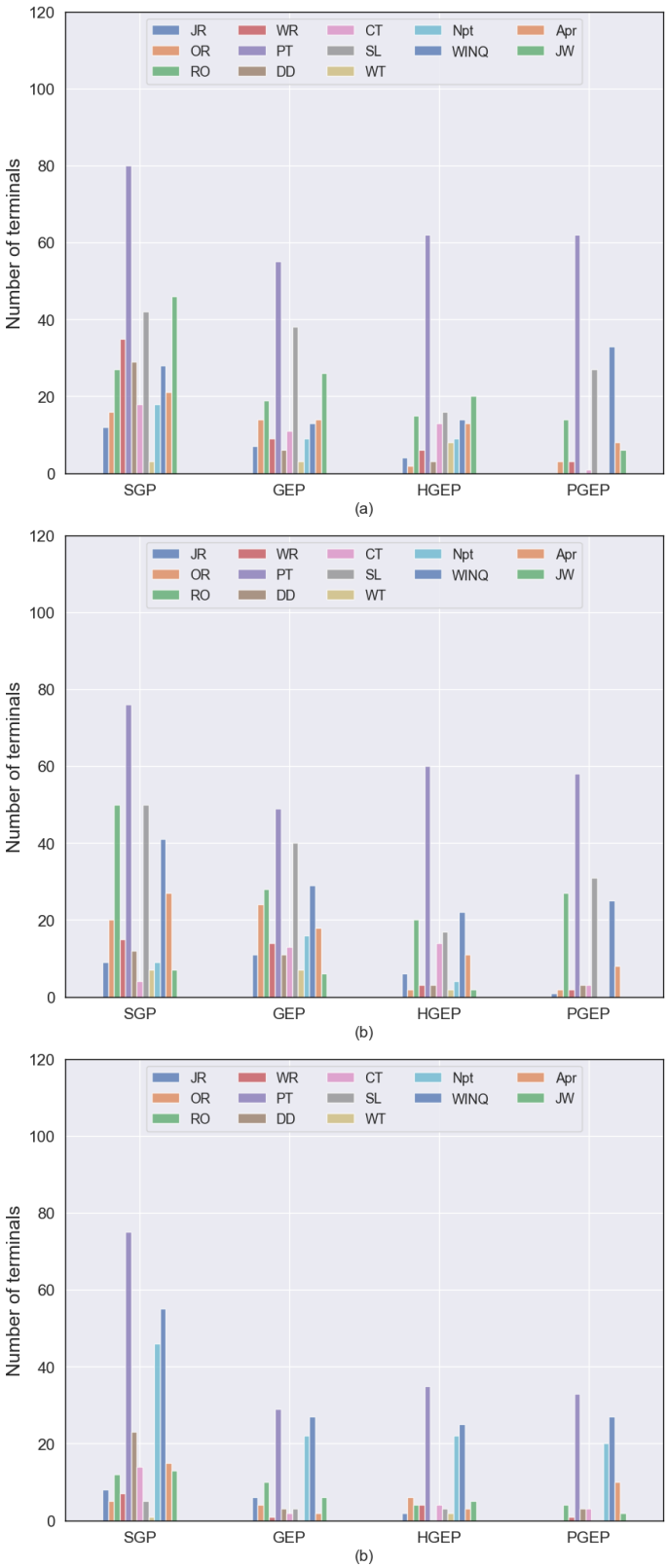


Figure 5.6: Terminals distribution in the 20 best rules for the four algorithms. (a): for the TWT objective, (b): for the MT objective, (c): for the MFT objective

Regarding the MT objective, the PGEP algorithm considered WT, Npt, and JW terminals to be neglectable, while the weight of RO and Apr terminals was increased. In

addition, the PGEP algorithm reduces the number of terminals in the best rules by 51.07%, 39.85%, and 3.61% compared with the SGP, GEP, and HGEP algorithms, respectively. In terms of the MFT objective, the PGEP rules have 63.08%, 13.45%, and 10.43% fewer terminals than the rules generated using the SGP, GEP, and HGEP algorithms. The most important terminals to reduce the mean flow time objective are PT, Npt, and WINQ, while redundant terminals include DD and SL. Consequently, four main findings can be drawn:

a) The weight of terminals varies according to the objective under study, and the selection ability of the GP and GEP algorithms without external feature selection mechanisms is relatively limited.
b) Although the rules generated using the GEP algorithm have relatively lower quality compared with the GP rules, the difference in computational time is significantly large which might inverse the results if the time is used as a stopping criterion.
c) The integration between the proposed feature selection approach and the fixed-length representation of the GEP algorithm can identify important features and exclude irrelevant features under different objective functions.
d) In contrast to the literature feature selection approach, the proposed approach reduces the size of evolved rules and speeds up the GEP algorithm without negatively affecting solution quality, and even improves quality in the case of MT objective.

The simplified mathematical version of the best PGEP rule evolved under the TWT objective is shown in Equation 5.3. The PT, WINQ, JW, SL, and RO terminals are active terminals in the evolved rule while the WR terminal is inactive. The remaining terminals are not used in the best TWT rule. Regarding the MT objective, Equation 5.4 shows the best PGEP rule in mathematical form. Four terminals are used to prioritize queued operations which are PT, WINQ, SL and RO indicating their great impact in minimizing the MT objective. The only disabled terminal is the JW terminal, while the rest of the terminals have been excluded. Equation 5.5 presents the mathematical function of the best PGEP rule used to minimize the MFT objective. The active terminals are Npt, WINQ, PT, Apr, RO, and PT, whereas the deactivated terminals include JR, DD, and JW.

$$TWT\ rule\ =\ 3.6 \times (PT\ +\ \frac{WINQ}{JW})\ +\ \frac{2\ \times\ SL}{RO} \qquad 5.3$$

$$MT\ rule\ =\ 2\times\ PT\ +\ WINQ\ +\ \frac{SL\ -\ 1.84}{RO} \quad 5.4$$

$$MFT\ rule\ =\ min(Npt, WINQ)\ +\ PT\ +\ min(Apr\ -\ RO, PT) \quad 5.5$$

Visualizing the mathematical formulation of the best-evolved rules provides some useful insights regarding important and irrelevant terminals for each objective. However, it is still challenging to know which characteristics of the operations will take the highest priority using these rules. Therefore, 20 decision situations were sampled from an actual simulation run and used to understand the phenotypic characterization of the best-evolved rules as recommended in (Hildebrandt and Branke, 2015a). The phenotypic characterization of a given rule demonstrates the impact of changing the values of the terminals of a set of jobs on their priority values. It is important to mention that these rules give high priority to jobs with lower priority values from the set of queued jobs. From these experiments, it was clear that the PGEP rule for the TWT objective gives high priority to jobs with a large number of remaining operations, low processing time, low slack, low work in the next queue, and high weight. Regarding the MT objective, the best PGEP rule favours jobs that have the same characteristics as the TWT objective except for job weight, which is an irrelevant factor. The PGEP rule that obtained the smallest MFT objective values assigns high priority values to jobs with low processing time, low average processing time of queued jobs, low processing time of next operation, and low work in the next queue.

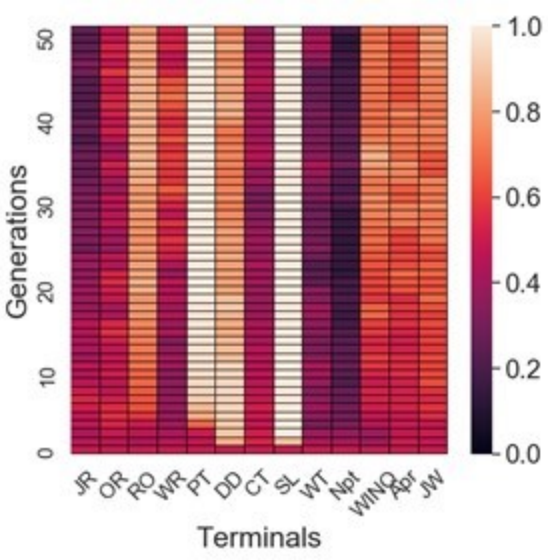
5.4.4 Further analysis of the proposed approach

The proposed feature selection approach is an online mechanism to guide GP and GEP algorithms in generating superior dispatching rules in compact structures as described in the previous and current chapters. Therefore, the activation probability (weight) of each terminal is tracked during the execution of the PGP algorithm introduced in the previous chapter and compared with the PGEP algorithm for two reasons.

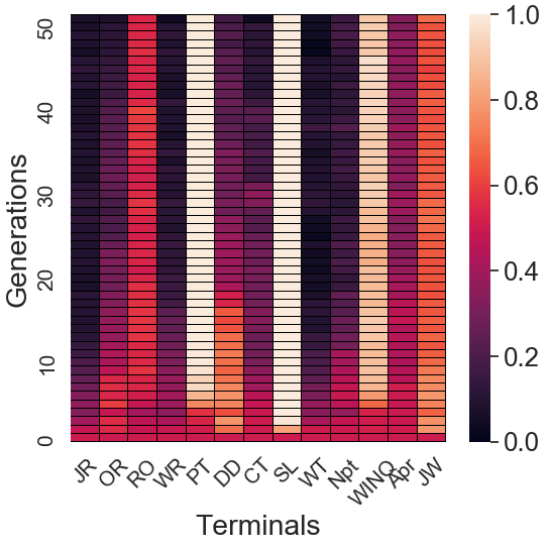
I. To ensure that the proposed approach is able to select important features and exclude redundant ones regardless of the evolutionary algorithm used. In other words, PGP and PGEP algorithms can identify the same sets of significant and redundant features for each objective.
II. To check whether the weights of the terminals differ across generations, supporting the main hypothesis that the importance of each terminal is not fixed throughout the run (binary discrimination is impractical).

The results obtained using the PGP and PGEP algorithms are shown in Figure 5.7. For the sake of convenience, the title below each sub-figure is represented as (a_b), where "a" denotes the objective function considered and "b" indicates the algorithm used. The colour of each cell in the heat maps represents the average activation probability of this terminal in 20 runs of the algorithm at a given generation. In addition, the average activation probability (weight) of the terminal across all generations for the PGP and PGEP algorithms under the three objectives is estimated and plotted in Figure 5.7 (summary).

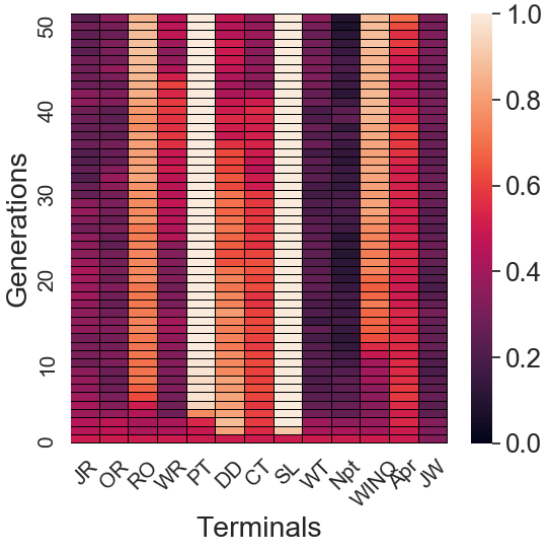
Although all algorithms had the same weight in the first generation, their weights varied greatly in subsequent ones. Regarding the TWT objective, the most important terminals (bright columns or values close to 1) in the PGP and PGEP algorithms are RO, PT, SL, WINQ, Apr, and JW. Although the DD terminal is an important terminal in the case of the PGP algorithm (0.8), it had a low activation probability using the PGEP algorithm (0.4). The inclusion of the SL terminal may be one of the reasons, as it can substitute some DD and CT terminals and thus indirectly affect their weights. For the MT objective, the most relevant terminals are RO, PT, SL, WINQ, and Apr while insignificant ones include JR, OR, WR, WT, Npt and JW. Regarding the MFT objective results shown in Figure 5.7 (MFT_PGP) and (MFT_PGEP), the set of significant terminals includes PT, Npt, WINQ, and Apr, while insignificant terminals include most of the due-date related terminals such as JR, OR, DD, SL, and WT. As shown in Figure 5.7 (summary), most terminals had the same or slightly different weight for both the PGP and PGEP algorithms under the same objective function. The difference in activation probability between the two algorithms for any terminal did not exceed 0.2, except for the DD terminal in the MT and TWT objectives and the WR terminal in the TWT objective. Therefore, it is clear that the proposed feature selection approach is able to identify significant terminals regardless of the evolutionary algorithm or objective function used.



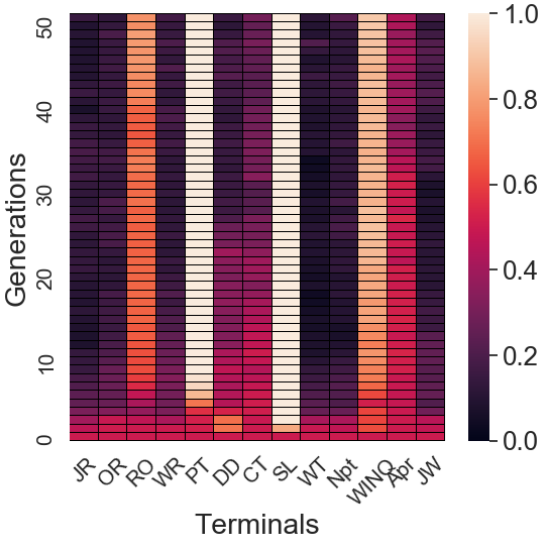
(TWT_PGP)



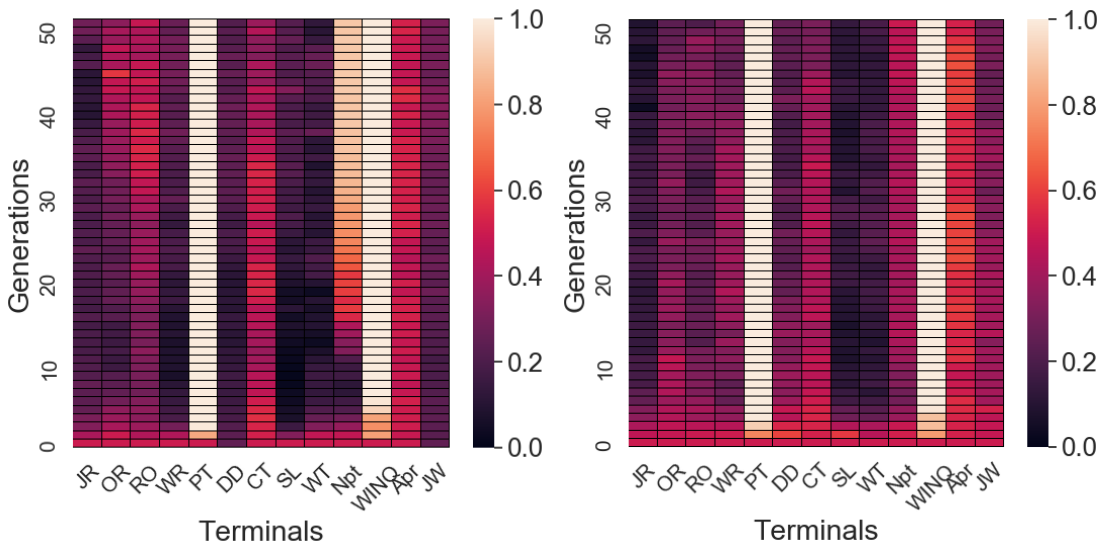
(TWT_PGEP)



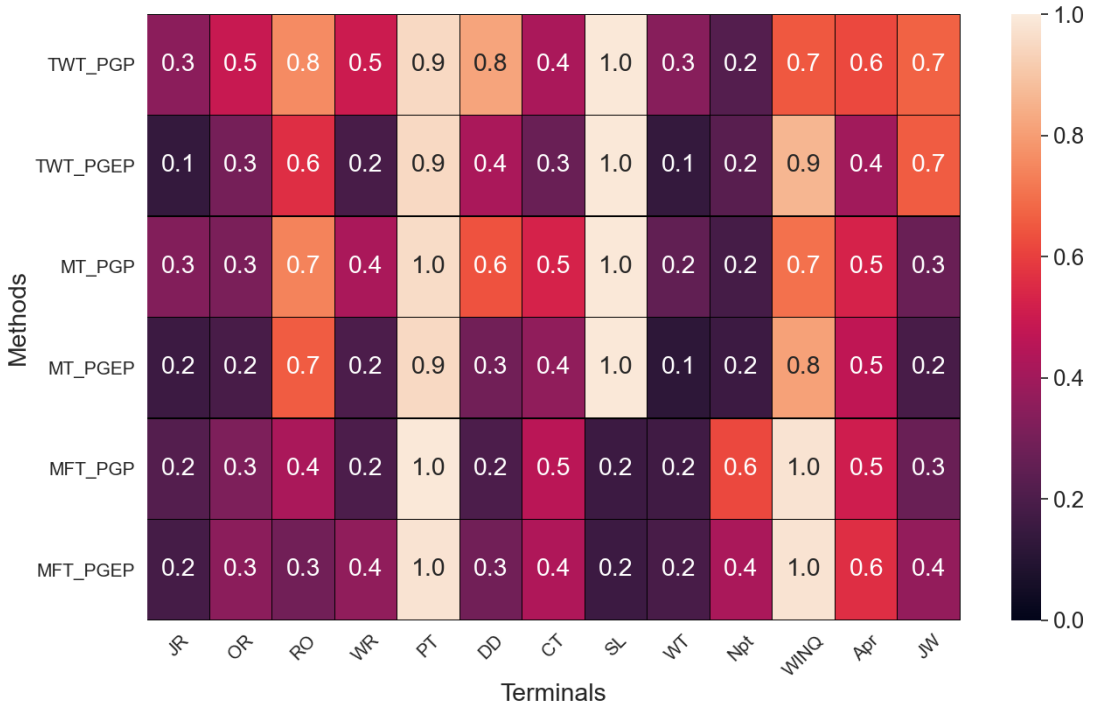
(MT_PGP)



(MT_PGEP)



(MFT_PGP) (MFT_PGEP)



(summary)

Figure 5.7: Matrix plot of the feature selection results using the PGP and PGEP algorithms under the TWT, MT, and MFT objectives.

5.5 Chapter Summary

This chapter proposed an online feature selection approach for the gene expression algorithm (PGEP) to evolve compact dispatching rules for the dynamic job shop scheduling problems. The proposed algorithm accelerated the search process by restricting the GP search space using the linear representation of the GEP algorithm as well as guiding the search to the most promising regions using the feature selection capability. This leads to generating high-quality scheduling rules at smaller sizes using only the most relevant terminals to the objective to be optimized.

Three algorithms from the literature were developed and compared with the proposed algorithm across various unseen job shop settings. The literature algorithms are the standard GP algorithm (SGP), gene expression programming algorithm (GEP), and GEP with the existing attribute vector (HGEP). In addition, three scheduling objective functions were investigated, including total weighted tardiness, mean tardiness, and mean flow time. Experimental results confirmed the ability of the PGEP algorithm in evolving rules with smaller sizes in a shorter computational time without sacrificing performance compared with literature methods. In addition, the distribution of terminals in the best generated rules for the four algorithms was compared. It was clear that the PGEP algorithm had the lowest number of terminals in the best rules, which reflected its ability to identify the most important terminals.

The PGEP best evolved rules were presented in mathematical form and their phenotypic characterizations were analysed using 20 decision situations sampled from an actual simulation run. The weight of terminals across the evolutionary process of the PGEP algorithm was recorded. Then, the obtained results were compared with those in the previous chapter to verify whether the performance of the feature selection approach changed based on the nature of the underlying evolutionary algorithm. Experimental results demonstrate the ability of the proposed feature selection approach to identify the same set of critical terminals regardless of the evolutionary algorithm used.

Chapter 6. SURROGATE−ASSISTED GEP FOR DYNAMIC JOB SHOP SCHEDULING PROBLEMS

6.1 Introduction

In contrast to chapters 3, 4, and 5 which focused on reducing the evolutionary training time by reducing the size of evolved rules, this chapter speeds up the learning process by reducing the computational time required to evaluate the performance of evolved rules. In other words, the approaches presented and analysed in this chapter are relatively independent of the GP and GEP algorithms and are primarily linked to the simulation model used to imitate the DJSSP under study. Typically, there are a large number of dispatching rules are generated at each evolutionary generation, and thus their fitness values are estimated across a range of training scenarios representing different job shop settings. Each of these scenarios is a DES model that simulates the behaviour and constraints of a particular dynamic job shop setting. Due to the stochastic nature of the DJSSP models, multiple replications of each simulation model are required to obtain the steady-state performance of a given rule. Therefore, fitness evaluation is the most computationally demanding component in any evolutionary computation algorithm, and GP and GEP algorithms are not exceptions. In order to reduce the computational burden of simulation models that significantly reduce the training time of evolutionary algorithms, surrogate models have been developed. Surrogate models, also known as metamodels, response surfaces, or emulators, are simplified approximations of complex simulation models that are trained using input-output data at several selected locations in the design parameter space (Jin, 2011).

Using surrogate models that can imitate the underlying simulation model as accurately as possible with low computational costs is expected to offer several advantages as follows.

I. It helps in the early identification of promising dispatching rules, and thus directs the search to promising search regions.
II. Low-quality rules can be quickly discarded without sacrificing high computational costs in evaluating their fitness values.
III. The time saved can be used to explore more dispatching rules (intermediate population) resulting in achieving higher quality rules.

Therefore, the main objective of this chapter is to propose a surrogate assisted GEP approach to reduce the computational time required for the fitness evaluation of evolved rules without significantly affecting the prediction accuracy. Consequently, three surrogate models are developed by integrating the proposed approach with three surrogate models from the literature (Nguyen et al., 2017d) that showed a lower computational budget and high accuracy. Moreover, this chapter assesses the efficiency of the proposed surrogates using the following objectives.

I. The models have to be independent of the structure of evolved rules or the evolutionary algorithm used, which will extend their application with other algorithms and scheduling decisions (Hildebrandt and Branke, 2015b).
II. The proposed models should achieve a low computational budget while maintaining the same level of prediction accuracy as the literature surrogates (Nguyen et al., 2017d)
III. The computational time needed for training the surrogate models needs to be affordable to reduce the overhead costs. In addition, it is preferable that the developed surrogate model be somewhat explainable to increase confidence for later use in real-world applications.

The rest of this chapter is structured as follows. Section 6.2 describes the proposed surrogate assisted GEP approach. The numerical experiments are illustrated in Section 6.3. The obtained rules results are presented in Section 6.4. Finally, Section 6.5 presents the conclusions of this chapter.

6.2 Proposed Surrogate-Assisted GEP Approach

To evaluate the performance of a given dispatching rule, a DES model is usually used to estimate the absolute fitness value. The absolute fitness value means the fitness value of a rule using the full simulation length. The DES length is determined by the total number of jobs j_a that need to be processed. Accordingly, the simulation time $t(j_a)$ required to assess the performance of a population of n rules is equal to $n \times t(j_a)$. Therefore, the idea of the proposed surrogate model is to collect training data while evaluating some selected rules m (training rules). Afterward, the collected data is used to develop an Machine Learning (ML) model that can reduce the simulation length required in evaluating the remaining rules $(n - m)$. In other words, the remaining rules are partially evaluated using the DES model with fewer jobs j_f i.e., until the number of finished jobs is equal to j_f. Consequently, the surrogate model is used to predict the absolute fitness values of the remaining rules. In this chapter, a multiple linear regression is used as the ML method. Linear regression is selected for several reasons as follows.

a) It is computationally affordable machine learning techniques compared with other complex methods such as neural network, support vector machine, etc.
b) It provides useful information about design parameters and their weights.
c) The results obtained can be partially interpreted and analysed.

The amount of change in computational time (rt) required for fitness assessment is shown in Equation 6.1, where s indicates the surrogate model training time. It is clear that the time required to evaluate the fitness values of evolved rules can be reduced by reducing the number of training rules m, reducing the number of jobs used for early termination j_f, and reducing the training time of the surrogate model s (model complexity and machine learning technique used).

$$rt = actual\ eval.time - the\ proposed\ approach\ eval.time$$
$$rt = n \times t(j_a) - \left(m \times t(j_a) + (n - m) \times t(j_f) + s\right) \quad (6.1)$$

The proposed surrogate model is integrated with the GEP algorithm. The proposed surrogate assisted GEP algorithm consists of six main steps, as shown in Figure 6.1. The algorithm starts by initializing a population of dispatching rules using the GEP representation and a predefined set of functions and terminals at Step 1. In Step 2, the initialized rules are divided into two groups, training rules and the remaining rules. In

addition, design parameters or reference points R are selected within the design space. In other words, fitness values at different percentages of finished jobs are used as reference points. Then, the training rules are fully evaluated using the DES model and training data is collected at the reference points as shown in Step 3. The training data comprises the fitness values of the training rules at different values for the number of completed jobs and absolute fitness values. In Step 4, a surrogate model is developed using the collected training data and a supervised ML technique. The surrogate model is in the form of a mathematical function that estimates the absolute performance of a given rule using fitness-related values collected at the design parameters. Consequently, the remaining rules are partially evaluated using the DES model with fewer jobs j_f i.e., until the number of finished jobs is equal to j_f as shown in Step 5. Finally, in Step 6, the surrogate model predicts absolute fitness values of the remaining rules using the data collected from the shortened DES runs.

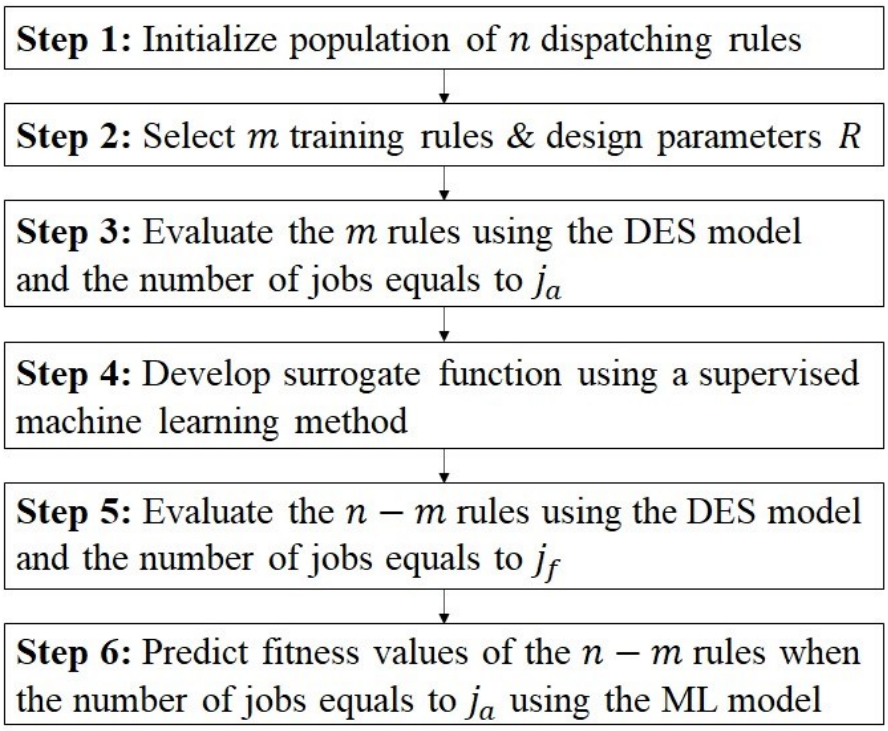


Figure 6.1: The proposed surrogate assisted GEP approach

Figure 6.2 shows a comparison of the proposed fitness assessment procedure with the fitness assessment method used in the literature, where j_0 denotes the first job that arrives at the job shop. Four equally spaced reference rules are used $R = \{r_1, r_2, r_3, r_4\}$ for the sake of clarification. In this example, if the number of completed jobs becomes equal to 20%, 40%, 60%, and 80% of the total number of jobs j_a, then fitness values of a given rule are calculated and used as the values of r_1, r_2, r_3, r_4 respectively.

Afterward, the collected data is used to train a surrogate model. The remaining rules are evaluated using the shortened DES model. Then, the surrogate model is used to predict the absolute fitness values of the remaining rules. Consequently, there are two new parameters in the proposed approach that must be investigated. They are the size of training rules (m) and the number of reference points (R)

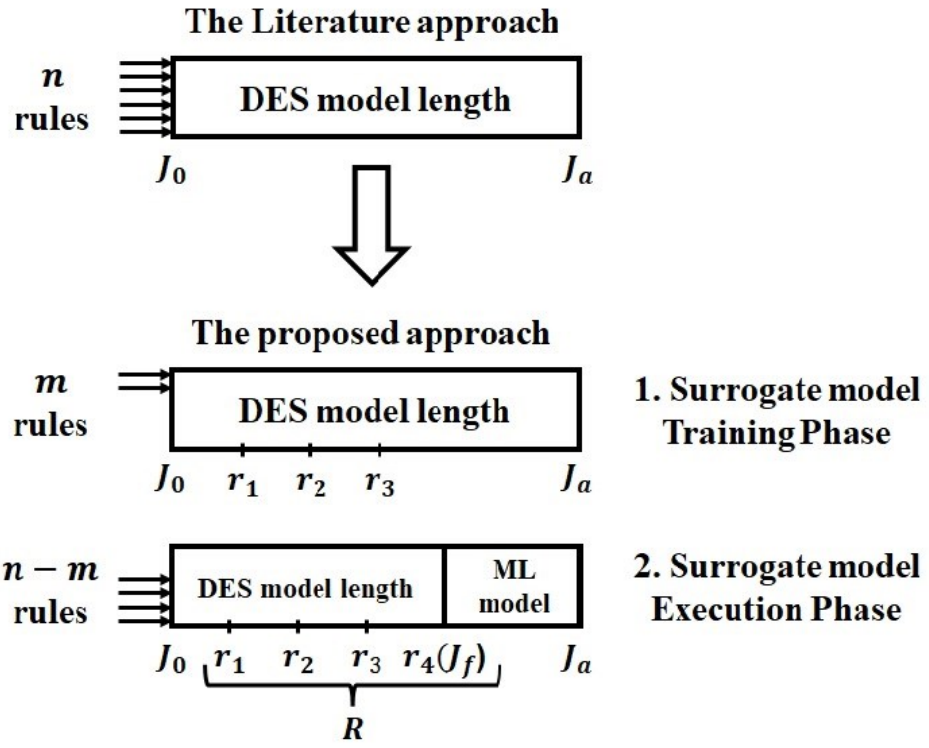


Figure 6.2: The fitness assessment method used in the literature approaches compared with the proposed approach

6.3 Numerical Experiments

All experiments are carried out on a DES model of a symmetrical job shop. The following are the common simulation settings used across all experiments:

- Jobs arrivals follow Poisson distribution.
- Job shop utilization level is 90%.
- Processing times follow uniform distribution U[1, 49].
- No machine break-down; pre-emption is not allowed.
- Job due dates are assigned using the total work content method with a tightness factor of 1.5.
- Two objective functions are considered: mean tardiness and mean flow time.

At each simulation replication, the job shop starts empty. All the collected data up to the warm-up period are discarded. Statistics from the warm-up period to the completion of the total number of required jobs are used to calculate performance measures. Six models are developed to assess the effectiveness of the proposed approach compared with literature methods as shown in Table 6.1. A reference model is used to evaluate the accuracy of the surrogate models. In the reference model, there are 10 machines, the simulation length is 2500 jobs, and the warm-up length is 500 jobs. Also, three literature models, namely OrigShort, HalfShop, and MiniShop are considered for the purpose of comparison (Nguyen et al., 2017d). The proposed approach is integrated with the three literature models referred to as ProShort, ProHalf, and ProMini. The reason behind this integration is that the proposed surrogate in contrast to literature surrogates reduces the simulation length without simplifying the job shop under study. Therefore, the proposed surrogate is applicable not only to the actual job shop configuration but also to its simplified versions. For OrigShort and ProShort models, the same job shop settings are used as the actual model except that the simulation model length and warm-up period are shorter. On the other hand, the other surrogate models have fewer machines and a maximum number of operations per job as well as shorter simulation length and warm-up period compared with the reference model.

Table 6.1 Job shop settings for the six surrogate models

Models	No. of machines	Max. no. of operations	Model length	Warm-up
OrigShort & ProShort	10	10	1000	200
HalfShop & ProHalf	5	5	500	100
MiniShop & ProMini	2	2	250	50

The three proposed models are compared with their counterpart literature models using two performance measures: computational time and accuracy. The prediction accuracy is estimated using the rank correlation coefficient ρ between the performance of rules using the reference model and the performance of rules using a given model (Nguyen et al., 2017d). The accuracy of a specific surrogate model is estimated as follows:

1. A population of dispatching rules $N = \{\Delta_1, \Delta_2, \ \dots, \Delta_n\}$.
2. Apply N to the reference model to obtain actual fitness $\gamma = \{f(\Delta_1), f(\Delta_2), \dots, f(\Delta_n)\}$ and the corresponding rank $\gamma_r = f_r(\Delta_1), f_r(\Delta_2), \dots, f_r(\Delta_n)\}$ sorted on descending order.
3. Apply the N rules to the surrogate model to obtain predicted fitness $\gamma' = \{f'(\Delta_1), f'(\Delta_2), \dots, f'(\Delta_n)\}$ and the corresponding rank $\gamma_r' = \{ f_r'(\Delta_1), f_r'(\Delta_2), \dots, f_r'(\Delta_n)\}$ sorted on descending order.
4. Estimate the correlation between the two models using the following function.

$$\rho = \frac{\sum_{i=1}^{n}(f_r(\Delta_i)-\bar{f_r})(f_r'(\Delta_i)-\bar{f_r}')}{\sqrt{\sum_{i=1}^{n}(f_r(\Delta_i)-\bar{f_r})^2 \sum_{i=1}^{n}(f_r'(\Delta_i)-\bar{f_r}')^2}} \qquad 6.2$$

The GEP algorithm is used to generate 300 rules represented as chromosomes. The number of genes at each chromosome is set to two; and the addition is used as a linking function. Moreover, the head length is set to 8. Table 6.2 shows the set of terminals and functions used. In addition, the underlined terminals are used when optimizing the mean tardiness objective, and are excluded in the case of the mean flow time objective.

Table 6.2 GEP terminal and function sets

Attribute	**Explanation**
JR	Job release date
OR	Operation ready time
WR	Work remaining of the job
PT	Operation processing time
RO	Number of remaining operations in a job
WT	Operation waiting time
NPT	Processing time of the next operation
WINQ	Work in the next queue
APR	Average processing time of queued job
DD	Job due date
CT	Machine ready time (current time)
SL	Job slack
Function set	$+, -, \times, /, max,$ and abs

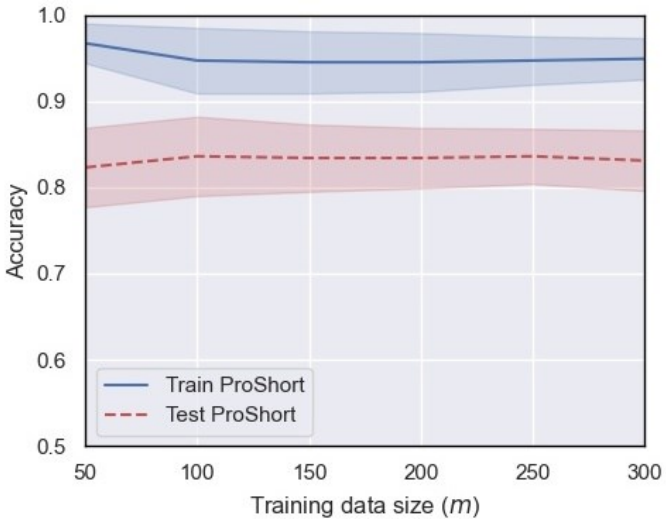
6.4 Results

First, the effects of the size of training rules and the number of reference points on the accuracy of the proposed model are examined. Second, the best values for the two parameters are used in the three proposed surrogates when compared with their counterpart in the literature.

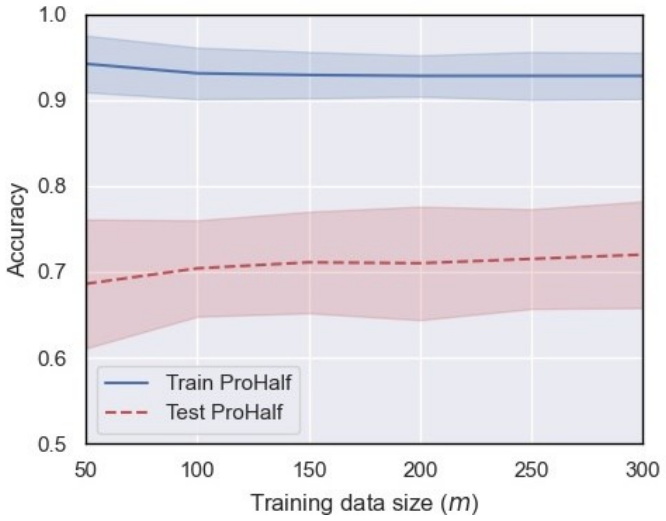
6.4.1 Fine-tuning the Surrogate Model Parameters

In order to ensure that a certain model does not overfit the training data, the performance of the model is tracked during the training and testing phases across different settings. The training performance indicates the accuracy of a surrogate model in predicting the rank of evolved rules using the same dynamic job shop problem instances. In contrast, testing performance indicates the accuracy of a surrogate model in predicting the rank of evolved rules using unseen instances. The parameters analysis results obtained from the ProShort, ProHalf, and ProMini models are presented for each objective function. Moreover, 20 simulation replications are used for each configuration. Figure 6.3 (a), (b), and (c) respectively illustrate the accuracy of the ProShort, ProHalf, and ProMini models using different sizes of training rules ranging from 50 to 300 rules with the number of reference points set to 10 for the mean tardiness objective. It is clear that the size of the training data does not greatly affect the prediction accuracy of the proposed surrogates. One of the reasons might be that the proposed approach uses training rules to understand the behavior of the job shop rather than the rules' phenotypic characterization.

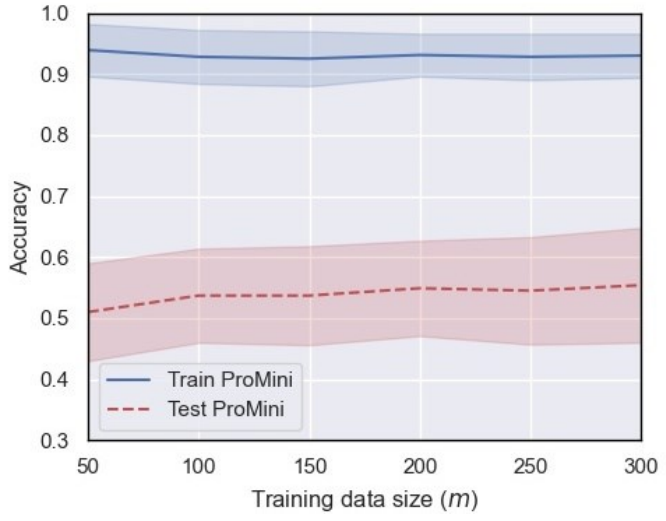
Although all models had a high prediction accuracy in the training instances, their performance deteriorated in the testing scenarios. The ProShort model had the smallest gap between the training and testing performance by about 10% on average, whereas the ProMini model had the largest performance difference by about 40% on average. As expected, the ProShort model had the highest prediction accuracy on the testing instances because it used the same job shop settings except for the simulation length and warm-up period. In contrast, the ProMini showed the lowest accuracy because it used a highly simplified version of the job shop plus short simulation and warmup lengths. The ProHalf model showed moderate performance compared with the other surrogates. Similar prediction accuracy results were obtained for the mean flow time objective as shown in Figure 6.4 (a), (b), and (c) for the ProShort, ProHalf, and ProMini models, respectively.



(a)

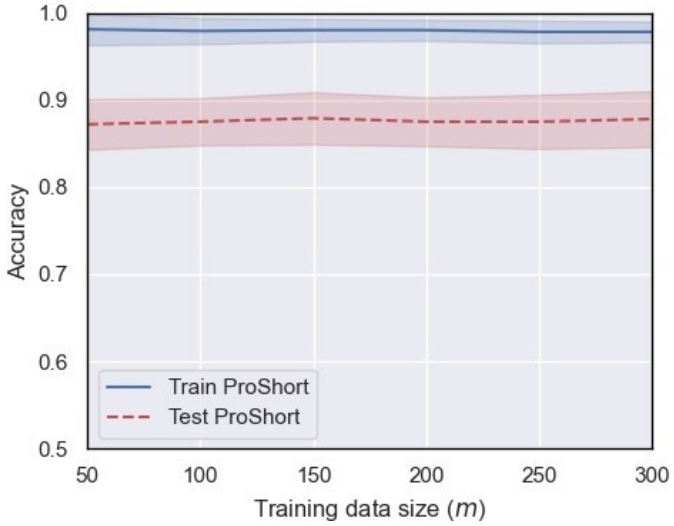


(b)

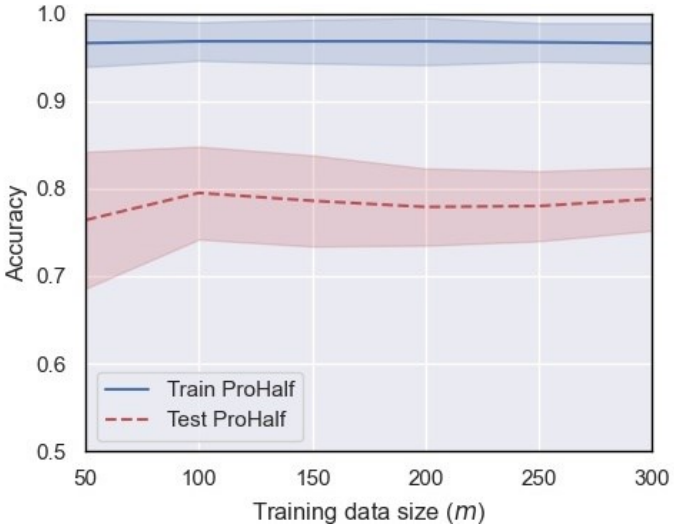


(c)

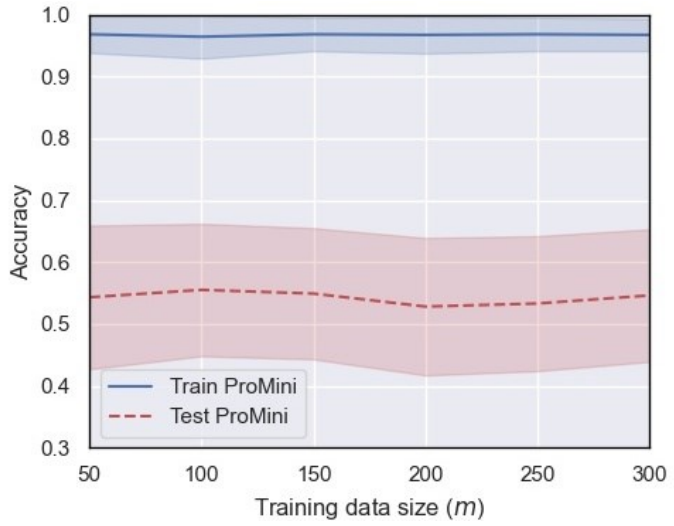
Figure 6.3: The effect of the number of training rules on the accuracy for the mean tardiness objective



(a)



(b)



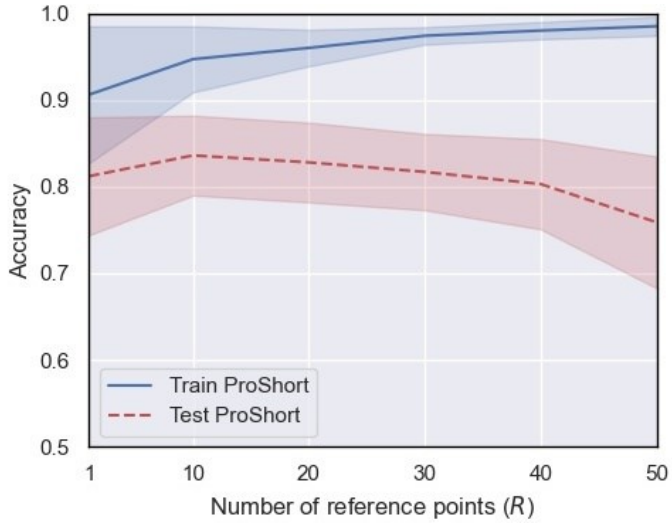
(c)

Figure 6.4: The effect of the number of training rules on the accuracy for the mean flow time objective

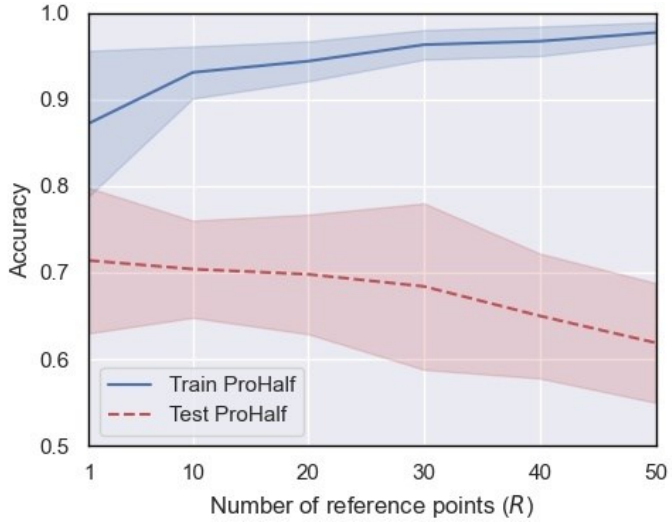
The main difference is that the testing performance of the ProShort and ProHalf models for the mean flow time objective is relatively higher than in the case of the mean tardiness objective. There are two reasons that might be the reason behind this difference as follows.

a) It is noted that the mean flow time objective is more robust than the mean tardiness, and hence its values are easier to predict (Pinedo, 2012).
b) More terminals were used in the MT objective compared with the MFT resulting in more complex rules and more difficulty in predicting their behavior.

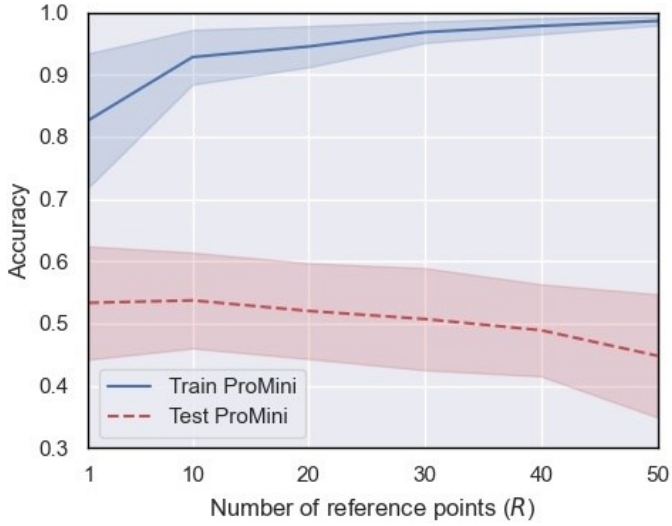
Several surrogate model runs were carried out to assess the prediction accuracy using different reference point values ranging from 1 to 50 points and 100 training rules for each objective function. It is worth noting that increasing the number of reference rules increases the complexity of the ML model generated. This in turn may increase training time, reduce the model understandability, and increase the risk of overfitting. For instance, if 20 reference points are taken into account when developing a linear regression model, then the model will have an intercept and 20 independent variables to predict the fitness value of a given rule. Figure 6.5 (a), (b), and (c) respectively show the accuracy of the ProShort, ProHalf, ProMini models across different values of reference points for the mean tardiness objective. As expected, an increase in the number of reference points increased training accuracy but negatively affected test accuracy. Because the generated models became more complex and learned the details and noise in the training instance, and thus had lower performance in the new instances. Regarding the ProShort model, the accuracy during the training and testing phases was stable at around 0.97, and 0.83 when the number of reference points was less than 30. The gap between training and testing accuracies widens when R exceeded 30 points. For example, when the number of points reached 50, the training accuracy was about 0.98 while the testing accuracy was 0.76. Similar behavior was observed in the ProHalf and ProMini models but in a more pronounced way. The training and testing accuracies for the ProHalf model were 0.98 and 0.62 when the number of reference points was set to 50. In addition, the training and testing accuracies for the ProMini model were 0.98 and 0.46 when the number of reference points was set to 50. Regarding the mean flow time objective, Figure 6.6 (a), (b), and (c) respectively show the accuracy of the ProShort, ProHalf, ProMini models for the same range of reference points. Similar to the mean tardiness objective, the training accuracy increased as the number of reference points increased, whereas testing performance decreased when the number exceeded 10 points.



(a)

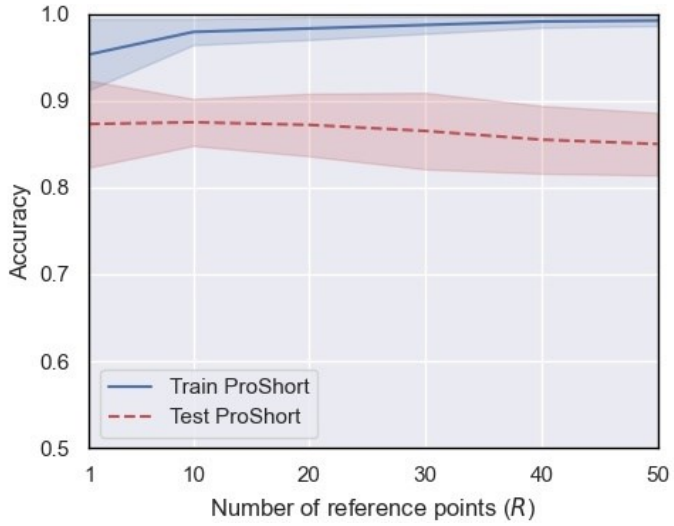


(b)

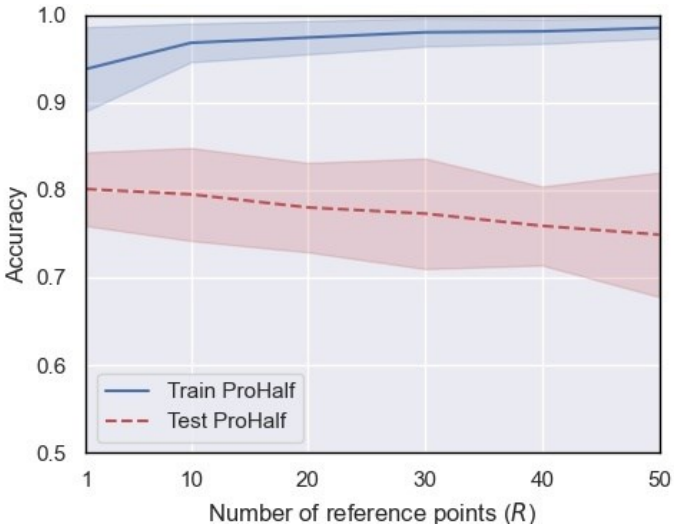


(c)

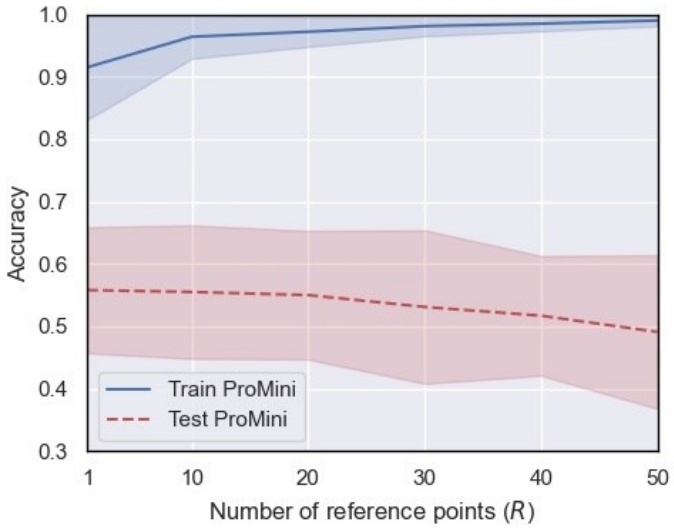
Figure 6.5: The effect of the number of reference points on the accuracy for the mean tardiness objective



(a)



(b)



(c)

Figure 6.6: The effect of the number of reference points on the accuracy for the mean flow time objective

6.4.2 Computational Time

In the following experiments, 100 training rules and 10 reference points (best settings) were used in the proposed surrogates. The three proposed models were compared with their counterparts in the literature. Figure 6.7 shows the results obtained with respect to computational time for the six surrogate models under the mean tardiness objective. The OrigShort and ProShort models had the highest computational costs compared with the other models, whereas MiniShop and ProMini models obtained the lowest computational times. In addition, the proposed surrogates required a smaller computational budget compared with the literature models. Table 6.3 shows the means and standard deviations of the achieved results. Moreover, a Wilcoxon-signed-sum test with a significant level of 0.05 was used to assess whether these differences were statistically significant. Table 6.3 shows the p-values obtained from comparing the proposed and the literature models as well as the percentage decrease in the computational time for the two compared models. It was clear that the proposed surrogate assisted GEP approach significantly reduced the computational time of the literature surrogates. Specifically, the proposed approach reduced the computational time of the OrigShort, HalfShop, and MiniShop models by about 5.56%, 5.24%, and 8.46% respectively.

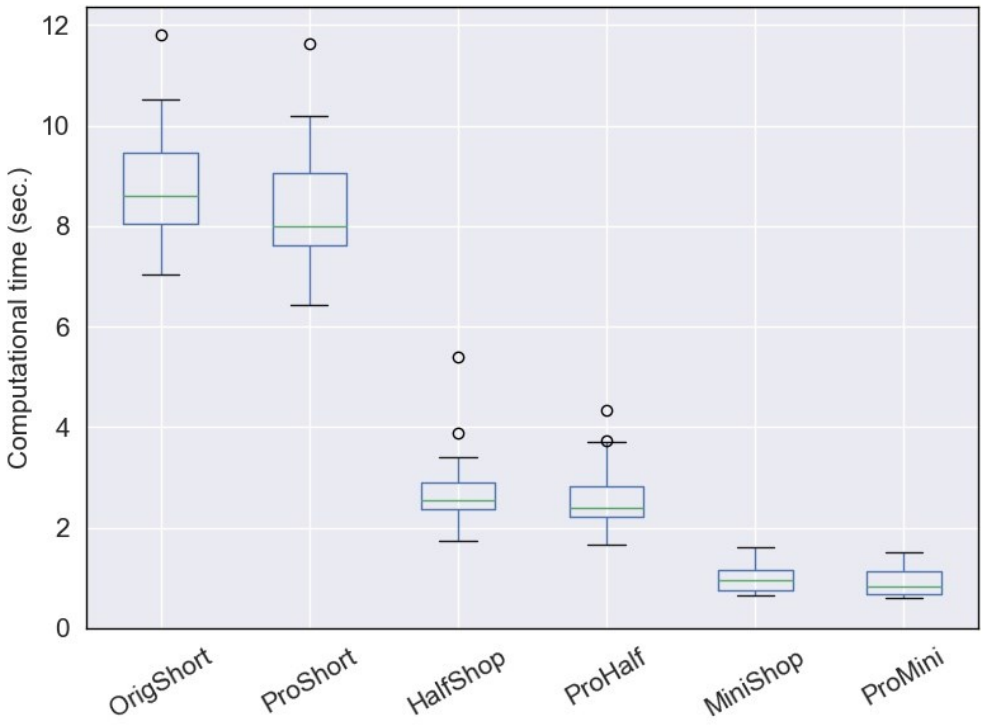


Figure 6.7: Computational times of the six surrogate models under the mean tardiness objective

Table 6.3 Obtained results for the MT objective regarding computational time

Surrogate Model	Computational Time for the MT objective		
	Obtained results	% Decrease	p-value
OrigShort	8.762 ± 1.188	5.563	0.008
ProShort	8.275 ± 1.242		
HalfShop	2.745 ± 0.788	5.236	0.012
ProHalf	2.602 ± 0.681		
MiniShop	0.976 ± 0.265	8.461	0.004
ProMini	0.894 ± 0.263		

Similar results were achieved with respect to the mean flow time objective as the proposed approach helped reduce the computational time significantly for the OrigShort, HalfShop, MiniSHop models by about 8.33, 6.39, and 6.05%, respectively. It is worth noting that the reduction in computational time was more pronounced in the MFT objective compared with the MT objective.

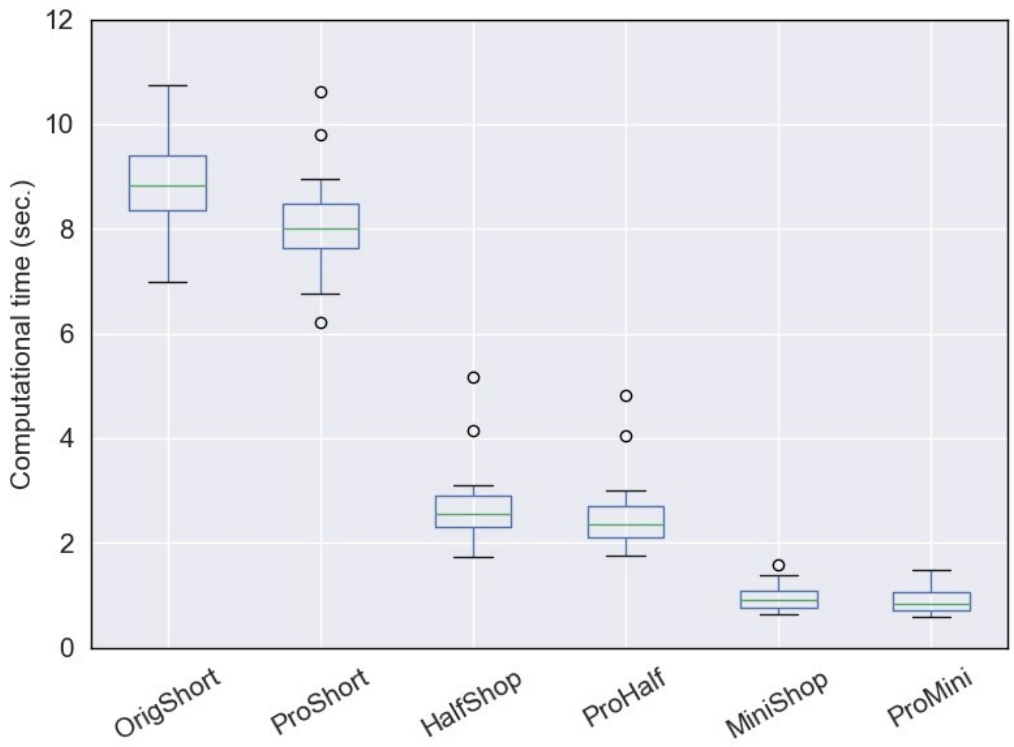


Figure 6.8: Computational times of the six surrogate models under the mean flow time objective

Table 6.4 Obtained results for the MFT objective regarding computational time

Surrogate Model	Computational Time for the MFT objective		
	Obtained results	% Decrease	p-value
OrigShort	8.807 ± 1.015	8.328	0.0
ProShort	8.073 ± 1.03		
HalfShop	2.712 ± 0.774	6.385	0.0
ProHalf	2.539 ± 0.735		
MiniSHop	0.953 ± 0.265	6.046	0.0
ProMini	0.895 ± 0.246		

6.4.3 Prediction Accuracy

The results obtained regarding the prediction accuracy of the six surrogate models for the mean tardiness objective are shown in Figure 6.9. Moreover, Table 6.5 illustrates the percentage decrease in accuracy, and the estimated p-value from comparing the proposed models with their counterparts in the literature. The was no significant difference in prediction accuracy between the proposed models and the literature models (p-values > 0.05). Specifically, the estimated p-values from comparing OrigShort vs. ProShort, HalfShop vs. ProHalf, and MiniShop vs. ProMini were 0.53, 0.21, and 0.94, respectively. Integrating the proposed surrogate assisted GEP approach with the OrigShort, HalfShop, and Minishop models reduced their accuracy by about 0.77 %, 1.84 %, and 0.35 % on average, respectively. Therefore, it is clear that the proposed approach helped to significantly reduce the computational costs of fitness assessment without causing a significant loss in prediction accuracy with respect to the mean tardiness objective. Regarding the mean flow time objective, Figure 6.10 represents the prediction accuracy of the three proposed surrogates compared with their counterparts from the literature. In addition, the statistical results are shown in Table 6.6. Similar results were obtained as the MT objective. Specifically, the estimated p-values from the comparison of OrigShort vs. ProShort, HalfShop vs. ProHalf, and MiniShop vs. ProMini were 0.53, 0.21, and 0.94, respectively. Moreover, integrating the proposed approach with the OrigShort, HalfShop, and MiniShop models reduced their accuracy by about 0.63 %, 1.25 %, and 1.63 % on average, respectively.

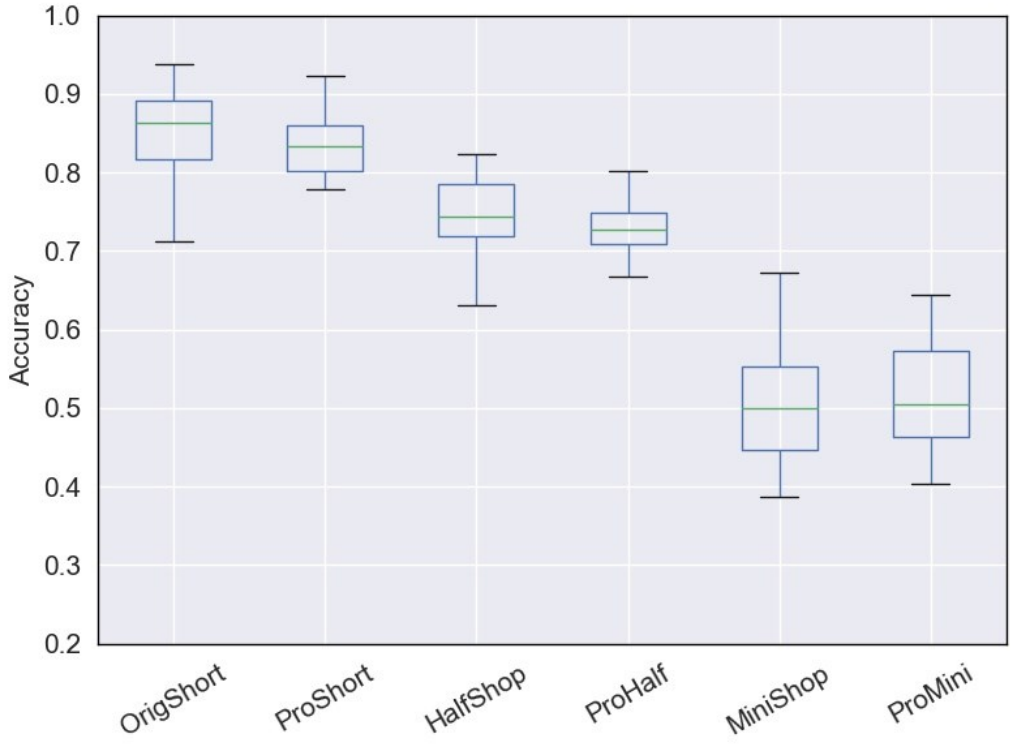


Figure 6.9: Rank correlation coefficients of the six surrogate models under the mean tardiness objective

Table 6.5 Obtained results regarding prediction accuracy for the mean tardiness objective

Surrogate Model	Prediction accuracy for the MT objective		
	Obtained results	% Decrease	p-value
OrigShort	0.84 ± 0.076	0.767	0.53
ProShort	0.834 ± 0.052		
HalfShop	0.743 ± 0.052	1.836	0.21
ProHalf	0.73 ± 0.042		
MiniSHop	0.518 ± 0.098	0.347	0.94
ProMini	0.517 ± 0.072		

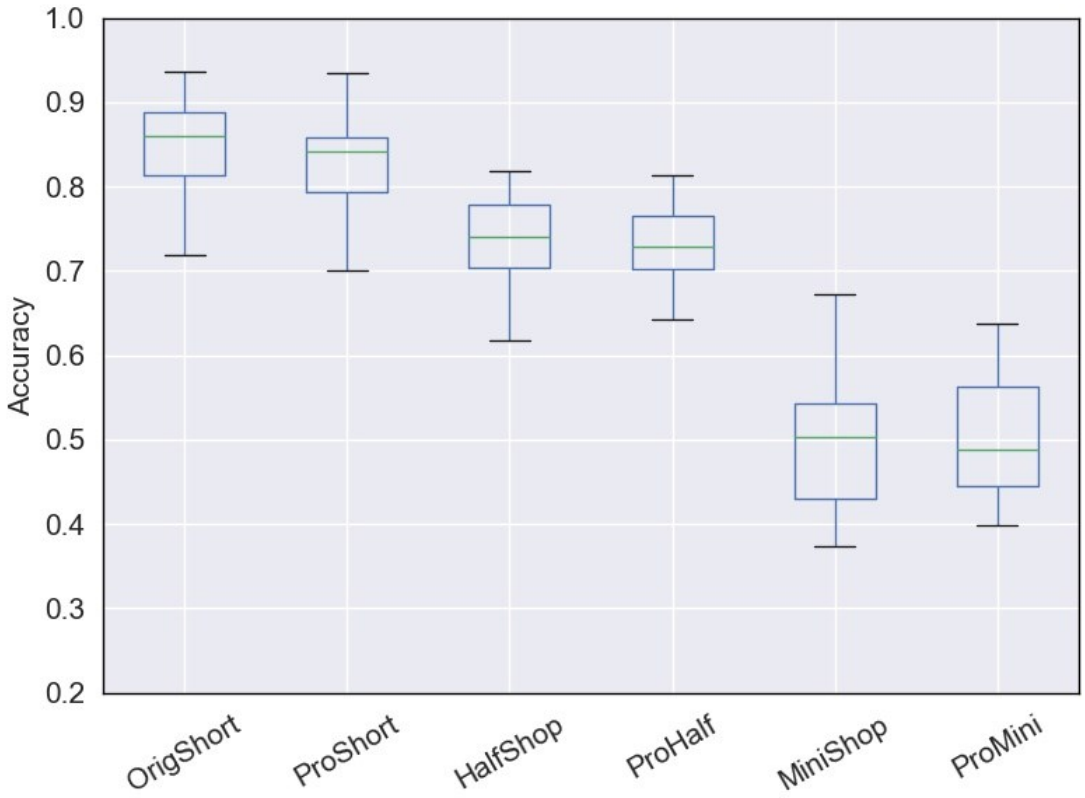


Figure 6.10: Rank correlation coefficients of the six surrogate models under the mean flow time objective

Table 6.6 Obtained results regarding prediction accuracy for the mean flow time objective

Surrogate Model	Computational Time for the MFT objective		
	Obtained results	% Decrease	p-value
OrigShort	0.838 ± 0.076	0.627	0.683
ProShort	0.832 ± 0.056		
HalfShop	0.738 ± 0.055	1.254	0.791
ProHalf	0.728 ± 0.047		
MiniSHop	0.511 ± 0.1	1.634	0.852
ProMini	0.503 ± 0.076		

6.5 Sample of the Proposed Surrogates

The main reason behind the significant reduction in computational time is that the proposed surrogate models are linear functions that are easy to train and evaluate. Technically, using these functions in fitness assessment is computationally cheaper than running a DES model which requires calculating the priority values of hundreds of operations using non-linear functions (GEP dispatching rules). In addition, it is worth noting that the collection of the training data used in the proposed surrogates does not require any additional operations because the fitness value is updated at the same time when there is a completed job. Equations 6.3, 6.4, and 6.5, respectively, represent the surrogate models generated in the ProShort, ProHalf, and ProMini models using a single replication for the mean tardiness objective. The p-values of most of the independent variables in the three models were less than 0.05 indicating their significant influence on the predicted fitness values. The R-squared values (Coefficient of determination) for the ProShort, ProHalf, ProMini were 0.984, 0.963, and 0.891, respectively. In addition, the adjusted R-squared values for the ProShort, ProHalf, ProMini were 0.982, 0.959, and 0.879, respectively. These results proved the ability of the proposed surrogates to capture most of the variations in the underlying simulation model.

$$y_{1000} = 1.58 - 0.6\,x_{200} - 0.95\,x_{277} - 0.74\,x_{354} + 0.4\,x_{431} + 0.44\,x_{508} + 0.63\,x_{585} + 0.08\,x_{662} + 0.58\,x_{739} - 1.06\,x_{816} + 1.15\,x_{893} \quad (6.3)$$

$$y_{500} = 63.83 - 0.32\,x_{100} + 0.56\,x_{138} + 0.2\,x_{176} - 0.28\,x_{214} + 0.06\,x_{252} - 0.48\,x_{290} - 0.06\,x_{328} - 0.4\,x_{366} - 2.85\,x_{404} + 4.57\,x_{442} \quad (6.4)$$

$$y_{250} = 28.04 + 0.22\,x_{50} - 0.63\,x_{69} + 0.44\,x_{88} + 0.16\,x_{107} + 0.15\,x_{126} - 3.93\,x_{145} + 2.69\,x_{164} + 0.21\,x_{183} - 0.12\,x_{202} + 1.7\,x_{221} \quad (6.5)$$

Moreover, Equations 6.6, 6.7, and 6.8, respectively, represent the surrogate models generated in the ProShort, ProHalf, and ProMini models using a single replication for the mean flow time objective. Similar regression results were obtained where the R-squared values for the ProShort, ProHalf, ProMini were 0.974, 0.964, and 0.906, respectively. In addition, the adjusted R-squared values for the ProShort, ProHalf, ProMini were 0.971, 0.96, and 0.895, respectively. Finally, most of the independent variables had p-values less than 0.05 representing their significant impact on the performance of a given model.

$$y_{1000} = 6.17 - 0.04\,x_{200} - 0.05\,x_{277} - 0.04\,x_{354} + 0.19\,x_{431} + 0.08\,x_{508} + 0.15\,x_{585} + 0.19\,x_{662} + 0.12\,x_{739} - 0.17\,x_{816} + 0.06\,x_{893} \quad 6.6$$

$$y_{500} = -1.86 - 0.06\,x_{100} - 0.02\,x_{138} + 0.07\,x_{176} + 0.19\,x_{214} + 0.07\,x_{252} + 0.24\,x_{290} + 0.11\,x_{328} + 0.08\,x_{366} + 0.21\,x_{404} + 0.08\,x_{442} \quad 6.7$$

$$y_{250} = 28.04 + 0.22\,x_{50} - 0.63\,x_{69} + 0.44\,x_{88} + 0.16\,x_{107} + 0.15\,x_{126} - 3.93\,x_{145} + 2.69\,x_{164} + 0.21\,x_{183} - 0.12\,x_{202} + 1.7\,x_{221} \quad 6.8$$

6.6 Chapter Summary

This chapter proposed a surrogate-assisted gene expression programming algorithm to reduce the computational time required for fitness evaluations. The idea of the approach is to reduce simulation length by early termination of the actual run and use a computationally inexpensive model to replace the excluded simulation length. To develop a surrogate model for a specific simulation model, fitness-related data must be collected during expensive simulation runs of a subset of training rules. Afterward, the collected data with their corresponding fitness values were used to train a machine learning model, which is a multiple linear regression in this work. Three surrogate models, ProShort, ProHalf, and ProMini, were developed by integrating the proposed approach with three simulation models from the literature, OrigShop, HalfShop, and MiniShop, respectively. The effect of the two new parameters used in the proposed approach on prediction accuracy was analysed. In addition, two objective functions were used including mean tardiness and mean flow time. Then, the proposed surrogate models using the best achieved settings were compared with their counterparts in the literature regarding computational time and prediction accuracy. A Wilcoxon-signed-sum test with a significant level of 0.05 was used to assess whether the difference between the proposed and literature models was statistically significant.

Experimental results showed that the proposed approach reduced the computational time of the OrigShort, HalfShop, and MiniShop models by about 5.56%, 5.24%, and 8.46%, respectively, for the mean tardiness objective. In addition, the ProShort, ProHalf, and ProMini models had smaller computational time compared with the OrigShort, HalfShop, and MiniShop by about 8.33%, 6.39%, and 6.05%, respectively, for the mean flow time objective. Regarding the prediction accuracy, there was no

significant difference between both the proposed models and their counterparts in the literature. Specifically, the percentage decrease in the prediction accuracy between OrigShort vs. ProShort, HalfShop vs. ProHalf, and MiniShop vs. ProMini was about 0.77%, 1.84%, and 0.35%, respectively, in the case of the mean tardiness objective. Regarding the mean flow time objective, the percentage decrease in the prediction accuracy between OrigShort vs. ProShort, HalfShop vs. ProHalf, and MiniShop vs. ProMini was about 0.63%, 1.25%, and 1.63%, respectively. Finally, several samples of the proposed surrogates for each of the objective functions were represented in a mathematical format with their regression results including R-squared and adjusted R-squared values.

Chapter 7. CONCLUSIONS

This thesis focused on the automatic generation of dispatching rules for static and dynamic job shop scheduling problems using the genetic programming algorithm. Specifically, the overall objective of this thesis was to automatically generate **high-quality** dispatching rules in **concise structures** (a smaller number of terminals and functions) and **low computational costs** for different job shop scheduling problems using the GP algorithm. In order to achieve this goal, several GP approaches were proposed in this thesis to reduce the size of evolved rules and reduce the computational burden during the GP training and evaluation phases. The effectiveness of each approach was evaluated across a wide range of job shop configurations using static and dynamic conditions, scheduling objectives, and performance measures.

The remainder of this chapter is organized as follows. Section 7.1 highlights the achieved objectives and provides the main conclusions for each chapter. Potential research directions for future work are presented in Section 7.2.

7.1 Achieved Objectives & Main Conclusions

This section describes the five research objectives that have been fulfilled in this thesis as follows.

I. This thesis proposed a distance metric to measure the similarity between the GP individuals with the aim of **promoting diversity,** which leads to the generation of dispatching rules with better fitness values. The proposed metric was used to numerically estimate the genotypic difference between the newly generated rules and the best rule evolved so far in a computationally affordable manner

compared with other literature metrics. Promoting diversity among GP individuals helped to avoid premature convergence that typically occurs when the GP algorithm is used for SJSSPs. Therefore, higher quality rules were generated by efficiently exploring different regions of the search space.

II. This thesis introduced a multi-objective framework by integrating NSGA-II with the GP algorithm to simultaneously optimize diversity value (using the proposed distance metric), rule length, and solution quality of GP individuals for SJSSPs. Considering the three objectives simultaneously with the same weight helped to deal with their conflicting nature. Finally, the proposed approach increased selection pressure toward rules with high diversity values, smaller sizes, and better fitness values resulting in better exploration of the search space and **a high reduction in GP computation costs**.

III. This thesis developed a feature selection approach for the tree-based GP algorithm with the aim to reduce the size of evolved rules using a smaller set of GP terminals for the DJSSPs. The proposed approach consists of two main components, an attribute vector to collect useful evolutionary information, and an adaptive discrimination scheme to estimate the probability of selecting each of the terminals during the GP run. Finally, reducing the size of evolved rules using the proposed approach **reduced the training time needed for the GP algorithm** because smaller rules are computationally less expensive to evaluate. In addition, smaller rules are easier to understand and implement in the industry than complex ones, and therefore have a higher probability of adoption in real-world applications.

IV. This thesis modified the feature selection approach proposed for the GP algorithm to be applicable to the GEP algorithm for the DJSSPs where a fixed-linear representation was used. In addition, a GP feature selection approach from the literature was modified and integrated with the GEP algorithm. The aim was to evaluate the effect of imposing an additional constraint on the size of evolved rules in the case of the contained GP representation that is less susceptible to the bloating effect. Finally, using both the fixed GEP representation in parallel with the feature selection capability enforced more pressure towards generating smaller rules and **reducing training time**.

V. This thesis proposed a surrogate assisted GEP approach to **reduce the fitness evaluation time of evolved rules** in dynamic job shop settings without sacrificing accuracy. Consequently, the time saved from the fitness evaluation

phase was used to evaluate an additional number of dispatching rules, resulting in achieving higher quality rules. The proposed approach replaced part of the simulation length of an expensive DES model with a simple mathematical function. This function was trained using fitness-related information collected during the evaluation of some training rules. Then, the remaining rules were evaluated using the simplified model with a shortened simulation length and the developed surrogate model. The proposed approach depends only on the behaviour of the underlying model, and therefore it can be used with evolutionary algorithms other than GP or GEP algorithms.

The following are the main conclusions of this thesis, drawn from Chapter 3 to Chapter 6.

- Technically, increasing diversity among the evolved rules helps avoid premature convergence, and thus increases the solution quality of generated rules. Therefore, Chapter 3 proposed a genotypic distance metric to measure the similarity of scheduling rules evolved using the GP algorithm. The proposed metric differs from the similarity metrics in the literature as follows:
    a) It takes into account the location of the node as well as the edges connecting it to its parents in estimating similarity values.
    b) It does not require any simulation runs to estimate the fitness values of the compared rules, as in the case of phenotypic metrics.
    c) It gives more weight to the nodes closest to the root node than to the farthest nodes.

  In addition, a multi-objective framework was introduced to take advantage of the proposed metric in increasing diversity among GP evolved rules as well as reducing the computational burden of the GP algorithm. The proposed framework integrated NSGA-II with the GP algorithm to optimize diversity value, rule length, and solution quality for SJSSPs. Optimizing the three objectives at the same time supports the automatic generation of high-quality rules in concise structures and shorter computational time. Two versions of the framework were compared with three algorithms from the literature using two objective functions, makespan, and mean tardiness across ten benchmark SJSSP instances. For each objective, four performance measures were taken into account: fitness value, genotypic diversity, phenotypic diversity, and the average length of the evolved rules. Experimental results demonstrated the effectiveness

of the proposed methods in generating a phenotypically diverse population of scheduling rules with smaller sizes and higher solution quality compared with the literature methods.

- For the sake of reducing the size of evolved rules in DJSSPs and thus speeding up the process of automatically generating dispatching rules using the GP algorithm, Chapter 4 proposed an online feature selection approach for the tree-based GP algorithm. The proposed approach offered several advantages over the approaches reported in the literature, as follows.
    a) It uses a new attribute vector representation to estimate the weight of each terminal without being affected by the occurrence of redundant terminals or complex rule structures.
    b) It is an online feature selection approach to select important features in the current generation using the estimated weights of terminals from the previous generation.
    c) It uses a probabilistic selection method rather than the inclusion or exclusion method to provide a broad preference scheme for each feature.

  The proposed algorithm was compared with three algorithms from the literature as well as 30 manually-made rules using three objective functions, TWT, MT, and MFT objectives across training and testing scenarios. For each objective, five performance measures were used including solution quality, computational time, the average size of rules, the average number of active terminals, and the average number of absent terminals. The best rules generated using the proposed algorithm had fewer terminals than the rules generated using the other methods for the three objective functions. Finally, the set of significant terminals obtained during the GP run using the proposed approach was similar to that generated using one of the best offline feature selection approaches in the literature.
- In order to verify whether the proposed feature selection approach can reduce the size of evolved rules and computational time in representations other than the tree structure, Chapter 5 developed a feature selection approach for the GEP algorithm for DJSSPs. The proposed algorithm speeded up the search process by restricting the GP search space using a fixed linear representation. In addition, it directed the search to the most promising regions using the feature selection capability. The proposed approach extended the current literature as follows.

a) It is the first attempt to propose a bloating control technique for constrained GP representation in the field of automated design of scheduling rules.
b) It significantly reduces the training time and size of generated rules compared with the current approaches, resulting in greater potential for use in complex manufacturing environments.

The proposed algorithm was compared with three algorithms from the literature using a large set of training and testing scenarios for three objective functions. Results demonstrated the ability of the proposed approach to significantly reduce the computational time of the GEP algorithm by evolving high-quality rules in simpler structures compared with the literature approaches. Finally, to ensure that the proposed approach is not significantly affected by the GP representation or objective function used, the set of terminals selected across generations using the proposed approach was compared with the one obtained using the algorithm proposed in the previous chapter. The results showed that the two algorithms have the same set of selected terminals except for two terminals in the TWT objective and one terminal in the MT objective.

- Instead of reducing the size of evolved rules (considered in the previous chapters) to reduce the time required for generating dispatching rules, Chapter 6 proposed a surrogate assisted GEP approach to achieve the same objective but by reducing the fitness evaluation times. Three surrogate models were developed by integrating the proposed approach with three literature models. The proposed approach extended the existing literature through the following contributions:
  a) It is the first attempt to use machine learning to abstract a discrete event simulation model of DJSSPs.
  b) It reduces fitness evaluation time without significantly affecting accuracy.
  c) It is independent of the structure of GP evolved rules, and thus it can be adopted with other GP approaches.

  The three literature surrogates were compared with the three proposed ones using several scenarios as well as MT and MFT objectives functions. Experimental results proved that the proposed surrogates have significantly lower computational costs with a neglectable loss in prediction accuracy under the two considered objectives. Finally, regression results (R-squared and adjusted R-squared) for the proposed surrogates supported the ability of the proposed models to imitate the behaviour of the underlying DES model.

7.2 Future Research Directions

The field of automatic generation of dispatching rules using the GP algorithm is relatively new, and there are many promising research directions to be considered. The following points define the future research activities, motivated by the work presented in each chapter. Future works are classified into two main groups, general and specific research directions. The former includes broad directions that can extend most of the methods proposed in this thesis, while the latter includes specific directions dedicated to a particular approach.

General research directions:

- Centralized scheduling is the main problem domain investigated in this thesis. Therefore, one possible extension of this work is to use the proposed approach in other domains such as distributed scheduling, decentralized scheduling, and cloud manufacturing scheduling.
- All scheduling problems considered in this thesis are one piece and identical machines job shop. Therefore, possible extensions are to evaluate the performance of the proposed approaches under other scheduling problems such as batch processing, parallel machines, FJSSPs, manufacturing cells, etc.
- Dispatching rules developed in this thesis are used to solve one scheduling decision, which is the job sequencing decision. Therefore, other types of dispatching rules used for other decisions can be considered in future work, including machine routing rules, due date assigning rules, etc.
- The proposed approaches in this thesis are developed mainly for the GP-based hyper-heuristic algorithm. Therefore, future research work can be guided toward integrating the proposed approaches with other global search-based hyper-heuristic algorithms such as particle swarm optimization, ant colony optimization, etc.

Specific research directions:

- A possible extension of the distance metric proposed in Chapter 3 is to consider cluster similarity instead of the individual similarity currently used. In other words, individuals are grouped into clusters with the aim of increasing the distance between the cluster with the best individuals and other clusters rather than increasing the distance between the best-evolved rule and other rules. This idea might support increasing the exploration and exploitation ability of the GP

algorithm because a limited number of relatively similar rules are kept within the cluster of the best rules (exploitation), while other clusters are compared with it (exploration).

- The proposed GP feature selection proposed in Chapter 4 uses the information collected from the previous generation in estimating the weight of terminals in the current generation. Therefore, the next step of this work might be to utilize attribute vectors from all the previous generations. Although, this may increase the computational budget of the training phase, checking the trade-off between improving performance measures (rule length & solution quality) and increasing computational time is helpful. Another possible research direction is to propose a local search mechanism to explore the search space of the attribute vectors of evolved rules. The proposed mechanism changes the state of some terminals, from active to inactive or vice versa, in a limited set of rules and evaluates the rules' performance after these changes. If the rules achieve better solutions, then these changes will be accepted, otherwise, other changes will be applied. Because the time for fitness evaluations will be significantly increased, it is recommended that a surrogate model should be used in this stage.
- Possible future work for the GEP feature selection approach proposed in Chapter 5 is to add some dynamics to the fixed-length representation used in the GEP algorithm. Although the proposed approach obtained high-quality results in the DJSSP instances studied, the fixed chromosome length might lead to low-quality results in more complex environments. In other words, fixed-size rules include a limited amount of related information that cannot be exceeded, and thus they might get poor performance in more challenging problem domains. Therefore, it may be useful to start with a very large chromosome size in the first generation, and then select only a limited number of high-quality rules with large sizes for the next generation. Consequently, the rules required to reach population size are generated randomly with shorter sizes until a predetermined size limit is reached i.e., the average size of individuals decreases with increasing generations.
- Regarding the surrogate models proposed in Chapter 6, future research activities have to be geared towards implementing these models in actual GEP runs. Although the proposed models do not significantly misestimate the performance of evolved rules, this marginal error might negatively affect solution quality if it occurs with the best rules. Therefore, it is necessary to assess the impact of

integrating the proposed models with the GEP algorithm instead of the original simulation model. Then, the obtained results must be compared with the standard GEP algorithm in which surrogates are not used. In addition, it might be useful to use the original simulation model as well as the surrogate in an interchangeable manner based on some predefined conditions.

REFERENCE LIST

Adibi, M.A., Zandieh, M., Amiri, M., 2010. Multi-objective scheduling of dynamic job shop using variable neighborhood search. Expert Syst. Appl. 37, 282–287. https://doi.org/10.1016/j.eswa.2009.05.001

Afzal, W., Torkar, R., 2011. On the application of genetic programming for software engineering predictive modeling: A systematic review. Expert Syst. Appl. 38, 11984–11997. https://doi.org/10.1016/j.eswa.2011.03.041

Akram, K., Kamal, K., Zeb, A., 2016. Fast simulated annealing hybridized with quenching for solving job shop scheduling problem. Appl. Soft Comput. 49, 510–523. https://doi.org/10.1016/j.asoc.2016.08.037

Alfaro-Cid, E., Merelo, J.J., de Vega, F.F., Esparcia-Alcázar, A.I., Sharman, K., 2010. Bloat Control Operators and Diversity in Genetic Programming: A Comparative Study. Evol. Comput. 18, 305–332. https://doi.org/10.1162/evco.2010.18.2.18206

Al-Hinai, N., ElMekkawy, T.Y., 2011. Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. Int. J. Prod. Econ. 132, 279–291. https://doi.org/10.1016/j.ijpe.2011.04.020

Amir Haeri, M., Ebadzadeh, M.M., Folino, G., 2017. Statistical genetic programming for symbolic regression. Appl. Soft Comput. 60, 447–469. https://doi.org/10.1016/j.asoc.2017.06.050

Aytug, H., Lawley, M.A., McKay, K., Mohan, S., Uzsoy, R., 2005. Executing production schedules in the face of uncertainties: A review and some future directions,

in: European Journal of Operational Research. North-Holland, pp. 86–110. https://doi.org/10.1016/j.ejor.2003.08.027

Baker, K.R., 1984. Sequencing Rules and Due-Date Assignments in a Job Shop. Manag. Sci. 30, 1093–1104. https://doi.org/10.1287/mnsc.30.9.1093

Bangsow, S., 2020. Tecnomatix plant simulation. Springer.

Beasley, J.E., 1990. OR-Library: distributing test problems by electronic mail. J. Oper. Res. Soc. 41, 1069–1072.

Bilkay, O., Anlagan, O., Kilic, S.E., 2004. Job shop scheduling using fuzzy logic. Int. J. Adv. Manuf. Technol. 23, 606–619. https://doi.org/10.1007/s00170-003-1771-2

Blackstone, J.H., PHILLIPS, D.T., HOGG, G.L., 1982. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. Int. J. Prod. Res. 20, 27–45. https://doi.org/10.1080/00207548208947745

Blickle, T., 2000. Tournament selection. Evol. Comput. 1, 181–186.

Blickle, T., Thiele, L., 1995. A Mathematical Analysis of Tournament Selection., in: ICGA. Citeseer, pp. 9–15.

Branke, J., Hildebrandt, T., Scholz-Reiter, B., 2015. Hyper-heuristic evolution of dispatching rules: A comparison of rule representations. Evol. Comput. 23, 249–277. https://doi.org/10.1162/EVCO_a_00131

Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M., 2016a. Automated Design of Production Scheduling Heuristics: A Review. IEEE Trans. Evol. Comput. 20, 110–124. https://doi.org/10.1109/TEVC.2015.2429314

Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M., 2016b. Automated Design of Production Scheduling Heuristics: A Review. IEEE Trans. Evol. Comput. 20, 110–124. https://doi.org/10.1109/TEVC.2015.2429314

Burke, E., Gustafson, S., Kendall, G., Krasnogor, N., 2002. Advanced Population Diversity Measures in Genetic Programming, in: Guervós, J.J.M., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., Fernández-Villacañas, J.-L. (Eds.), Parallel Problem Solving from Nature — PPSN VII, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 341–350. https://doi.org/10.1007/3-540-45712-7_33

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S., 2003. Hyper-Heuristics: An Emerging Direction in Modern Search Technology, in: Glover, F.,

Kochenberger, G.A. (Eds.), Handbook of Metaheuristics. Springer US, Boston, MA, pp. 457–474. https://doi.org/10.1007/0-306-48056-5_16

Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., McCollum, B., Ochoa, G., Parkes, A.J., Petrovic, S., 2011. The Cross-Domain Heuristic Search Challenge – An International Research Competition, in: Coello, C.A.C. (Ed.), Learning and Intelligent Optimization. Springer, Berlin, Heidelberg, pp. 631–634. https://doi.org/10.1007/978-3-642-25566-3_49

Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R., 2013. Hyper-heuristics: A survey of the state of the art. J. Oper. Res. Soc. 64, 1695–1724. https://doi.org/10.1057/jors.2013.71

Burke, E.K., Gustafson, S., Kendall, G., 2004. Diversity in genetic programming: an analysis of measures and correlation with fitness. IEEE Trans. Evol. Comput. 8, 47–62. https://doi.org/10.1109/TEVC.2003.819263

Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R., 2019. A Classification of Hyper-Heuristic Approaches: Revisited, in: Gendreau, M., Potvin, J.-Y. (Eds.), Handbook of Metaheuristics. Springer International Publishing, Cham, pp. 453–477. https://doi.org/10.1007/978-3-319-91086-4_14

Burks, A.R., Punch, W.F., 2015. An Efficient Structural Diversity Technique for Genetic Programming, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO '15. Association for Computing Machinery, New York, NY, USA, pp. 991–998. https://doi.org/10.1145/2739480.2754649

Çaliş, B., Bulkan, S., 2015. A research survey: review of AI solution strategies of job shop scheduling problem. J. Intell. Manuf. 26, 961–973. https://doi.org/10.1007/s10845-013-0837-8

Campbell, H.G., Dudek, R.A., Smith, M.L., 1970. A Heuristic Algorithm for the n Job, m Machine Sequencing Problem. Manag. Sci. 16, B-630. https://doi.org/10.1287/mnsc.16.10.B630

Chaudhry, I.A., Khan, A.A., 2016. A research survey: review of flexible job shop scheduling techniques. Int. Trans. Oper. Res. 23, 551–591. https://doi.org/10.1111/itor.12199

Chen, L., Zheng, H., Zheng, D., Li, D., 2015. An ant colony optimization-based hyper-heuristic with genetic programming approach for a hybrid flow shop scheduling

problem, in: 2015 IEEE Congress on Evolutionary Computation (CEC). Presented at the 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 814–821. https://doi.org/10.1109/CEC.2015.7256975

Cowling, P., Kendall, G., Han, L., 2002. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600). Presented at the Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), pp. 1185–1190 vol.2. https://doi.org/10.1109/CEC.2002.1004411

Cowling, P., Kendall, G., Soubeiga, E., 2000. A hyperheuristic approach to scheduling a sales summit, in: Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling. Lecture Notes in Computer Science. Springer Verlag, pp. 176–190. https://doi.org/10.1007/3-540-44629-x_11

Crane, E.F., McPhee, N.F., 2006. The Effects of Size and Depth Limits on Tree Based Genetic Programming, in: Yu, T., Riolo, R., Worzel, B. (Eds.), Genetic Programming Theory and Practice III, Genetic Programming. Springer US, Boston, MA, pp. 223–240. https://doi.org/10.1007/0-387-28111-8_15

de Jong, E.D., Watson, R.A., Pollack, J.B., 2001. Reducing bloat and promoting diversity using multi-objective methods, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, GECCO'01. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 11–18.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002a. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6, 182–197. https://doi.org/10.1109/4235.996017

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002b. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6, 182–197. https://doi.org/10.1109/4235.996017

Dimopoulos, C., Zalzala, A.M.S., 2001. Investigating the use of genetic programming for a classic one-machine scheduling problem. Adv. Eng. Softw. 32, 489–498. https://doi.org/10.1016/S0965-9978(00)00109-5

Dolgui, A., Ivanov, D., Sethi, S.P., Sokolov, B., 2019. Scheduling in production, supply chain and Industry 4.0 systems by optimal control: fundamentals, state-of-the-art

and applications. Int. J. Prod. Res. 57, 411–432. https://doi.org/10.1080/00207543.2018.1442948

Dominic, P.D.D., Kaliyamoorthy, S., Kumar, M.S., 2004. Efficient dispatching rules for dynamic job shop scheduling. Int. J. Adv. Manuf. Technol. 24, 70–75. https://doi.org/10.1007/s00170-002-1534-5

Drake, J.H., Kheiri, A., Özcan, E., Burke, E.K., 2020. Recent advances in selection hyper-heuristics. Eur. J. Oper. Res. 285, 405–428.

Du, H., Wang, Z., Zhan, W., Guo, J., 2018. Elitism and Distance Strategy for Selection of Evolutionary Algorithms. IEEE Access 6, 44531–44541. https://doi.org/10.1109/ACCESS.2018.2861760

Duenas, A., Petrovic, D., 2008. An approach to predictive-reactive scheduling of parallel machines subject to disruptions. Ann. Oper. Res. 159, 65–82. https://doi.org/10.1007/s10479-007-0280-3

Đurasević, M., Jakobović, D., Knežević, K., 2016. Adaptive scheduling on unrelated machines with genetic programming. Appl. Soft Comput. 48, 419–430. https://doi.org/10.1016/j.asoc.2016.07.025

Ekárt, A., Németh, S.Z., 2000. A Metric for Genetic Programs and Fitness Sharing, in: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (Eds.), Genetic Programming, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 259–270. https://doi.org/10.1007/978-3-540-46239-2_19

Ferreira, C., 2001. Gene Expression Programming: a New Adaptive Algorithm for Solving Problems. Complex Syst. 13, 87–129.

Flórez, E., Gómez, W., Bautista, L., 2013. An ant colony optimization algorithm for job shop scheduling problem. Appl. Mech. Mater. 321–324, 2116–2121.

Friedlander, A., Neshatian, K., Zhang, M., 2011. Meta-learning and feature ranking using genetic programming for classification: Variable terminal weighting, in: 2011 IEEE Congress of Evolutionary Computation, CEC 2011. pp. 941–948. https://doi.org/10.1109/CEC.2011.5949719

Gao, K.Z., He, Z.M., Huang, Y., Duan, P.Y., Suganthan, P.N., 2020. A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing. Swarm Evol. Comput. 57, 100719. https://doi.org/10.1016/j.swevo.2020.100719

Garey, M.R., Johnson, D.S., Sethi, R., 1976. The Complexity of Flowshop and Jobshop Scheduling. Math. Oper. Res. 1, 117–129. https://doi.org/10.1287/moor.1.2.117

Geiger, C.D., Uzsoy, R., Aytuğ, H., 2006a. Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. J. Sched. 9, 7–34. https://doi.org/10.1007/s10951-006-5591-8

Geiger, C.D., Uzsoy, R., Aytuğ, H., 2006b. Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. J. Sched. 9, 7–34. https://doi.org/10.1007/s10951-006-5591-8

Giffler, B., Thompson, G.L., 1960. Algorithms for Solving Production-Scheduling Problems. Oper. Res. 8, 487–503. https://doi.org/10.1287/opre.8.4.487

Gromicho, J.A.S., van Hoorn, J.J., Saldanha-da-Gama, F., Timmer, G.T., 2012. Solving the job-shop scheduling problem optimally by dynamic programming. Comput. Oper. Res. 39, 2968–2977. https://doi.org/10.1016/j.cor.2012.02.024

Gustafson, S., Vanneschi, L., 2008. Crossover-Based Tree Distance in Genetic Programming. IEEE Trans. Evol. Comput. 12, 506–524. https://doi.org/10.1109/TEVC.2008.915993

Hildebrandt, T., Branke, J., 2015a. On using surrogates with genetic programming. Evol. Comput. 23, 343–367. https://doi.org/10.1162/EVCO_a_00133

Hildebrandt, T., Branke, J., 2015b. On using surrogates with genetic programming. Evol. Comput. 23, 343–367. https://doi.org/10.1162/EVCO_a_00133

Hildebrandt, T., Heger, J., Scholz-Reiter, B., 2010a. Towards improved dispatching rules for complex shop floor scenarios - A genetic programming approach, in: Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10. ACM Press, New York, New York, USA, pp. 257–264. https://doi.org/10.1145/1830483.1830530

Hildebrandt, T., Heger, J., Scholz-Reiter, B., 2010b. Towards improved dispatching rules for complex shop floor scenarios - A genetic programming approach, in: Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10. ACM Press, New York, New York, USA, pp. 257–264. https://doi.org/10.1145/1830483.1830530

Holland, J.H., 1992. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press.

Holthaus, O., Rajendran, C., 2000. Efficient jobshop dispatching rules: Further developments. Prod. Plan. Control 11, 171–178. https://doi.org/10.1080/095372800232379

Horng, S.-C., Lin, S.-S., Yang, F.-Y., 2012. Evolutionary algorithm for stochastic job shop scheduling with random processing time. Expert Syst. Appl. 39, 3603–3610. https://doi.org/10.1016/j.eswa.2011.09.050

Hughes, M., 2021. Investigating the effects Diversity Mechanisms have on Evolutionary Algorithms in Dynamic Environments. cornell university.

Hunt, R., Johnston, M., Zhang, M., 2014a. 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, GECCO '14. Association for Computing Machinery, New York, NY, USA, pp. 927–934. https://doi.org/10.1145/2576768.2598224

Hunt, R., Johnston, M., Zhang, M., 2014b. Evolving machine-specific dispatching rules for a two-machine job shop using genetic programming, in: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014. Institute of Electrical and Electronics Engineers Inc., pp. 618–625. https://doi.org/10.1109/CEC.2014.6900655

Hunt, R., Richard, J., Zhang, M., 2016a. Evolving Dispatching Rules with Greater Understandability for Dynamic Job Shop Scheduling Mark Johnston.

Hunt, R., Richard, J., Zhang, M., 2016b. Evolving Dispatching Rules with Greater Understandability for Dynamic Job Shop Scheduling Mark Johnston.

Hunt, R., Richard, J., Zhang, M., 2016c. Evolving Dispatching Rules with Greater Understandability for Dynamic Job Shop Scheduling Mark Johnston.

Hussain, K., Mohd Salleh, M.N., Cheng, S., Shi, Y., 2019. Metaheuristic research: a comprehensive survey. Artif. Intell. Rev. 52, 2191–2233. https://doi.org/10.1007/s10462-017-9605-z

Jabeen, H., Baig, A.R., 2010. Review of classification using genetic programming. Int. J. Eng. Sci. Technol. 2, 94–103.

Jackson, D., 2010. Phenotypic Diversity in Initial Genetic Programming Populations, in: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (Eds.), Genetic Programming, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 98–109. https://doi.org/10.1007/978-3-642-12148-7_9

Jackson, J.R., 1956. An extension of Johnson's results on job IDT scheduling. Nav. Res. Logist. Q. 3, 201–203.

Jakobović, D., Budin, L., 2006. Dynamic scheduling with genetic programming, in: European Conference on Genetic Programming. Springer, Berlin, Heidelberg, pp. 73–84. https://doi.org/10.1007/11729976_7

Jiang, Z., Yuan, S., Ma, J., Wang, Q., 2021. The evolution of production scheduling from Industry 3.0 through Industry 4.0. Int. J. Prod. Res. 0, 1–21. https://doi.org/10.1080/00207543.2021.1925772

Jin, Y., 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm Evol. Comput. 1, 61–70. https://doi.org/10.1016/j.swevo.2011.05.001

Johnson, S.M., 1954. Optimal two- and three-stage production schedules with setup times included. Nav. Res. Logist. Q. 1, 61–68. https://doi.org/10.1002/nav.3800010110

Jones, A., Rabelo, L.C., Sharawi, A.T., 1998. Survey of job shop scheduling techniques. NISTIR Natl. Inst. Stand. Technol. Gaithersburg MD.

Kaskavelis, C.A., Caramanis, M.C., 1998. Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems. IIE Trans. 30, 1085–1097. https://doi.org/10.1023/A:1007515931780

Kelly, J., Hemberg, E., O'Reilly, U.-M., 2019. Improving Genetic Programming with Novel Exploration - Exploitation Control, in: Sekanina, L., Hu, T., Lourenço, N., Richter, H., García-Sánchez, P. (Eds.), Genetic Programming, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 64–80. https://doi.org/10.1007/978-3-030-16670-0_5

Kennedy, J., 2006. Swarm Intelligence, in: Zomaya, A.Y. (Ed.), Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies. Springer US, Boston, MA, pp. 187–219. https://doi.org/10.1007/0-387-27705-6_6

Kiran, D.R., 2019. Chapter 1 - Elements of production planning and control, in: Kiran, D.R. (Ed.), Production Planning and Control. Butterworth-Heinemann, pp. 1–20. https://doi.org/10.1016/B978-0-12-818364-9.00001-9

Korytkowski, P., Rymaszewski, S., Wiśniewski, T., 2013. Ant colony optimization for job shop scheduling using multi-attribute dispatching rules. Int. J. Adv. Manuf. Technol. 67, 231–241. https://doi.org/10.1007/s00170-013-4769-4

Kouider, A., Bouzouia, B., 2012. Multi-agent job shop scheduling system based on co-operative approach of idle time minimisation. Int. J. Prod. Res. 50, 409–424. https://doi.org/10.1080/00207543.2010.539276

Koza, J. R., 1994a. Genetic programming II: Automatic discovery of reusable subprograms, Cambridge, MA, USA. ed, cs.bham.ac.uk. Cambridge, USA.

Koza, John R., 1994. Genetic programming as a means for programming computers by natural selection. Stat. Comput. 4, 87–112. https://doi.org/10.1007/BF00175355

Koza, J. R., 1994b. Genetic programming II: Automatic discovery of reusable subprograms, Cambridge, MA, USA. ed, cs.bham.ac.uk. Cambridge, USA.

Kumar, R., 2012. Blending roulette wheel selection & rank selection in genetic algorithms. Int. J. Mach. Learn. Comput. 2, 365–370.

Lawler, E.L., Wood, D.E., 1966. Branch-and-Bound Methods: A Survey. Oper. Res. 14, 699–719. https://doi.org/10.1287/opre.14.4.699

Liu, G., Song, S., Wu, C., 2012. Two Techniques to Improve the NEH Algorithm for Flow-Shop Scheduling Problems, in: Huang, D.-S., Gan, Y., Gupta, P., Gromiha, M.M. (Eds.), Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 41–48. https://doi.org/10.1007/978-3-642-25944-9_6

Liu, Y., Wang, L., Wang, X.V., Xu, X., Zhang, L., 2019. Scheduling in cloud manufacturing: state-of-the-art and research challenges. Int. J. Prod. Res. 57, 4854–4879. https://doi.org/10.1080/00207543.2018.1449978

Lohn, J.D., Hornby, G.S., Linden, D.S., 2005. An Evolved Antenna for Deployment on Nasa's Space Technology 5 Mission, in: O'Reilly, U.-M., Yu, T., Riolo, R., Worzel, B. (Eds.), Genetic Programming Theory and Practice II, Genetic Programming. Springer US, Boston, MA, pp. 301–315. https://doi.org/10.1007/0-387-23254-0_18

Luke, S., Panait, L., 2006. A Comparison of Bloat Control Methods for Genetic Programming. Evol. Comput. 14, 309–344. https://doi.org/10.1162/evco.2006.14.3.309

Maravelias, C.T., Sung, C., 2009. Integration of production planning and scheduling: Overview, challenges and opportunities. Comput. Chem. Eng., FOCAPO 2008 – Selected Papers from the Fifth International Conference on Foundations of Computer-Aided Process Operations 33, 1919–1930. https://doi.org/10.1016/j.compchemeng.2009.06.007

Masood, A., Mei, Y., Chen, G., Zhang, M., 2016. Many-objective genetic programming for job-shop scheduling, in: 2016 IEEE Congress on Evolutionary Computation, CEC 2016. Institute of Electrical and Electronics Engineers Inc., pp. 209–216. https://doi.org/10.1109/CEC.2016.7743797

McKay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M., 2010. Grammar-based Genetic Programming: a survey. Genet. Program. Evolvable Mach. 11, 365–396. https://doi.org/10.1007/s10710-010-9109-y

Mehta, S.V., Uzsoy, R.M., 1998. Predictable scheduling of a job shop subject to breakdowns. IEEE Trans. Robot. Autom. 14, 365–378. https://doi.org/10.1109/70.678447

Mei, Y., Nguyen, S., Xue, B., Zhang, M., 2017a. An Efficient Feature Selection Algorithm for Evolving Job Shop Scheduling Rules With Genetic Programming. IEEE Trans. Emerg. Top. Comput. Intell. 1, 339–353. https://doi.org/10.1109/tetci.2017.2743758

Mei, Y., Nguyen, S., Xue, B., Zhang, M., 2017b. An Efficient Feature Selection Algorithm for Evolving Job Shop Scheduling Rules With Genetic Programming. IEEE Trans. Emerg. Top. Comput. Intell. 1, 339–353. https://doi.org/10.1109/tetci.2017.2743758

Mei, Y., Nguyen, S., Xue, B., Zhang, M., 2017c. An Efficient Feature Selection Algorithm for Evolving Job Shop Scheduling Rules With Genetic Programming. IEEE Trans. Emerg. Top. Comput. Intell. 1, 339–353. https://doi.org/10.1109/tetci.2017.2743758

Mei, Y., Zhang, M., Nyugen, S., 2016. Feature selection in evolving job shop dispatching rules with Genetic Programming, in: GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference. Association for Computing

Machinery, Inc, New York, NY, USA, pp. 365–372. https://doi.org/10.1145/2908812.2908822

Miller, J.F., Harding, S.L., 2008. Cartesian genetic programming, in: Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '08. Association for Computing Machinery, New York, NY, USA, pp. 2701–2726. https://doi.org/10.1145/1388969.1389075

Minguillon, F.E., Lanza, G., 2019. Coupling of centralized and decentralized scheduling for robust production in agile production systems. Procedia CIRP, 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy 79, 385–390. https://doi.org/10.1016/j.procir.2019.02.099

Miyashita, K., 2000. Job-shop scheduling with genetic programming, in: Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation. pp. 505–512.

Mizrak, P., Bayhan, G.M., 2006. Comparative Study of Dispatching Rules in a Real-Life Job Shop Environment. Appl. Artif. Intell. 20, 585–607. https://doi.org/10.1080/08839510600779738

Mohan, J., Lanka, K., Rao, A.N., 2019. A review of dynamic job shop scheduling techniques, in: Procedia Manufacturing. Elsevier B.V., pp. 34–39. https://doi.org/10.1016/j.promfg.2019.02.006

Mori, N., McKay, B., Hoai, N.X., Essam, D., Takeuchi, S., 2008. A New Method for Simplifying Algebraic Expressions in Genetic Programming called Equivalent Decision Simplification. Scis Isis 2008, 1671–1676. https://doi.org/10.14864/softscis.2008.0.1671.0

Nawaz, M., Enscore, E.E., Ham, I., 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11, 91–95. https://doi.org/10.1016/0305-0483(83)90088-9

Nguyen, S., Mei, Y., Xue, B., Zhang, M., 2018a. A hybrid genetic programming algorithm for automated design of dispatching rules. Evol. Comput. 27, 467–596. https://doi.org/10.1162/evco_a_00230

Nguyen, S., Mei, Y., Xue, B., Zhang, M., 2018b. A hybrid genetic programming algorithm for automated design of dispatching rules. Evol. Comput. 27, 467–596. https://doi.org/10.1162/evco_a_00230

Nguyen, S., Mei, Y., Xue, B., Zhang, M., 2018c. A hybrid genetic programming algorithm for automated design of dispatching rules. Evol. Comput. 27, 467–596. https://doi.org/10.1162/evco_a_00230

Nguyen, S., Mei, Y., Zhang, M., 2017a. Genetic programming for production scheduling: a survey with a unified framework. Complex Intell. Syst. 3, 41–66. https://doi.org/10.1007/s40747-017-0036-x

Nguyen, S., Mei, Y., Zhang, M., 2017b. Genetic programming for production scheduling: a survey with a unified framework. Complex Intell. Syst. 3, 41–66. https://doi.org/10.1007/s40747-017-0036-x

Nguyen, S., Mei, Y., Zhang, M., 2017c. Genetic programming for production scheduling: a survey with a unified framework. Complex Intell. Syst. 3, 41–66. https://doi.org/10.1007/s40747-017-0036-x

Nguyen, S., Zhang, M., Johnston, M., Tan, K.C., 2014a. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. IEEE Trans. Evol. Comput. 18, 193–208. https://doi.org/10.1109/TEVC.2013.2248159

Nguyen, S., Zhang, M., Johnston, M., Tan, K.C., 2014b. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. IEEE Trans. Evol. Comput. 18, 193–208. https://doi.org/10.1109/TEVC.2013.2248159

Nguyen, S., Zhang, M., Johnston, M., Tan, K.C., 2013a. Dynamic multi-objective job shop scheduling: A genetic programming approach. Stud. Comput. Intell. 505, 251–282. https://doi.org/10.1007/978-3-642-39304-4_10

Nguyen, S., Zhang, M., Johnston, M., Tan, K.C., 2013b. Learning iterative dispatching rules for job shop scheduling with genetic programming. Int. J. Adv. Manuf. Technol. 67, 85–100. https://doi.org/10.1007/s00170-013-4756-9

Nguyen, S., Zhang, M., Johnston, M., Tan, K.C., 2013c. Dynamic multi-objective job shop scheduling: A genetic programming approach. Stud. Comput. Intell. 505, 251–282. https://doi.org/10.1007/978-3-642-39304-4_10

Nguyen, S., Zhang, M., Tan, K.C., 2017d. Surrogate-Assisted Genetic Programming with Simplified Models for Automated Design of Dispatching Rules. IEEE Trans. Cybern. 47, 2951–2965. https://doi.org/10.1109/TCYB.2016.2562674

Nguyen, S., Zhang, M., Tan, K.C., 2015a. Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems, in: 2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings. Institute of Electrical and Electronics Engineers Inc., pp. 2781–2788. https://doi.org/10.1109/CEC.2015.7257234

Nguyen, S., Zhang, M., Tan, K.C., 2015b. Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems, in: 2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings. Institute of Electrical and Electronics Engineers Inc., pp. 2781–2788. https://doi.org/10.1109/CEC.2015.7257234

Nie, L., Gao, L., Li, P., Li, X., 2013a. A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. J. Intell. Manuf. 24, 763–774. https://doi.org/10.1007/s10845-012-0626-9

Nie, L., Gao, L., Li, P., Li, X., 2013b. A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. J. Intell. Manuf. 24, 763–774. https://doi.org/10.1007/s10845-012-0626-9

Nie, L., Gao, L., Li, P., Zhang, L., 2011. Application of gene expression programming on dynamic job shop scheduling problem, in: Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2011. pp. 291–295. https://doi.org/10.1109/CSCWD.2011.5960088

Nie, L., Shao, X., Gao, L., Li, W., 2010. Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems. Int. J. Adv. Manuf. Technol. 50, 729–747. https://doi.org/10.1007/s00170-010-2518-5

Nordgren, W.B., 2002. Flexsim simulation environment, in: Proceedings of the Winter Simulation Conference. IEEE, pp. 250–252.

Onar, S.Ç., Öztayşi, B., Kahraman, C., Yanık, S., Şenvar, Ö., 2016. A Literature Survey on Metaheuristics in Production Systems, in: Talbi, E.-G., Yalaoui, F., Amodeo, L. (Eds.), Metaheuristics for Production Systems, Operations Research/Computer Science Interfaces Series. Springer International Publishing, Cham, pp. 1–24. https://doi.org/10.1007/978-3-319-23350-5_1

O'Neill, M., 2009. Riccardo Poli, William B. Langdon, Nicholas F. McPhee: A Field Guide to Genetic Programming. Genet. Program. Evolvable Mach. 10, 229–230. https://doi.org/10.1007/s10710-008-9073-y

Ouelhadj, D., Petrovic, S., 2009. A survey of dynamic scheduling in manufacturing systems. J. Sched. 12, 417–431. https://doi.org/10.1007/s10951-008-0090-8

Ozturk, G., Bahadir, O., Teymourifar, A., 2020. Extracting New Dispatching Rules for Multi-objective Dynamic Flexible Job Shop Scheduling with Limited Buffer Spaces. Cogn. Comput. 12, 195–205. https://doi.org/10.1007/s12559-018-9595-4

Ozturk, G., Bahadir, O., Teymourifar, A., 2019. Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming. Int. J. Prod. Res. 57, 3121–3137. https://doi.org/10.1080/00207543.2018.1543964

Palmer, D.S., 1965. Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time—A Quick Method of Obtaining a Near Optimum. J. Oper. Res. Soc. 16, 101–107. https://doi.org/10.1057/jors.1965.8

Pandey, H.M., Chaudhary, A., Mehrotra, D., 2014. A comparative review of approaches to prevent premature convergence in GA. Appl. Soft Comput. 24, 1047–1077. https://doi.org/10.1016/j.asoc.2014.08.025

Park, B.J., Choi, H.R., Kim, H.S., 2003. A hybrid genetic algorithm for the job shop scheduling problems. Comput. Ind. Eng. 45, 597–613. https://doi.org/10.1016/S0360-8352(03)00077-9

Pinedo, M.L., 2012. Scheduling: Theory, Algorithms, and Systems, 4th ed. Springer-Verlag, New York. https://doi.org/10.1007/978-1-4614-2361-4

Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R., 2007. Genetic programming: An introductory tutorial and a survey of techniques and applications. Univ. Essex UK Tech Rep CES-475 927–1028.

Rajendran, C., Holthaus, O., 1999. Comparative study of dispatching rules in dynamic flowshops and jobshops. Eur. J. Oper. Res. 116, 156–170. https://doi.org/10.1016/S0377-2217(98)00023-X

Ramasesh, R., 1990. Dynamic job shop scheduling: A survey of simulation research. Omega 18, 43–57. https://doi.org/10.1016/0305-0483(90)90017-4

Riley, M., Mei, Y., Zhang, M., 2016. Improving job shop dispatching rules via terminal weighting and adaptive mutation in genetic programming, in: 2016 IEEE Congress on Evolutionary Computation (CEC). Presented at the 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 3362–3369. https://doi.org/10.1109/CEC.2016.7744215

Sabar, N.R., Ayob, M., Kendall, G., Qu, R., 2015. Automatic Design of a Hyper-Heuristic Framework With Gene Expression Programming for Combinatorial Optimization Problems. IEEE Trans. Evol. Comput. 19, 309–325. https://doi.org/10.1109/TEVC.2014.2319051

Sareni, B., Krahenbuhl, L., 1998. Fitness sharing and niching methods revisited. IEEE Trans. Evol. Comput. 2, 97–106. https://doi.org/10.1109/4235.735432

Schmidt, M., Lipson, H., 2011. Age-Fitness Pareto Optimization, in: Riolo, R., McConaghy, T., Vladislavleva, E. (Eds.), Genetic Programming Theory and Practice VIII, Genetic and Evolutionary Computation. Springer, New York, NY, pp. 129–146. https://doi.org/10.1007/978-1-4419-7747-2_8

Sels, V., Gheysen, N., Vanhoucke, M., 2012a. A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions. Int. J. Prod. Res. 50, 4255–4270. https://doi.org/10.1080/00207543.2011.611539

Sels, V., Gheysen, N., Vanhoucke, M., 2012b. A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions. Int. J. Prod. Res. 50, 4255–4270. https://doi.org/10.1080/00207543.2011.611539

Shady, S., Kaihara, T., Fujii, N., Kokuryo, D., 2021a. A New Representation and Adaptive Feature Selection for Evolving Compact Dispatching Rules for Dynamic Job Shop Scheduling with Genetic Programming, in: Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems, IFIP Advances in Information and Communication Technology. Springer International Publishing, pp. 646–654. https://doi.org/10.1007/978-3-030-85906-0_70

Shady, S., Kaihara, T., Fujii, N., Kokuryo, D., 2021b. Evolving Dispatching Rules Using Genetic Programming for Multi-objective Dynamic Job Shop Scheduling with Machine Breakdowns. Procedia CIRP, 54th CIRP CMS 2021 - Towards Digitalized Manufacturing 4.0 104, 411–416. https://doi.org/10.1016/j.procir.2021.11.069

Shady, S., Kaihara, T., Fujii, N., Kokuryo, D., 2021c. A New Representation and Adaptive Feature Selection for Evolving Compact Dispatching Rules for Dynamic Job Shop Scheduling with Genetic Programming, in: Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems, IFIP Advances in Information and Communication Technology. Springer International Publishing, pp. 646–654. https://doi.org/10.1007/978-3-030-85906-0_70

Shady, S., Kaihara, T., Fujii, N., Kokuryo, D., 2021d. Evolving Dispatching Rules Using Genetic Programming for Multi-objective Dynamic Job Shop Scheduling with Machine Breakdowns. Procedia CIRP, 54th CIRP CMS 2021 - Towards Digitalized Manufacturing 4.0 104, 411–416. https://doi.org/10.1016/j.procir.2021.11.069

Shady, S., Kaihara, T., Fujii, N., Kokuryo, D., 2020a. Automatic Design of Dispatching Rules with Genetic Programming for Dynamic Job Shop Scheduling, in: IFIP International Conference on Advances in Production Management Systems. Springer, pp. 399–407. https://doi.org/10.1007/978-3-030-57993-7_45

Shady, S., Kaihara, T., Fujii, N., Kokuryo, D., 2020b. A Hyper-Heuristic Framework using GP for Dynamic Job Shop Scheduling Problem, in: Proceedings of the 64th Annual Conference of the Institute of Systems, Control and Information Engineers (ISCIE). pp. 248–252.

Shady, S., Kaihara, T., Fujii, N., Kokuryo, D., 2020c. Automatic Design of Dispatching Rules with Genetic Programming for Dynamic Job Shop Scheduling, in: IFIP International Conference on Advances in Production Management Systems. Springer, pp. 399–407. https://doi.org/10.1007/978-3-030-57993-7_45

Shady, S., Kaihara, T., Fujii, N., Kokuryo, D., 2020d. A Proposal on Dispatching Rule Generation Mechanism Using GP for Dynamic Job Shop Scheduling with Machine Breakdowns, in: Scheduling Symposium 2020. Osaka, pp. 155–160.

Shi, W., Song, X., Sun, J., 2015. Automatic Heuristic Generation with Scatter Programming to Solve the Hybrid Flow Shop Problem. Adv. Mech. Eng. 7, 587038. https://doi.org/10.1155/2014/587038

Simon, F.Y.-P., Takefuji, 1988. Integer linear programming neural networks for job-shop scheduling, in: IEEE 1988 International Conference on Neural Networks. Presented at the IEEE 1988 International Conference on Neural Networks, pp. 341–348 vol.2. https://doi.org/10.1109/ICNN.1988.23946

Stadtler, H., Kilger, C., Meyr, H., 2015. Supply chain management and advanced planning: concepts, models, software, and case studies. Springer.

Storer, R.H., Wu, S.D., Vaccari, R., 1992. New search spaces for sequencing problems with application to job shop scheduling. Manag. Sci. 38, 1495–1509.

Swiercz, A., 2017. Hyper-Heuristics and Metaheuristics for Selected Bio-Inspired Combinatorial Optimization Problems. Heuristics Hyper-Heuristics-Princ. Appl.

Taillard, E., 1993. Benchmarks for basic scheduling problems. Eur. J. Oper. Res., Project Management anf Scheduling 64, 278–285. https://doi.org/10.1016/0377-2217(93)90182-M

Tay, J.C., Ho, N.B., 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput. Ind. Eng. 54, 453–473. https://doi.org/10.1016/j.cie.2007.08.008

Teramoto, K., Morinaga, E., Wakamatsu, H., Arai, E., 2020. A Neighborhood Limitation Method for Job-Shop Scheduling Based on Simulated Annealing. システム制御情報学会論文誌 33, 171–181. https://doi.org/10.5687/iscie.33.171

Toptal, A., Sabuncuoglu, I., 2010. Distributed scheduling: a review of concepts and applications. Int. J. Prod. Res. 48, 5235–5262. https://doi.org/10.1080/00207540903121065

Vanneschi, L., Castelli, M., Silva, S., 2014. A survey of semantic methods in genetic programming. Genet. Program. Evolvable Mach. 15, 195–214. https://doi.org/10.1007/s10710-013-9210-0

Vázquez-Rodríguez, J.A., Ochoa, G., 2011. On the automatic discovery of variants of the NEH procedure for flow shop scheduling using genetic programming. J. Oper. Res. Soc. 62, 381–396. https://doi.org/10.1057/jors.2010.132

Waller, A., 2012. Witness simulation software, in: Proceedings of the Winter Simulation Conference. pp. 1–12.

Wang, B., 2018. The Future of Manufacturing: A New Perspective. Engineering 4, 722–728. https://doi.org/10.1016/j.eng.2018.07.020

Wang, Z., Zhang, J., Yang, S., 2019. An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals. Swarm Evol. Comput. 51, 100594. https://doi.org/10.1016/j.swevo.2019.100594

Weckman, G.R., Ganduri, C.V., Koonce, D.A., 2008. A neural network job-shop scheduler. J. Intell. Manuf. 19, 191–201. https://doi.org/10.1007/s10845-008-0073-9

Willis, M.-J., Hiden, H.G., Marenbach, P., McKay, B., Montague, G.A., 1997. Genetic programming: an introduction and survey of applications, in: Second International Conference On Genetic Algorithms In Engineering Systems: Innovations And Applications. Presented at the Second International Conference On Genetic Algorithms In Engineering Systems: Innovations And Applications, pp. 314–319. https://doi.org/10.1049/cp:19971199

Wong, P., Zhang, M., 2006. Algebraic simplification of GP programs during evolution, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06. Association for Computing Machinery, New York, NY, USA, pp. 927–934. https://doi.org/10.1145/1143997.1144156

Xu, B., Mei, Y., Wang, Y., Ji, Z., Zhang, M., 2021. Genetic Programming with Delayed Routing for Multiobjective Dynamic Flexible Job Shop Scheduling. Evol. Comput. 29, 75–105. https://doi.org/10.1162/evco_a_00273

Xu, J., Liu, S.-C., Zhao, C., Wu, J., Lin, W.-C., Yu, P.-W., 2019. An iterated local search and tabu search for two-parallel machine scheduling problem to minimize the maximum total completion time. J. Inf. Optim. Sci. 40, 751–766. https://doi.org/10.1080/02522667.2018.1468610

Yadav, S.L., Sohal, A., 2017. Comparative study of different selection techniques in genetic algorithm. Int. J. Eng. Sci. Math. 6, 174–180.

Yamada, T., Nakano, R., 1992. A genetic algorithm applicable to large-scale job-shop problems., in: PPSN. Presented at the Proceedings of the Second international workshop on parallel problem solving from Nature, pp. 281–290.

Yang, B., Geunes, J., 2008. Predictive–reactive scheduling on a single resource with uncertain future jobs. Eur. J. Oper. Res. 189, 1267–1283. https://doi.org/10.1016/j.ejor.2006.06.077

Yin, W.J., Liu, M., Wu, C., 2003. Learning single-machine scheduling heuristics subject to machine breakdowns with genetic programming, in: 2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings. IEEE Computer Society, pp. 1050–1055. https://doi.org/10.1109/CEC.2003.1299784

Yska, D., Mei, Y., Zhang, M., 2018. Genetic Programming Hyper-Heuristic with Cooperative Coevolution for Dynamic Flexible Job Shop Scheduling, in: Castelli, M., Sekanina, L., Zhang, M., Cagnoni, S., García-Sánchez, P. (Eds.), Genetic Programming. Springer International Publishing, Cham, pp. 306–321. https://doi.org/10.1007/978-3-319-77553-1_19

Zandieh, M., Adibi, M.A., 2010. Dynamic job shop scheduling using variable neighbourhood search. Int. J. Prod. Res. 48, 2449–2458. https://doi.org/10.1080/00207540802662896

Zhang, C., Zhou, Y., Peng, K., Li, X., Lian, K., Zhang, S., 2021. Dynamic flexible job shop scheduling method based on improved gene expression programming. Meas. Control 54, 1136–1146. https://doi.org/10.1177/0020294020946352

Zhang, F., Mei, Y., Nguyen, S., Zhang, M., 2021a. Evolving Scheduling Heuristics via Genetic Programming with Feature Selection in Dynamic Flexible Job-Shop Scheduling. IEEE Trans. Cybern. 51, 1797–1811. https://doi.org/10.1109/TCYB.2020.3024849

Zhang, F., Mei, Y., Nguyen, S., Zhang, M., 2021b. Evolving Scheduling Heuristics via Genetic Programming with Feature Selection in Dynamic Flexible Job-Shop Scheduling. IEEE Trans. Cybern. 51, 1797–1811. https://doi.org/10.1109/TCYB.2020.3024849

Zhang, F., Mei, Y., Nguyen, S., Zhang, M., 2021c. Evolving Scheduling Heuristics via Genetic Programming with Feature Selection in Dynamic Flexible Job-Shop Scheduling. IEEE Trans. Cybern. 51, 1797–1811. https://doi.org/10.1109/TCYB.2020.3024849

Zhang, F., Mei, Y., Zhang, M., 2019a. A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling, in: GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference. Association for Computing Machinery, Inc, New York, NY, USA, pp. 347–355. https://doi.org/10.1145/3321707.3321790

Zhang, F., Mei, Y., Zhang, M., 2019b. Evolving Dispatching Rules for Multi-objective Dynamic Flexible Job Shop Scheduling via Genetic Programming Hyper-heuristics, in: 2019 IEEE Congress on Evolutionary Computation (CEC). Presented at the 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 1366–1373. https://doi.org/10.1109/CEC.2019.8790112

Zhang, F., Mei, Y., Zhang, M., 2019c. A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling, in: GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference. Association for Computing Machinery, Inc, New York, NY, USA, pp. 347–355. https://doi.org/10.1145/3321707.3321790

Zhang, F., Mei, Y., Zhang, M., 2019d. A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling, in: GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference. Association for Computing Machinery, Inc, New York, NY, USA, pp. 347–355. https://doi.org/10.1145/3321707.3321790

Zhang, F., Nguyen, S., Mei, Y., Zhang, M., 2021d. Genetic Programming for Production Scheduling : An Evolutionary Learning Approach. Springer Verlag, Singapore.

Zhou, H., Feng, Y., Han, L., 2001. The hybrid heuristic genetic algorithm for job shop scheduling. Comput. Ind. Eng. 40, 191–200. https://doi.org/10.1016/S0360-8352(01)00017-1

Zhou, Y., Yang, J., Huang, Z., 2020a. Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. Int. J. Prod. Res. 58, 2561–2580. https://doi.org/10.1080/00207543.2019.1620362

Zhou, Y., Yang, J., Huang, Z., 2020b. Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. Int. J. Prod. Res. 58, 2561–2580. https://doi.org/10.1080/00207543.2019.1620362

Zhou, Y., Yang, J.J., 2019. Automatic design of scheduling policies for dynamic flexible job shop scheduling by multi-objective genetic programming based hyper-heuristic, in: Procedia CIRP. Elsevier B.V., pp. 439–444. https://doi.org/10.1016/j.procir.2019.02.118

Zhou, Y., Yang, J.J., Zheng, L.Y., 2019. Hyper-Heuristic Coevolution of Machine Assignment and Job Sequencing Rules for Multi-Objective Dynamic Flexible Job Shop Scheduling. IEEE Access 7, 68–88. https://doi.org/10.1109/ACCESS.2018.2883802

LIST OF PUBLICATIONS

Journal papers:

1. Salama Shady, Toshiya Kaihara, Nobutada Fujii, and Daisuke Kokuryo. "A novel feature selection for evolving compact dispatching rules using genetic programming for dynamic job shop scheduling." International Journal of Production Research, 2022: 1-24, https://doi.org/10.1080/00207543.2022.2053603.
2. Salama Shady, Toshiya Kaihara, Nobutada Fujii, and Daisuke Kokuryo. "Multi-objective Approach with a Distance Metric in Genetic Programming for Job Shop Scheduling" International Journal of Automation Technology, Vol.16, No.3, pp. 296-308, 2022., https://doi.org/10.20965/ijat.2022.p0296.
3. Salama Shady, Toshiya Kaihara, Nobutada Fujii, and Daisuke Kokuryo. " Feature Selection Approach for Evolving Reactive Scheduling Policies for Dynamic Job Shop Scheduling Problem Using Gene Expression Programming." International Journal of Production Research 2022, https://doi.org/10.1080/00207543.2022.2092041.

Conference papers:

4. Salama Shady, Toshiya Kaihara, Nobutada Fujii, and Daisuke Kokuryo. "A hyper-heuristic framework using GP for dynamic job shop scheduling problem." In Proceedings of the 64th Annual Conference of the Institute of Systems, Control and Information Engineers (ISCIE), pp. 248-252. 2020.
5. Salama Shady, Toshiya Kaihara, Nobutada Fujii, and Daisuke Kokuryo. "Automatic design of dispatching rules with genetic programming for dynamic

job shop scheduling." In IFIP International Conference on Advances in Production Management Systems, pp. 399-407. Springer, Cham, 2020, https://doi.org/10.1007/978-3-030-57993-7_45.

6. Salama Shady, Toshiya Kaihara, Nobutada Fujii, and Daisuke Kokuryo. "A Proposal on Dispatching Rule Generation Mechanism Using GP for Dynamic Job Shop Scheduling with Machine Breakdowns." In Scheduling Symposium 2020, pp. 155-160. 2020.
7. Salama Shady, Toshiya Kaihara, Nobutada Fujii, and Daisuke Kokuryo. "Evolving Dispatching Rules Using Genetic Programming for Multi-objective Dynamic Job Shop Scheduling with Machine Breakdowns." Procedia CIRP 104 (2021): 411-416, https://doi.org/10.1016/j.procir.2021.11.069.
8. Salama Shady, Toshiya Kaihara, Nobutada Fujii, and Daisuke Kokuryo. "A New Representation and Adaptive Feature Selection for Evolving Compact Dispatching Rules for Dynamic Job Shop Scheduling with Genetic Programming." In IFIP International Conference on Advances in Production Management Systems, pp. 646-654. Springer 2021, https://doi.org/10.1007/978-3-030-85906-0_70.
9. Salama Shady, Toshiya Kaihara, Nobutada Fujii, Daisuke Kokuryo, SURROGATE ASSISTED GENE EXPRESSION PROGRAMMING FOR AUTOMATED DESIGN OF JOB SHOP SCHEDULING RULES, Proceedings of the 2022 International Symposium on Flexible Automation -ISFA2022, pp.324-330.

Doctor Thesis, Kobe University

RESEARCH ON AUTOMATIC GENERATION OF DISPATCHING RULES USING GENETIC PROGRAMMING FOR JOB SHOP SCHEDULING PROBLEMS", 222 pages

Submitted on 7, 12, 2022

The date of publication is printed in cover of repository version published in Kobe University Repository Kernel.